UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS DE SOUSA FERNANDES

RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES BRASILEIRAS INCORPORANDO MAX-MARGIN OBJECT DETECTION E REDES NEURAIS CONVOLUCIONAIS

FORTALEZA

2019

LUCAS DE SOUSA FERNANDES

RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES BRASILEIRAS INCORPORANDO MAX-MARGIN OBJECT DETECTION E REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Yuri Lenon Barbosa Nogueira

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F399r Fernandes, Lucas de Sousa.
Reconhecimento automático de placas veiculares brasileiras incorporando Max-Margin Object Detection e redes neurais convolucionais / Lucas de Sousa Fernandes. – 2019.
71 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Computação, Fortaleza, 2019.
Orientação: Prof. Dr. Yuri Lenon Barbosa Nogueira.

1. Reconhecimento Automático de Placas Veiculares . 2. Detecção de objetos. 3. Reconhecimento de caracteres. 4. Visão Computacional. I. Título.

CDD 005

LUCAS DE SOUSA FERNANDES

RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES BRASILEIRAS INCORPORANDO MAX-MARGIN OBJECT DETECTION E REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Yuri Lenon Barbosa Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal do Ceará (UFC)

Me. Paulo Bruno de Sousa Serafim
Instituto Atlântico

AGRADECIMENTOS

Ao Prof. Dr. Yuri Lenon Barbosa Nogueira por me orientar em minha monografia.

Aos meus pais e irmãos, por toda sua paciência durante os momentos em que estive ausente pela dedicação aos estudos, sempre entendendo e me apoiando para o melhor futuro possível.

Aos colegas de laboratório que estiveram ao meu lado durante esta jornada.

Agradeço a todos os professores por terem contribuído com a minha formação profissional, acadêmica e como pessoa, em especial ao Prof. Dr. José Gilvan Rodrigues Maia e ao Prof. Dr. Yuri Lenon Barbosa Nogueira por terem me ajudado a encontrar meu caminho enquanto profissional.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

“Por Frodo.”

(Passo Largo)

RESUMO

O Reconhecimento Automático de Placas Veiculares (ALPR, *Automatic License Plate Recognition*) consiste em identificar e ler o texto de placas de um ou mais veículos em uma imagem. Os sistemas de ALPR estão presentes em diversas aplicações, como fiscalização de trânsito e controle de acesso a estacionamentos. Em geral, um sistema ALPR é composto de três fases: detecção da placa veicular, segmentação dos caracteres e reconhecimentos dos caracteres segmentados. As soluções envolvendo sistemas que lidam com as peculiaridades das placas veiculares brasileiras podem apresentar resultados mais específicos ou utilizam técnicas de alto custo computacional para conseguir tratar ambientes menos controlados. Diante disso, este trabalho elabora uma proposta de um sistema que busca ser compatível com dados do mundo real utilizando técnicas mais simples, como processamento de imagens, e outras mais complexas mas não muito custosas, como *Max Margin Object Detection* e redes neurais convolucionais simples. O sistema proposto tem resultados promissores, em especial na etapa de detecção da placa veicular e na classificação de caracteres, encontrando maiores dificuldades na etapa de segmentação de caracteres.

Palavras-chave: Reconhecimento Automático de Placas Veiculares. ALPR. Detecção de objetos. Reconhecimento de caracteres.

ABSTRACT

Automatic License Plate Recognition (ALPR) consists in identifying and reading license plate texts of a least one of vehicles in a image. The ALPR systems are present in various applications, such as traffic fiscalization and parking lot access control. In general, an ALPR system is composed by three stages: license plate detection, character segmentation and segmented characters recognition. The solutions involving systems that deals with brazillian license plate particularities may show more especific results or use high computational cost techniques in order to hanle less controlled environments. Thus, this work proposes of a system that seeks to be compatible with real world data using simple image processing techniques and also more complex but not too costly techniques, such as Max-Margin Object Detection and simple convolutional neural networks. The proposed system shows promising results, specially in license plate detection stage and character classification, finding its biggest dificulties in character segmentation stage.

Keywords: Automatic License Plate Recognition. ALPR. Object detection. Character recognition.

LISTA DE FIGURAS

Figura 1 – Imagem e seu histograma 18
Figura 2 – Exemplo de equalização de histograma 18
Figura 3 – Imagem binária 19
Figura 4 – Aplicação do operador de Laplace para detecção de arestas 22
Figura 5 – Aplicação da máscara de aguçamento obtida através da soma do resultado do operador de Laplace com a imagem original 23
Figura 6 – Elementos estruturantes (abaixo com as bordas vazias e acima sem bordas) . 24
Figura 7 – Aplicação de erosão e dilatação numa imagem binária, onde (a) é a imagem original e (b) e (c) são a erosão e a dilatação, respectivamente 24
Figura 8 – Exemplo de fechamento 25
Figura 9 – Processo de obtenção do HOG para uma imagem 26
Figura 10 – HOG de uma face 27
Figura 11 – Exemplo de conjunto de treinamento para aprendizagem de máquina, onde os pontos azuis são de uma classe e os pontos vermelhos de outra 27
Figura 12 – Exemplo de regressão linear, onde os pontos vermelhos pertencem ao conjunto de dados do treinamento, a reta azul é a função com parâmetros estimados pelo método dos mínimos quadrados e x' é um dado de teste cuja saída estimada pela regressão linear é y' 28
Figura 13 – Modelo de neurônio artificial (Perceptron) 29
Figura 14 – Modelo de MLP 30
Figura 15 – Arquitetura da rede neural convolucional LeNet-5 31
Figura 16 – Exemplo de operação de max pooling com filtro de tamanho 2×2, onde o espaço colorido é correspondente ao filtro naquele espaço 32
Figura 17 – Três janelas deslizantes e suas avaliações. O método guloso escolheria apenas a janela do centro, com 7 de score, enquanto um método otimizado escolheria as duas janelas exteriores, totalizando 12 de score. 34
Figura 18 – Etapas para detecção da placa 37
Figura 19 – HOG da placa 37
Figura 20 – Ilustração da visão frontal (VF): o retângulo amarelo tracejado representa uma VF marcada manualmente enquanto o retângulo vermelho é a VF estimada baseada na placa (em azul) 38

Figura 21 – Amostras da base UFPR-ALPR: as primeiras três linhas mostram a variedade em ambiente e veículo, bem como a variação nas placas, enquanto a última linha mostra anotações de veículo e placa; as placas estão embaçadas por questões de privacidade. 39
Figura 22 – Primeira fase da segmentação de caracteres 40
Figura 23 – Segunda fase da segmentação de caracteres 41
Figura 24 – Exemplos do processo de segmentação de caracteres 42
Figura 25 – Topologia da rede MLP utilizada para classificar caracteres 42
Figura 26 – Exemplos de reconhecimento dos caracteres da placa 43
Figura 27 – Exemplos de leitura das placas 44
Figura 28 – Exemplos de leitura das placas 46
Figura 29 – Diagrama resumindo o método ALPR proposto 47
Figura 30 – Amostra da base UFPR-ALPR (com placa borrada por questões de privacidade) 48
Figura 31 – Amostras da base UFPR-ALPR exemplificando a variação de veículos de ambientes 49
Figura 32 – HOG de placa 49
Figura 33 – Exemplo de placa detectada pelo método MMOD (com a placa borrada por questões de privacidade dos dados) 50
Figura 34 – Aplicação de equalização de histograma em uma imagem de placa 51
Figura 35 – Aguçamento de uma imagem de placa 51
Figura 36 – Imagem de placa binarizada através do método de Otsu 52
Figura 37 – Imagem resultante de um fechamento em uma imagem binarizada de placa . 52
Figura 38 – Segmentação inicial em uma imagem de placa 53
Figura 39 – Resultado de uma regressão linear aplicada nas posições dos caracteres . . . 53
Figura 40 – Imagem binária de placa com inclinação corrigida 53
Figura 41 – Corte na imagem binária de placa 54
Figura 42 – Imagem dos caracteres da placa e sua respectiva projeção horizontal 54
Figura 43 – Imagem de placa com seus caracteres segmentados 54
Figura 44 – Aplicação de equalização de histograma em uma imagem de caractere . . . 55
Figura 45 – Imagem binária de um caractere 55
Figura 46 – Exemplo de imagem utilizada para analisar o método de detecção de placas 58
Figura 47 – Exemplo de imagem com falsos positivos detectados 60

Figura 48 – Exemplos de segmentação de caracteres 61
Figura 49 – Ajuste automático de taxa de aprendizagem: taxa de aprendizagem por épocas 62
Figura 50 – Evolução da aprendizagem 63

LISTA DE TABELAS

Tabela 1 – Arquitetura da rede para reconhecimento de dígitos 56
Tabela 2 – Arquitetura da rede para reconhecimento de letras 56
Tabela 3 – Resultados da detecção de placas veiculares considerando a base secundária de imagens 59
Tabela 4 – Avaliação do classificador de dígitos 62
Tabela 5 – Avaliação do classificador de letras 62
Tabela 6 – Resultados individuais por dígito (com valores de 0 a 1) 64
Tabela 7 – Resultados individuais por letra (de 0 a 1) 64
Tabela 8 – Resultado da leitura das placas com o sistema proposto 65
Tabela 9 – Comparação dos resultados em diferentes métodos 65

LISTA DE ABREVIATURAS E SIGLAS

ALPR	Automatic Licence Plate Recognition
CCA	Análise de Componentes Conectados ou *Connected Components Analysis*
CNN	Rede Neural Convolucional ou *Convolutional Neural Network*
GPU	Unidade de Processamento Gráfico ou *Graphics Processing Unit*
HOG	Histograma de Gradientes Orientados ou *Histogram of Oriented Gradients*
IoU	Intersection Over Union
MLP	Perceptron de Múltiplas Camadas ou *Multilayer Perceptron*
MMOD	Max-Margin Object Detection
RNA	Rede Neural Artificial
SVM	Máquina de Vetores de Suporte ou*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO 15
1.1 Objetivos 15
1.1.1 Objetivos específicos 15
1.2 Metodologia 16
1.3 Organização do trabalho 16
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Pré-processamento digital de imagens 17
2.1.1 Equalização de histograma 17
2.1.2 Binarização 18
2.1.2.1 Método de binarização de Otsu *19*
2.1.3 Filtragem espacial 20
2.1.3.1 Aguçamento de imagens utilizando filtragem espacial *20*
2.1.4 Morfologia matemática 22
2.1.4.1 Dilatação e erosão *24*
2.1.4.2 Fechamento *25*
2.1.5 Histograma de Gradientes Orientados (HOG) 25
2.2 Aprendizagem de máquina 27
2.2.1 Regressão Linear 28
2.2.2 Redes Neurais Artificiais 28
2.2.2.1 Perceptron e Multilayer Perceptron *29*
2.2.2.2 Aprendizado por gradiente de backpropagation *30*
2.2.2.3 Deep Learning e as Redes Neurais Convolucionais *31*
2.3 Detecção de objetos 32
2.3.1 Max-Margin Object Detection (MMOD) 33
3 TRABALHOS RELACIONADOS 36
3.1 Detecção de placas 36
3.2 Segmentação dos caracteres 40
3.3 Leitura dos caracteres 42
4 PROPOSTA DE UM MÉTODO ALPR 47
4.1 Analisando a base de dados 47

4.2 **Detecção de placas veiculares** 48
4.2.1 ***Max-Margin Object Detection (MMOD) com HOG*** 49
4.3 **Segmentação de caracteres** 50
4.3.1 ***Equalização de Histograma*** 50
4.3.2 ***Aguçamento*** 50
4.3.3 ***Binarização*** 51
4.3.4 ***Fechamento*** 51
4.3.5 ***Eliminação de falsos caracteres e correção da inclinação da placa*** 52
4.3.6 ***Segmentação final*** 53
4.4 **Classificação de caracteres e reconhecimento da placa** 54
4.4.1 ***Equalização de histograma*** 55
4.4.2 ***Binarização*** 55
4.4.3 ***Classificação de dígitos e letras*** 56
5 **RESULTADOS** 58
5.1 **Detecção de placas veiculares** 58
5.2 **Segmentação dos caracteres** 60
5.3 **Classificação dos caracteres** 61
5.4 **Leitura da placa** 65
6 **CONCLUSÃO** 66
6.1 **Considerações gerais** 66
6.2 **Trabalhos Futuros** 66
REFERÊNCIAS 68
APÊNDICES 71
APÊNDICE A – Repositório com o código e modelos do trabalho 71
ANEXOS 71

1 INTRODUÇÃO

O Reconhecimento Automático de Placas Veiculares, ou *Automatic Licence Plate Recognition (ALPR)*, trata de, dada uma imagem possivelmente contendo um ou mais veículos, identificar e ler a placa de cada veículo presente na imagem. Os sistemas de ALPR estão presentes em um número importante de aplicações úteis, como monitoramento de trânsito, coleta automática de pedágio, fiscalização e aplicação de leis de trânsito, controle de acesso de estacionamento e etc. Em geral, esses sistemas utilizam de três etapas: detecção das placas veiculares, segmentação dos caracteres das placas e reconhecimento dos caracteres segmentados (DU *et al.*, 2012).

Em cada país, as placas veiculares têm suas próprias particularidades, com regras e padrões estabelecidos. No Brasil, o mais comum, por ser o padrão particular vigente desde 1990 (VOGEL, 2018), é que as placas veiculares sejam cinza com sete caracteres alfanuméricos pretos, sendo os três primeiros letras e os últimos quatro números.

Além de lidar com essas peculiaridades, um sistema ALPR precisa tratar das condições das imagens recebidas, como iluminação variável, presença ou não de sombras ou borrões, inclinações e etc. Quanto menos controlado é o ambiente das imagens, mais similar ao mundo real o ambiente é. Diante disso, sistemas mais robustos utilizam de técnicas com custos computacionais elevados, exigindo equipamentos mais fortes.

Nesse sentido, este trabalho tem como objetivo elaborar um novo sistema de ALPR para placas brasileiras, que possa ser similar a dados do mundo real sem custos computacionais muito elevados.

1.1 Objetivos

O presente trabalho tem como objetivo principal propor um novo método para reconhecimento automático de placas veiculares brasileiras que seja compatível com dados do mundo real e utilize as técnicas mais recentes da literatura, como *deep learning*, sem exigir custos computacionais muito elevados.

1.1.1 Objetivos específicos

- Analisar diferentes sistemas ALPR em placas veiculares brasileiras.
- Elaborar um novo sistema ALPR para placas veiculares brasileiras, envolvendo:

  - Desenvolver um detector de placas veiculares brasileiras robusto para imagens similares ao mundo real.
  - Desenvolver um método para leitura das placas veiculares que incorpore *deep learning* sem custos computacionais muito elevados.
- Discutir os resultados do método ALPR proposto.

1.2 Metodologia

Para compreender detalhadamente o problema, foram estudados trabalhos recentes da literatura com propostas similares, entendendo a peculiaridade das placas veiculares brasileiras e possíveis soluções. Após isso, iniciou-se o estudo das tecnologias e métodos que poderiam ser utilizados, formando o alicerce teórico necessário para proposição de um novo método.

Enfim foi desenvolvido um protótipo de um novo sistema ALPR, de forma a abranger ambientes não-controlados e dados mais similares ao mundo real sem usar soluções muito complexas computacionalmente, e também foi realizado um estudo comparativo entre as técnicas conhecidas na literatura e os métodos propostos.

1.3 Organização do trabalho

Os capítulos deste trabalho estão divididos de forma a primeiro fundamentar as bases teóricas utilizadas para o desenvolvimento do método proposto no Capítulo 2 para então descrever detalhadamente o método proposto no Capítulo 4 e então discutir os resultados obtidos no Capítulo 5, comparando com os trabalhos relacionados ao tema apresentados no Capítulo 3, e então o Capítulo 6 apresenta as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém o arcabouço teórico necessário para o entendimento do método de ALPR proposto neste trabalho.

2.1 Pré-processamento digital de imagens

Uma *imagem* pode ser definida como uma função $f(x,y)$, onde x e y são coordenadas num plano e $f(x,y)$ é a intensidade (ou nível de cinza, em imagens monocromáticas) naquelas coordenadas. Quando os valores de x, y e da amplitude de f são valores discretos finitos, a imagem é dita como *imagem digital*. Deste modo, uma imagem digital pode ser descrita como uma matriz I com M linhas e N colunas, sendo x e y, as coordenadas da matriz, convencionalmente números inteiros variando de 0 ao tamanho do eixo da coordenada. Os elementos de uma imagem, nesse caso $I(x,y)$ com $x < N$ e $y < M$, são geralmente denominados de *pixel* e variam de 0, ausência total de intensidade (ou cor), e 255, nível máximo de intensidade (GONZALEZ; WOODS, 2008).

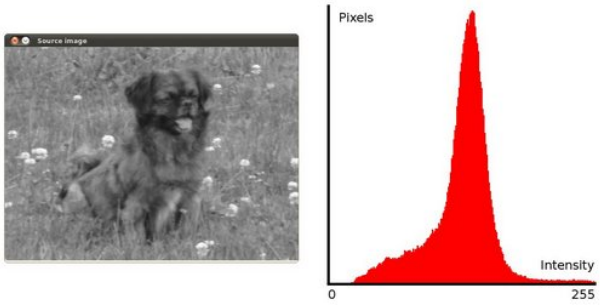
A partir desses conceitos, podem ser realizados diversos processos em imagens, operações sobre os pixels da imagens, a fim de obter novas imagens, coletar características, segmentar objetos, entre outros objetivos. Deste modo, o processamento digital de imagens engloba processos que tem como entrada e saída imagens e processos que extraem atributos de imagens (GONZALEZ; WOODS, 2008). As subseções seguintes abordam alguns desses processos.

2.1.1 Equalização de histograma

O histograma de uma imagem digital I de tamanho $M \times N$ e L níveis de cinza é definido como a função $h(r_k) = n_k$ onde r_k é uma intensidade de pixel e n_k é o número de pixels cuja intensidade é r_k, com k variando de 0 a $L-1$ (GONZALEZ; WOODS, 2008). A Figura 1 mostra uma imagem e seu histograma.

A equalização de histogramas é uma operação que uniformiza o nível de cinza (ou intensidade) da imagem, resultando numa melhoria do contraste. Considerando a imagem I anteriormente definida, essa operação consiste em mapear cada intensidade r_k para a intensidade

Figura 1 – Imagem e seu histograma



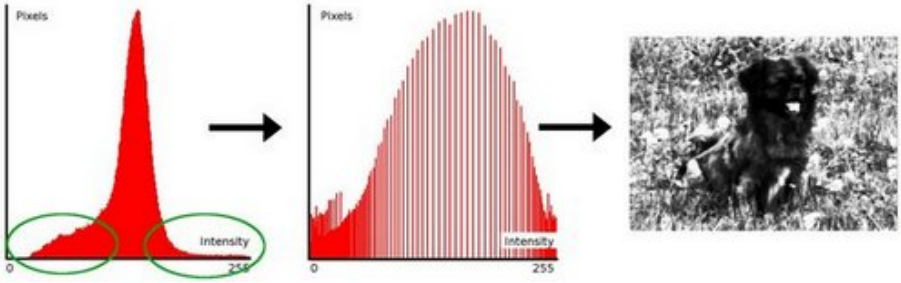
Fonte: OpenCV (2019a)

s_k descrita na seguinte equação (GONZALEZ; WOODS, 2008):

$$s_k = \frac{(L-1)}{MN} \sum_{j=0}^{k} n_j, \forall k \in \{0, ..., L-1\} \tag{2.1}$$

A Figura 2 mostra um exemplo de equalização de histograma.

Figura 2 – Exemplo de equalização de histograma



Fonte: OpenCV (2019a)

Esse método tem como vantagem de ser automático, utilizando como informação apenas os níveis de cinza da imagem, sem a necessidade de especificar outros parâmetros (GONZALEZ; WOODS, 2008).

2.1.2 Binarização

Imagens binárias são descritas como imagens de intensidade binária (preto ou branco) e o processo de *binarização* é constituído por uma operação que transforma a intensidade de um pixel em 1 (branco, indicando presença de um elemento), se a intensidade $I(x,y)$ desse pixel for maior que um dado limite T, ou 0 (preto, ausência de elemento) caso contrário (GONZALEZ; WOODS, 2008). Quando se usa uma informação global para encontrar o valor do limite T, diz-se que a binarização é *global*. O objetivo da binarização é segmentar um ou mais objetos do fundo da imagem (*background*) (ZAITOUN; AQEL, 2015). A Figura 3 mostra um exemplo de imagem binária.

Figura 3 – Imagem binária

Fonte: OpenCV (2019b)

2.1.2.1 Método de binarização de Otsu

A binarização ótima de imagens pode ser considerado um problema estatístico de decisão onde o objetivo é minimizar o erro em atribuir os pixels a duas classes (preto e branco). Dessa forma, encontrar um valor de limite ótimo é um problema complexo, de solução não trivial (GONZALEZ; WOODS, 2008). Uma solução alternativa é o método proposto por Otsu (1979), convencionalmente denominado de método de Otsu. Ele é considerado ótimo porque maximiza a variância entre as classes de elementos e tem a vantagem de depender apenas de cálculos no histograma unidimensional da imagem (GONZALEZ; WOODS, 2008). A ideia geral do método é que duas classes bem separadas devem ser distintas em relação às intensidades de seus pixels e que o limite que provê a melhor separação das classes é o melhor limite, ou seja, o limite ótimo.

Considere $\{0,1,2,..,L-1\}$ os L níveis de intensidade de uma imagem digital I de $M \times N$ pixels e n_i o número de pixels com intensidade i. O histograma normalizado dessa imagem é composto pelos valores $p_i = n_i/MN$ com $i = \{0,...,L-1\}$, $0 \leq x < N$ e $0 \leq y < M$. Considerando k como o limite que separa as classes 0 e 1, a probabilidade de um pixel pertencer a classe 0 pode ser descrita como a seguinte soma acumulativa.

$$P_0(k) = \sum_{i=0}^{k} p_i \tag{2.2}$$

Dessa forma, a probabilidade de um pixel pertencer a classe 1 seria:

$$P_1(k) = 1 - P_0(k) \tag{2.3}$$

A média de intensidade dos pixels das classes 0 e 1 é dada por, respectivamente:

$$m_0(k) = \frac{1}{P_0(k)} \sum_{i=0}^{k} i p_i \tag{2.4}$$

e

$$m_1(k) = \frac{1}{P_1(k)} \sum_{i=k+1}^{L-1} i p_i \tag{2.5}$$

A média acumulativa até um limite k é dada por:

$$m(k) = \sum_{i=0}^{k} ip_i \tag{2.6}$$

A média global da intensidade da imagem é dada por:

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{2.7}$$

A variância entre as classes é definida como:

$$\sigma^2(k) = P_0(k)(m_0(k) - m_G)^2 + P_1(k)(m_1(k) - m_G)^2 \tag{2.8}$$

A equação 2.8 pode ser escrita como:

$$\sigma^2(k) = P_0(k)P_1(k)(m_0(k) - m_1(k))^2 = \frac{(m_G(k)P_0(k) - m(k))^2}{P_0(k)(1 - P_0(k))} \tag{2.9}$$

Dessa forma, o método de Otsu consiste em encontrar a intensidade de pixel k^* tal que $\sigma^2(k^*)$ é máximo.

2.1.3 Filtragem espacial

Alguns processos em imagens digitais utilizam de informação local, das vizinhanças de um pixel, para, por exemplo, remover ruídos ou acentuar contornos, resultando numa nova imagem. O tipo mais comum de operador de vizinhança é o *filtro espacial* (ou *filtro linear*), em que o valor de saída de um pixel é determinado pela soma ponderada dos pixels de entrada. Essa operação é denominada *convolução*, que pode ser descrita na equação seguinte.

$$g(i,j) = \sum_{k,l} f(k,l)h(i-k, j-l) \tag{2.10}$$

Na Equação 2.10, f é a imagem de entrada, g é a imagem resultante e h é o que se denomina de *máscara de convolução* (SZELISKI, 2010). Essa operação pode ser resumida como:.

$$g = f * h \tag{2.11}$$

2.1.3.1 Aguçamento de imagens utilizando filtragem espacial

O *aguçamento* de imagens tem como objetivo destacar mudanças de intensidade, aprimorando arestas e outras descontinuidades e atenuando áreas mais regulares. Essa operação

pode ser considerada análoga à diferenciação e implementada utilizando filtragem espacial (GONZALEZ; WOODS, 2008).

A derivada em uma função $f(x)$ unidimensional pode ser definida na seguinte diferença.

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{2.12}$$

Do mesmo modo, a derivada de segunda ordem pode ser definida como:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x) \tag{2.13}$$

Em imagens, contornos podem ser analisados como transições de intensidade. Dessa forma, a derivada primeira numa imagem resultaria em arestas espessas, devido ao caráter dessas transições, mas a derivada segunda resultaria em arestas mais finas separadas por zeros (GONZALEZ; WOODS, 2008). Baseando-se nisso, uma abordagem para detectar as arestas consiste na utilização do operador de Laplace (ROSENFELD; KAK, 1982), definido na equação a seguir, considerando $f(x,y)$ como uma imagem (ou matriz bidimensional).

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y} \tag{2.14}$$

Para expressar essa operação em valores discretos, a Equação 2.14 pode ser usada em f em x e em y, resultando em:

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{2.15}$$

Essa operação pode ser implementada utilizando filtragem linear, através da máscara de convolução representada pela seguinte matriz.

$$H_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{2.16}$$

É comum que nas implementações dessas operações se utilize a matriz com o pixel central positivo e outra alteração relevante é a inclusão dos eixos diagonais na máscara (GONZALEZ; WOODS, 2008). Deste modo, aplicando as duas alterações se obtém:

$$H_2 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{2.17}$$

Figura 4 – Aplicação do operador de Laplace para detecção de arestas

(a) Imagem original (b) Resultado da operação de Laplace

Fonte: elaborado pelo autor.

O resultado das convoluções pode ser observado na Figura 4.

Como foi mencionado anteriormente, o operador de Laplace tem como finalidade a detecção de arestas, mas pode ser utilizado para o aguçamento de imagens. Nesse caso, o aguçamento seria definido como a soma da imagem original com as arestas obtidas pelo operador de Laplace (GONZALEZ; WOODS, 2008). Desse modo, uma máscara convolucional utilizada para aguçar imagens pode ser calculada somando o pixel central com o pixel o resultante da convolução com a máscara da operação de Laplace. Esse processo pode ser resumido numa só máscara de convolução, descrita a seguir.

$$H_3 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{2.18}$$

O resultado do aguçamento de uma imagem utilizando essa máscara pode ser observado na Figura 5.

2.1.4 Morfologia matemática

Alguns processos são usados extensivamente em imagens binárias, dentre eles as operações mais comuns são as ditas *operações morfológicas*, pois elas podem mudar a forma dos objetos nas imagens (SZELISKI, 2010). A *morfologia matemática* (ou simplesmente *morfologia*) pode ser considerada nesse contexto como uma ferramenta para extrair componentes de imagens úteis para a descrição de formas de regiões e também como um conjunto de técnicas para pré e pós-processamento (GONZALEZ; WOODS, 2008).

Figura 5 – Aplicação da máscara de aguçamento obtida através da soma do resultado do operador de Laplace com a imagem original

(a) Imagem original (b) Imagem aguçada

Fonte: elaborado pelo autor.

A linguagem da morfologia é a teoria de conjuntos, que nesse contexto representam objetos em uma imagem. Em imagens binárias, esses conjuntos são membros do espaço bidimensional de inteiros Z^2, onde os elementos dos conjuntos são tuplas cujas coordenadas (x,y) são as coordenadas de pixels brancos na imagem. Além das operações básicas de conjuntos, reflexões e translações são essenciais na morfologia. A reflexão de um conjunto B, denotada $\hat{B}$, é definida como:

$$\hat{B} = \{w | w = -b \quad \forall b \in B\} \tag{2.19}$$

E a translação de um conjunto B num ponto $z = (z_1, z_2)$, denotada $(B)_z$, é definida como:

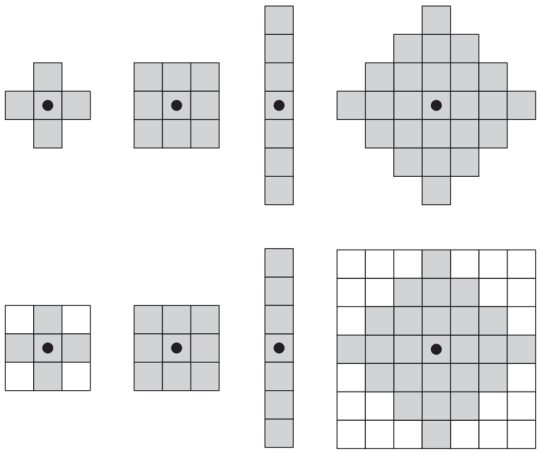
$$(B)_z = \{c | c = b + z, \quad \forall b \in B\} \tag{2.20}$$

Reflexões e translações de conjuntos são utilizados extensivamente na morfologia para formular operações baseadas nos chamados *elementos estruturantes*: pequenos subconjuntos (ou subimagens) binários usados para examinar uma imagem a procura de propriedades de interesse (GONZALEZ; WOODS, 2008). A Figura 6 contém exemplos de elementos estruturantes.

Outra operação importante de ser mencionada é a do *complemento*, que consiste em inverter a imagem binária, ou seja, o complemento de um conjunto A é definido como:

$$A = \{a | a \notin A\} \tag{2.21}$$

Figura 6 – Elementos estruturantes (abaixo com as bordas vazias e acima sem bordas)



Fonte: Gonzalez e Woods (2008)

2.1.4.1 Dilatação e erosão

Duas aplicações comuns da morfologia matemática são afinar ou engrossar objetos em imagens. Para isso, utiliza-se das operações de erosão e dilatação. Uma *erosão* no conjunto A com elemento estruturante B é definida como (GONZALEZ; WOODS, 2008):

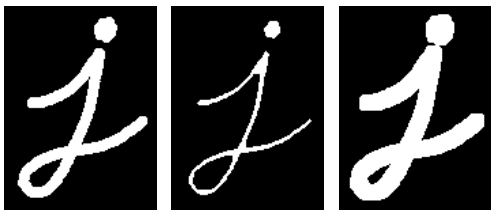
$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{2.22}$$

E uma *dilatação* no conjunto A com elemento estruturante B é definida como (GONZALEZ; WOODS, 2008):

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \emptyset\} \tag{2.23}$$

A Figura 7 mostra um exemplo de dilatação e erosão numa imagem binária.

Figura 7 – Aplicação de erosão e dilatação numa imagem binária, onde (a) é a imagem original e (b) e (c) são a erosão e a dilatação, respectivamente



(a) (b) (c)

Fonte: OpenCV (2019b)

2.1.4.2 Fechamento

A partir dos conceitos de dilatação, que engrossa objetos, e erosão, que afina objetos, pode-se conceituar um *fechamento*, que pode ser definido, dado conjunto A e elemento estruturante B, como:

$$A \bullet B = (A \oplus B) \ominus B \tag{2.24}$$

Geralmente, um fechamento suaviza contornos, une vãos estreitos, elimina pequenos buracos e preenche vazios nos contornos (GONZALEZ; WOODS, 2008). A Figura 8 ilustra uma abertura numa imagem binária.

Figura 8 – Exemplo de fechamento

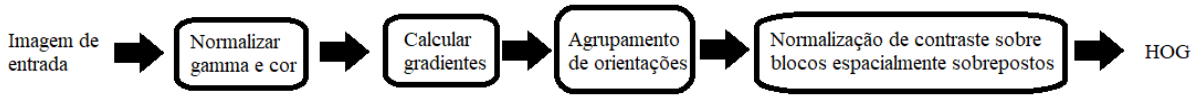
Fonte: OpenCV (2019b)

2.1.5 Histograma de Gradientes Orientados (HOG)

Uma das utilidades do processamento de imagens é ser uma etapa anterior a métodos robustos de inteligência artificial, que utilizariam de entrada características descritas pelo pré-processamento. Nesse sentido, Dalal e Triggs (2005) propuseram o Histograma de Gradientes Orientados ou *Histogram of Oriented Gradients* (HOG). Proposto inicialmente para a detecção de pedestres, o HOG é um descritor de características e que devido aos seus bons resultados foi usado posteriormente para atingir diversos problemas de detecção, inclusive para detecção e reconhecimento de placas veiculares (SALAZAR *et al.*, 2017). Em resumo, o método consiste em avaliar histogramas locais normalizados de orientações de gradientes da imagem a ser descrita em uma grade densa. A ideia básica é que a aparência e formato de objetos locais podem frequentemente serem caracterizados bem pela distribuição de gradientes de intensidade local ou direção de arestas, mesmo sem conhecimento preciso das posições de gradiente ou aresta correspondentes. Na prática isso é implementado dividindo a imagem em pequenas regiões de espaço, chamadas *células*, e cada célula acumulando um histograma local de direções

de gradiente ou orientações de arestas sobre os pixels da célula. Os histogramas combinados formam o vetor de características da imagem (DALAL; TRIGGS, 2005). A Figura 9 ilustra esse processo.

Figura 9 – Processo de obtenção do HOG para uma imagem



Fonte: elaborado pelo autor.

A normalização de gamma é descrita na seguinte equação, para cada pixel da imagem, sendo p' o pixel normalizado, p o pixel original e γ e c constantes (GONZALEZ; WOODS, 2008).

$$p' = cp^{\gamma} \tag{2.25}$$

O cálculo dos gradientes horizontal e vertical é realizado através da filtragem linear utilizando as máscaras descritas a seguir, sendo H_y para o gradiente vertical e H_x para o gradiente horizontal.

$$H_y = \begin{pmatrix} -1 \\ -0 \\ 1 \end{pmatrix}, \qquad H_x = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \tag{2.26}$$

Após a obtenção desses valores, sendo g_x o gradiente vertical e g_y o gradiente horizontal, a magnitude g e a inclinação θ do gradiente podem ser calculados como:

$$g = \sqrt{g_x^2 + g_y^2}, \qquad \theta = \arctan\left(\frac{g_y}{g_x}\right) \tag{2.27}$$

A imagem é então dividida em células de tamanho 8×8 o histograma é divido em *bins*, intervalos fixos, de ângulos. Para cada pixel é computado g votos no bin do histograma da célula mais próxima à inclinação correspondente. Enfim, as células são agrupadas em blocos e os histogramas dos blocos são normalizados pela norma L2, descrita como:

$$v' = \frac{v}{\sqrt{||v||_2^2 + c^2}} \tag{2.28}$$

Onde v' é o vetor do histograma normalizado, v o vetor original, $||v||_2$ a norma-2 do vetor e c uma constante pequena. Após a normalização, os vetores são combinados compondo o vetor de características final. A Figura 10 mostra graficamente o HOG, resultado final desse processo, numa face.

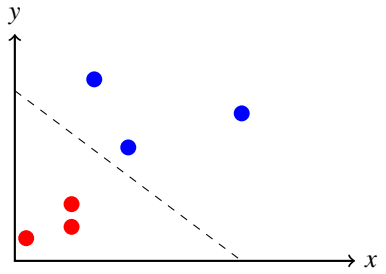
Figura 10 – HOG de uma face

Fonte: King (2015)

2.2 Aprendizagem de máquina

Muitos algoritmos que compartilham o objetivo de extrair informação de dados podem ser classificados como *aprendizagem de máquina*, ou *machine learning*. Entretanto, algoritmos de aprendizagem de máquina não se resumem a sumarizar dados, mas sim como aprender um modelo ou classificador dos dados e, então, descobrir algo sobre os dados (RAJARAMAN *et al.*, 2014). O conjunto de dados utilizado para a aprendizagem é denominado *conjunto de treinamento*, ilustrado por um exemplo simples na Figura 11. Outros conjuntos de dados são utilizados para avaliar o modelo ou classificador resultante. Esses conjuntos são, em geral, *conjuntos de validação e teste*, sendo o primeiro utilizado para aprimorar os parâmetros do algoritmo de treinamento e o segundo para avaliar o que foi aprendido simulando dados do mundo real.

Figura 11 – Exemplo de conjunto de treinamento para aprendizagem de máquina, onde os pontos azuis são de uma classe e os pontos vermelhos de outra



Fonte: elaborado pelo autor.

2.2.1 *Regressão Linear*

Um dos métodos mais simples que podem ser classificados como aprendizagem de máquina é a *regressão linear*, em que são observadas N amostras da variável de saída y_i e p atributos dos preditores $x_i = (x_{i1}, ..., x_{ip})$. O objetivo é predizer a saída dos preditores, tanto para realmente ter predição com dados futuros, quanto para descobrir que preditores são importantes (TIBSHIRANI *et al.*, 2015). Um modelo de regressão linear assume que:

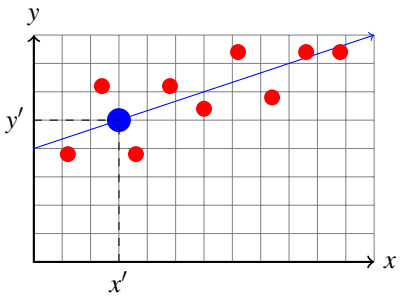
$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ik}\beta_j + e_i \tag{2.29}$$

onde β_0 e $\beta = \{\beta_1, ..., \beta_p\}$ são variáveis desconhecidas e e_i é o erro por termo. O método dos mínimos quadrados provê estimativas dos parâmetros minimizando a seguinte função objetiva dos mínimos quadrados (TIBSHIRANI *et al.*, 2015).

$$\min_{\beta_0,\beta}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ik}\beta_j)^2 \tag{2.30}$$

A Figura 12 ilustra um exemplo em um pequeno conjunto de dados de duas dimensões, onde a reta é a função, dada pela Equação 2.29, com parâmetros estimados pelo método dos mínimos quadrados, como mostra a Equação 2.30.

Figura 12 – Exemplo de regressão linear, onde os pontos vermelhos pertencem ao conjunto de dados do treinamento, a reta azul é a função com parâmetros estimados pelo método dos mínimos quadrados e x' é um dado de teste cuja saída estimada pela regressão linear é y'



Fonte: elaborado pelo autor.

2.2.2 *Redes Neurais Artificiais*

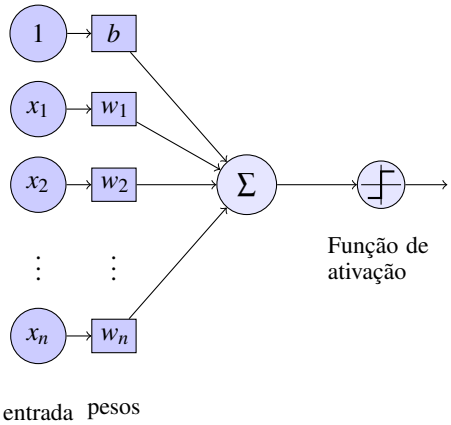
Inspirado em modelos biológicos, Rosenblatt (1957) propôs um método de aprendizagem de máquina chamado *Perceptron*, que é o modelo mais simples de uma *Rede Neural Artificial (RNA)*. Desde então, as Redes Neurais Artificiais vem sendo estudadas como alternativas para substituir o trabalho e processamento manual para obtenção de características por redes

multicamadas treináveis. Essas redes podem ser treinadas através dos *gradientes descendentes estocásticos*, que são calculados utilizando o algoritmo de retropropagação, ou *backpropagation* (LECUN *et al.*, 2015).

2.2.2.1 Perceptron e Multilayer Perceptron

As redes Perceptron podem ser denominadas de *neurônio artificial* e são definidas como um classificador linear binário com entrada $x = \{x_1, ..., x_d\}$, pesos $w = \{w_1, ..., w_d\}$, um *bias* b e uma função de ativação f para produzir uma saída y' binária de ativação, de modo que $y' = f(x * w + b)$ (RAJARAMAN *et al.*, 2014). A Figura 13 mostra um modelo de neurônio artificial. A ideia geral do Perceptron é que w define um hiperplano que separa as duas classes

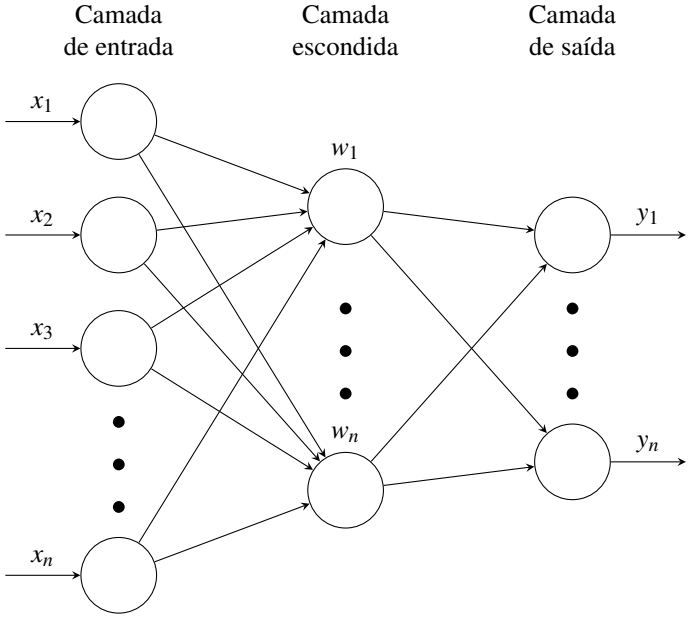
Figura 13 – Modelo de neurônio artificial (Perceptron)



Fonte: elaborado pelo autor.

de dados. Dessa forma, o Perceptron só converge se os dados forem linearmente separáveis, ou seja, se houver algum hiperplano que separe esses dados. Se não existe nenhum hiperplano que separe as classes, o Perceptron não define nenhum e não converge (RAJARAMAN *et al.*, 2014). Para contornar essa limitação, surgiu o *Perceptron de Múltiplas Camadas ou Multilayer Perceptron (MLP)*, que é composta de uma camada de neurônios de entrada, uma camada de neurônios de saída e pelo menos uma camada escondida, como mostra a Figura 14. Para cada camada escondida, tem-se os pesos w da camada e uma função de ativação, também presente na camada de saída. Através do algoritmo backpropagation, a ser descrito a seguir, os pesos são atualizados no treinamento para a MLP predizer melhor os dados (LECUN *et al.*, 2015).

Figura 14 – Modelo de MLP



Fonte: elaborado pelo autor.

2.2.2.2 Aprendizado por gradiente de backpropagation

O aprendizado da rede é divido em etapas chamadas épocas, onde os dados vão da camada de entrada até a saída e o erro é propagado da saída até a entrada. O cálculo do erro é feito sob as denominadas *funções de erro* ou *funções de loss*. A função de erro E^p mede a discrepância entre o valor desejado de saída e o valor de saída real da rede e a função de erro média E é a média entre os erros E^p para um dado conjunto. A técnica que leva a rede a aprender é o método de otimização *gradiente descendente estocástico*, que consiste em atualizar os pesos a partir do gradiente seguindo a seguinte equação para cada época k (LECUN *et al.*, 1998).

$$w_k = w_{k-1} - \varepsilon \frac{\partial E^{p_k}(K)}{\partial W} \tag{2.31}$$

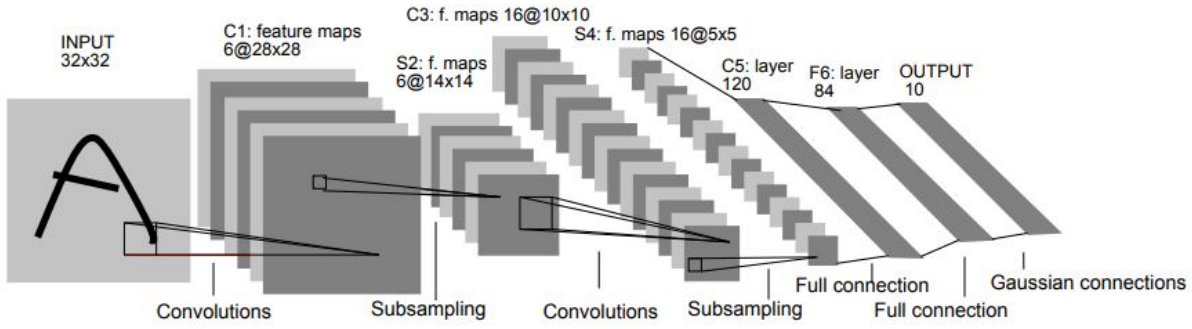
Onde ε é um parâmetro chamado taxa de aprendizagem, isto é, o quanto os pesos podem ser atualizados. Essa taxa é medida empiricamente, pois um valor baixo implica numa demora para convergir e um valor alto implica que os pesos são alterados de forma muito abrupta, podendo comprometer a otimização. Os gradientes são calculados através do algoritmo de *backpropagation*, cuja ideia básica é que os gradientes podem ser calculados eficientemente com a propagação da saída para a entrada. O algoritmo é uma aplicação prática da regra da cadeia para derivadas (LECUN *et al.*, 1998).

2.2.2.3 Deep Learning e as Redes Neurais Convolucionais

Classificadores tradicionais de aprendizagem de máquina precisam de um bom extrator de características para gerar representações dos aspectos da imagem que são importantes para discriminação, mas que são invariantes a aspectos irrelevantes (como a forma que um objeto está posicionado, por exemplo). Convencionalmente, para fortalecer esses classificadores é preciso certo trabalho manual para formular um bom descritor de características, necessitando de certa habilidade e domínio (em, por exemplo, processamento de imagens para gerar bons descritores de imagens), mas isso pode ser evitado se bons atributos forem aprendidos em um procedimento de aprendizado de propósito geral. Esse é o trunfo do *deep learning* (LECUN *et al.*, 2015).

Uma arquitetura profunda, isto é, de *deep learning*, é uma pilha multicamada composta de módulos simples sujeitos ao aprendizado, em sua maioria, e muitos calculando mapeamentos não-lineares de entrada-saída. Com múltiplas camadas de cálculo não-linear, um classificador profundo pode aprender a especificar detalhes extremamente minuciosos dos dados de entrada e a ignorar aspectos irrelevantes dos dados (LECUN *et al.*, 2015). A Figura 15 mostra uma arquitetura profunda proposta por LeCun *et al.* (1998), chamada *LeNet*.

Figura 15 – Arquitetura da rede neural convolucional LeNet-5

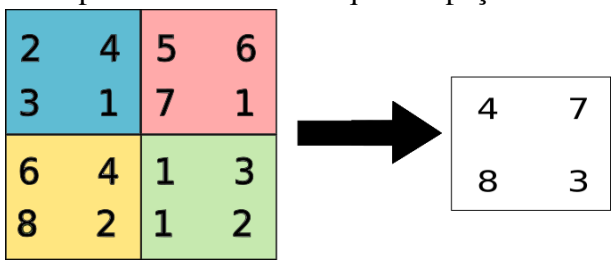


Fonte: LeCun *et al.* (1998)

A LeNet é uma Rede Neural Convolucional ou *Convolutional Neural Network* (CNN) proposta para reconhecimento de dígitos. As CNNs foram projetadas para processar dados na forma de múltiplos vetores, como imagens, por exemplo, e vem sendo apliacas com sucesso para detecção, segmentação e reconhecimento de objetos e regiões em imagens (LECUN *et al.*, 2015). A arquitetura dessas redes são compostas por vários estágios, com os primeiros responsáveis pela extração de características, compostos por dois tipos de camadas: *camadas*

convolucionais e *camadas de pooling*. As camadas convolucionais são compostas por filtros convolucionais, como apresentados na seção 2.1.3, aplicados nos dados de entrada com dimensão fixa e então os valores resultantes passam por uma função de ativação não-linear. As camadas de pooling aplicam filtros de pooling na resultante da camada anterior, que são responsáveis por sumarizar os valores correspondentes ao filtro nos dados, diminuindo a dimensionalidade e a complexidade dos dados. Como ilustrado na Figura 16, onde é mostrado um filtro de pooling máximo de tamanho 2×2. Os filtros podem saltar de pixels, e a quantidade de pixels saltados se chama *passo* ou *stride*. Além do filtro de pooling máximo, também existe o pooling da média, ou *Average Pooling*, que faz a operação de pooling se baseando na média dos pixels da máscara. Após esses estágios, os próximos são compostos *camadas densamente conectadas* até

Figura 16 – Exemplo de operação de max pooling com filtro de tamanho 2×2, onde o espaço colorido é correspondente ao filtro naquele espaço



Fonte: elaborado pelo autor.

a camada de saída, que funcionam de forma similar às camadas de uma MLP. O aprendizado de uma CNN também utiliza *backpropagation* (LECUN *et al.*, 2015). Exemplos de funções de ativação não-lineares utilizadas para as camadas da CNN incluem a função ReLU, inspirada biologicamente (HAHNLOSER *et al.*, 2000), proposta para esse fim por (GLOROT *et al.*, 2011) e descrita pela função f na equação seguinte, e a função $softmax$, que aplica uma normalização exponencial, geralmente utilizada na camada de saída da rede (NWANKPA *et al.*, 2018).

$$f(x) = \max(0, x) \tag{2.32}$$

2.3 Detecção de objetos

Detectar a presença e posição de objetos em imagens é uma das tarefas fundamentais da área da *visão computacional*. As técnicas de *detecção de objetos* envolvem, em geral, modelar um classificador a partir de imagens que contêm o objeto a ser detectado e imagens que não contêm e utilizar de *janelas deslizantes*, subimagens de tamanho fixo, para determinar se na posição da janela existe o objeto e então retornar as janelas que não se sobrepõem, onde a

sobreposição de janelas r_1 e r_2 é considerada se:

$$\frac{\text{Área}(r_1 \cap r_2)}{\text{Área}(r_1 \cup r_2)} > 0.5 \tag{2.33}$$

O problema da detecção de objetos pode ser definido na equação a seguir, considerando x uma imagem, $f(x,r)$ uma função de avaliação (ou *score*) na janela r em x e Y o conjunto de marcações válidas, que são conjuntos de janelas que não se sobrepõem, e y' o conjunto de detecções.

$$y' = \underset{y \in Y}{\text{argmax}} \sum_{r \in y} f(x,r) \tag{2.34}$$

Diversos métodos de detecção de objetos foram propostos, como, por exemplo, a utilização de HOG com o método de aprendizagem de máquina Máquina de Vetores de Suporte ou*Support Vector Machine* (SVM) (DALAL; TRIGGS, 2005) ou cascatas de características de Haar (VIOLA *et al.*, 2001). Entre eles, o método proposto por King (2015) se destaca por otimizar o procedimento padrão de janela deslizante, apresentando bons resultados para detecção de pedestres se comparado com outros trabalhos no tema (DALAL; TRIGGS, 2005), e será aprofundado nessa seção.

2.3.1 Max-Margin Object Detection (MMOD)

De forma geral, a solução mais simples para o problema de detecção de objetos é um método guloso que ordena as janelas cuja avaliação é positiva e escolhe, na ordem, as que não se sobrepõem. Esse método nem sempre retorna as melhores janelas, e um método ideal teria uma função de avaliação que tanto minimizaria o número de falsos alarmes quanto de detecções perdidas pelo método simples, como ilustrado na Figura 17. O trabalho de King (2015) propõe um novo método, que usa todas as janelas e otimiza a performance de detecção de objetos em termos de detecções perdidas e falsos alarmes.

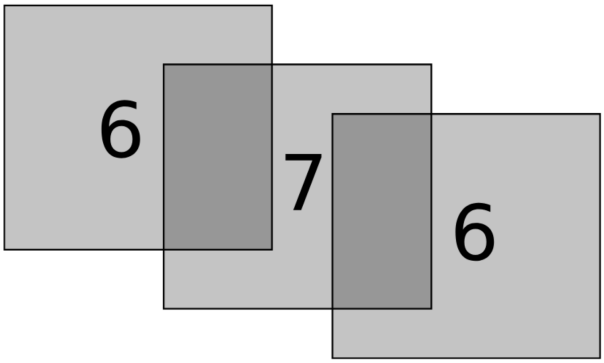
O método Max-Margin Object Detection (MMOD) considera funções de avaliação lineares em seus parâmetros, na forma:

$$f(x,r) = \langle w, \phi(x,r) \rangle \tag{2.35}$$

Onde ϕ extrai um vetor de características da janela r na imagem x e w é um vetor de parâmetros. Além disso, se a soma das avaliações de janela para um conjunto de retângulos y for considerada como $F(x,y)$, a Equação 2.34 fica:

$$y' = \underset{y \in Y}{\text{argmax}}\, F(x,y) = \underset{y \in Y}{\text{argmax}} \sum_{r \in y} f(x,r) \tag{2.36}$$

Figura 17 – Três janelas deslizantes e suas avaliações. O método guloso escolheria apenas a janela do centro, com 7 de score, enquanto um método otimizado escolheria as duas janelas exteriores, totalizando 12 de score.



Fonte: King (2015)

O algoritmo recebe um conjunto de imagens $\{x_1,...,x_n\} \subset X$ e marcações associadas $\{y_1,...,y_n\} \subset Y$ tendo como objetivo, portanto, achar um vetor w que implique nas menores quantidades de erros de detecção, ou seja, dado um certo par $(x_i, y_i) \in X \times Y$, o score da marcação correta de x_i deve ser maior que o score das marcações erradas. Portanto, a equação seguinte deve ser satisfeita sempre que possível.

$$F(x_i, y_i) > \max_{y \neq y_i} F(x_i, y) \tag{2.37}$$

Isso implica no seguinte problema convexo de otimização:

$$\begin{aligned} &\underset{w}{\text{minimizar}} \quad \frac{1}{2}||w||^2 \\ &\text{sujeito a} \qquad F(x_i, y_i) > \max_{y \in Y}[F(x_i, y) + \Delta(y, y_i)], \qquad \forall i \end{aligned} \tag{2.38}$$

onde $\Delta(y, y_i)$ denota a perda por predizer uma marcação de y quando a verdadeira marcação é y_i. Em particular, a perda é definida como:

$$\Delta(y, y_i) = L_{perd}\dot{(}\text{\# detecções perdidas}) + L_{fa}\dot{(}\text{\# falsos alarmes}) \tag{2.39}$$

Onde L_{perd} e L_{fa} controlam a importância relativa a obtenção de alta sensibilidade (baixo número de detecções perdidas) e alta precisão (baixo número de falsos alarmes), respectivamente. A Equação 2.38 é uma abordagem *hard-margin* do problema, ou seja, considera que os dados são linearmente separáveis. Como dados do mundo real são frequentemente ruidosos, não perfeitamente separáveis ou contêm *outliers*, essa abordagem é extendida para uma formulação *soft-margin* na equação seguinte, acrescentando o parâmetro C para controlar quão próximo dos dados de treinamento a otimização deve resultar e ε_i que é o limite superior na perda decorrida

do exemplo de treinamento (x_i, y_i).

$$\begin{aligned} &\underset{w,\varepsilon}{\text{minimizar}} \quad \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\varepsilon_i \\ &\text{sujeito a} \qquad F(x_i, y_i) > \max_{y \in Y}[F(x_i, y) + \Delta(y, y_i)] - \varepsilon_i, \qquad \forall i \end{aligned} \tag{2.40}$$

Essa Equação é a definidora do problema MMOD (KING, 2015) e, a partir da sua resolução com, por exemplo, o método de plano cortante (JOACHIMS *et al.*, 2009), o método guloso pode ser otimizado.

3 TRABALHOS RELACIONADOS

Apesar de ainda ser considerado um problema em aberto, vários trabalhos da literatura desenvolvem o reconhecimento automático de placas veiculares. No caso das placas brasileiras, que têm suas peculiaridades na fonte e na ordem entre dígitos e letras, diversos métodos que incluem essas singularidades foram propostos.

Tratando-se de métodos recentes que utilizam deep learning, foram estudados os trabalhos de Silva e Jung (2017) e Laroca *et al.* (2018). Além desses, Galindo *et al.* (2016) e Salazar *et al.* (2017) apresentam métodos compatíveis com o objetivo deste trabalho, sem a utilização de métodos muito custosos computacionalmente. É importante mencionar que os dois primeiros utilizam imagens coloridas em ambientes similares à visão de um motorista em um veículo em trânsito e os dois últimos utilizam imagens monocromáticas mais parecidas com imagens de radares ou sensores. Apesar dessa diferença, todos os casos tratam de ambientes não controlados. Com exceção de um trabalho (LAROCA *et al.*, 2018), carros são os únicos veículos considerados.

Para minuciar esses trabalhos, esse capítulo está divido em três seções que equivalem aos passos necessários para a tarefa de ALPR: detecção da placa, segmentação dos caracteres e leitura dos caracteres.

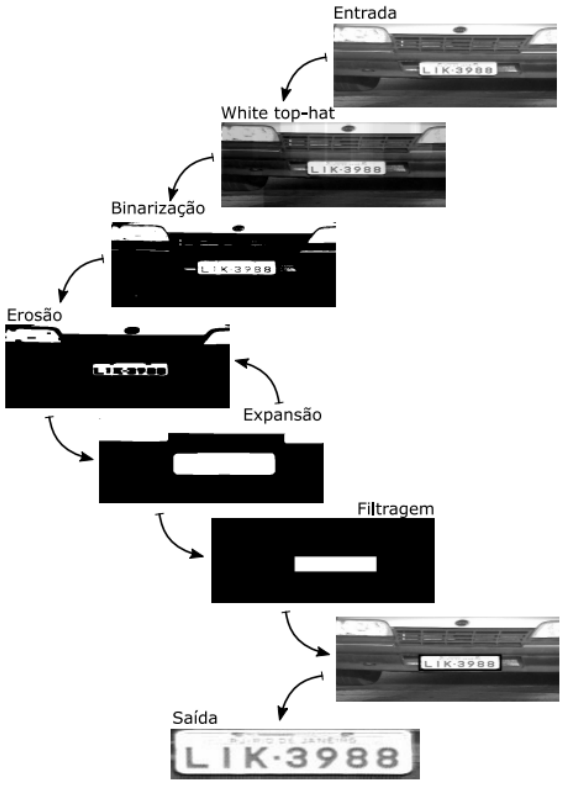
3.1 Detecção de placas

Tradicionalmente, os métodos conhecidos adotam a estratégia de detectar a placa veicular primeiro, a fim de utilizar apenas a quantidade de pixels necessários para as etapas seguintes, economizando processamento por não processar cada pixel da imagem posteriormente (DU *et al.*, 2012).

O método proposto por Galindo *et al.* (2016), esquematizado na Figura 18, consiste em aplicar a transformada *white Top-Hat* para corrigir efeitos de iluminação não uniforme na imagem, binarizar a imagem com um fator de 50%, implementar filtros de erosão e expansão para remover linhas, pontos e objetos menores que uma placa, e enfim filtrar os objetos maiores que as dimensões de uma placa para então segmentar a região encontrada.

É importante observar que o método depende de uma dimensão específica das placas e das condições de iluminação das imagens de entrada, conseguindo uma acurácia de 86% num conjunto de 100 imagens, segundo os autores.

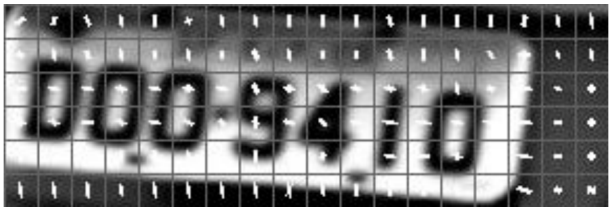
Figura 18 – Etapas para detecção da placa



Fonte: Galindo *et al.* (2016)

De forma similar, Salazar *et al.* (2017) realiza uma série de pré-processamentos para detectar as placas, mas no final utiliza um classificador. O pré-processamento é composto por: corrigir a iluminação da imagem utilizando o filtro matemático morfológico passa-baixa proposto por Wang *et al.* (2012), aplicar filtros Gaussianos, de Sobel (SOBEL, 1990) e de média, binarizar a imagem utilizando o método de Otsu (OTSU, 1979) e utilizar Análise de Componentes Conectados ou *Connected Components Analysis* (CCA) acompanhado de um fechamento de heurísticas para eliminar falsos candidatos a placas e segmentar candidatos a placa que podem ser verdadeiros. Após isso, utiliza-se um classificador SVM com HOG como descritor de características, exemplificado na Figura 19, para efetivamente apontar que placas são verdadeiras. Enfim, o ângulo de inclinação da placa é corrigido através de uma binarização com limiar adaptativo, esqueletização e transformada de Hough.

Figura 19 – HOG da placa



Fonte: Salazar *et al.* (2017)

A utilização do classificador HOG-SVM é um diferencial importante para conseguir lidar melhor com falsos candidatos a placas, porém o método ainda é dependente de um longo pré-processamento cujo resultado é influenciado pelas imagens utilizadas, semelhante ao trabalho de Galindo *et al.* (2016). Um classificador similar foi proposto no trabalho de Gonçalves *et al.* (2016a), mas utilizando um algoritmo de janela deslizante para detecção.

O método de Salazar *et al.* (2017) consegue uma precisão de $97,5\%$ e uma sensitividade de $98,4\%$, que pode ser considerado um bom desempenho. Entretanto, é preciso salientar que as imagens utilizadas nesse trabalho são de câmeras com luz infravermelha. Apesar de ser um ambiente não-controlado, podem ser resultados muito específicos, principalmente considerando o objetivo deste trabalho.

O sistema proposto por Silva e Jung (2017) trata desta etapa dividindo-a em duas fases: extração da Visão Frontal (VF) do veículo e detecção da placa contida nessa área. A VF, ilustrada na Figura 20, é definida como uma região em torno da placa, resultado de translações e escalonamentos na *bounding box* da placa (SILVA; JUNG, 2017). Tanto a VF como a placa são detectadas com *deep learning*, com modificação da arquitetura FAST-YOLO (SHAFIEE *et al.*, 2017). Esse sistema utiliza a base pública SSIG (GONÇALVES *et al.*, 2016b) contendo 2000 imagens para treinamento, validação e teste, seguindo a divisão de 40%, 20% e 40%, respectivamente.

Figura 20 – Ilustração da visão frontal (VF): o retângulo amarelo tracejado representa uma VF marcada manualmente enquanto o retângulo vermelho é a VF estimada baseada na placa (em azul)

Fonte: Silva e Jung (2017)

Para validar seu método, os autores consideram vários limites de similaridades de Jaccard, ou Intersection Over Union (IoU). Para $IoU = 0,5$, a precisão é de $95,07\%$ e a sensibilidade é de $99,51\%$, mas com $IoU = 0,7$ a precisão baixa para menos de 88% e a sensibilidade para menos de 90%. Logo quanto maior a IoU, menor o desempenho. Pode-se concluir que o método perde robustez quando é necessário que a área detectada seja mais similar

à área ideal, algo que pode ser necessário para as etapas seguintes de ALPR. Além disso, por exemplo, uma aplicação em tempo real só seria viável com uma Unidade de Processamento Gráfico ou *Graphics Processing Unit* (GPU) muito poderosa, pois essa etapa pode demorar $10,8$ ms executando em GPUs *high-end* e $94,4$ ms em GPUs *mid-end* (SILVA; JUNG, 2017), o que pode ser considerado muito custoso computacionalmente.

O sistema proposto por Laroca *et al.* (2018) é similar ao sistema de Silva e Jung (2017), baseando-se na arquitetura YOLO (REDMON *et al.*, 2016). A detecção da placa também é dividida em duas fases: extração da região do veículo e detecção da placa contida nessa região. Uma nova base pública é proposta nesse trabalho, a UFPR-ALPR, com 4000 imagens seguindo a divisão de 40%, 20% e 40% para treinamento, validação e teste, respectivamente. A Figura 21 contém amostras da base de imagens.

Figura 21 – Amostras da base UFPR-ALPR: as primeiras três linhas mostram a variedade em ambiente e veículo, bem como a variação nas placas, enquanto a última linha mostra anotações de veículo e placa; as placas estão embaçadas por questões de privacidade.

Fonte: Laroca *et al.* (2018)

Esse sistema foi avaliado tanto na base SSIG quanto na base UFPR-ALPR. Com uma confiança de $0,125$, o modelo proposto conseguiu em ambas uma precisão de 99% e sensibilidade de 100% para detecção de veículos e 98.33% de precisão e sensibilidade na detecção de placas da base UFPR-ALPR e 100% das duas medidas na detecção de placas da base SSIG. Para esses resultados foi utilizada a arquitetura FAST-YOLO (SHAFIEE *et al.*, 2017), com exceção da detecção de veículos na base UFPR-ALPR, onde foi usada a arquitetura YOLOv2 (REDMON;

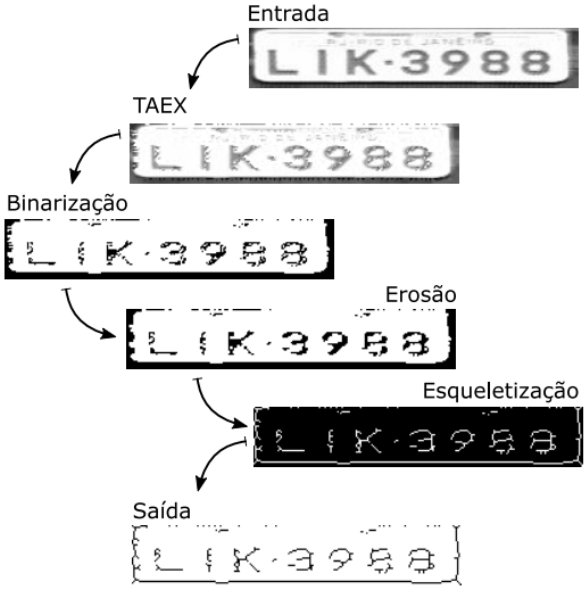
FARHADI, 2017). Também é importante mencionar que esse processamento demora 15,087 ms na base UFPR-ALPR e 8,14 ms na base SSIG, utilizando uma NVIDIA® TITAN XP, que é uma GPU *high-end* (LAROCA *et al.*, 2018). Como os valores são maiores que os vistos no trabalho de Silva e Jung (2017), a argumentação sobre o custo computacional é a mesma.

3.2 Segmentação dos caracteres

A segmentação dos caracteres é a etapa intermediária entre a detecção e a leitura da placa, utilizando a placa extraída na etapa anterior, e é necessária para que cada caractere possa ser reconhecido individualmente. Os trabalhos de Silva e Jung (2017) e Laroca *et al.* (2018) utilizam uma técnica só para a segmentação e reconhecimento, a ser abordado na próxima seção, então nesta seção serão abordados apenas os trabalhos de Galindo *et al.* (2016) e Salazar *et al.* (2017).

O sistema proposto por Galindo *et al.* (2016) divide essa etapa em duas fases. A primeira, ilustrada na Figura 22, realiza uma transformação de aguçamento extremo, aplica uma binarização com um fator de 65%, transformando a imagem em preto e branco, uma erosão e, enfim, uma esqueletização. Após isso, a segunda fase, ilustrada na Figura 23, segmenta apenas os dígitos usando projeção vertical e horizontal.

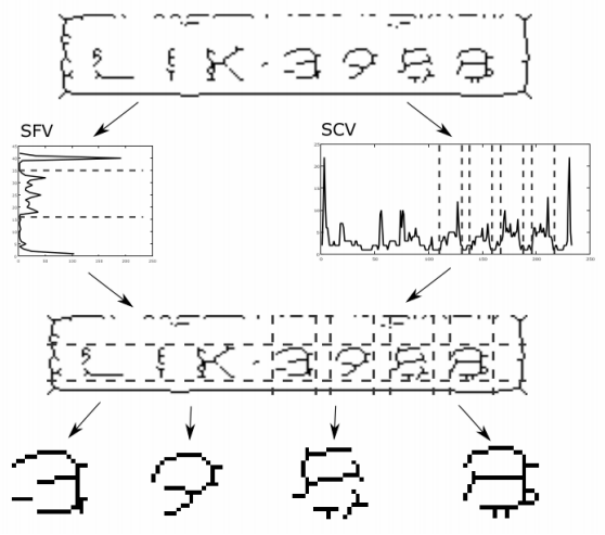
Figura 22 – Primeira fase da segmentação de caracteres



Fonte: Galindo *et al.* (2016)

Seguindo esse processamento, todas as imagens de placa tem os dígitos segmentados, mas os autores não especificam qual o desempenho dessa segmentação. Em uma base de imagens

Figura 23 – Segunda fase da segmentação de caracteres



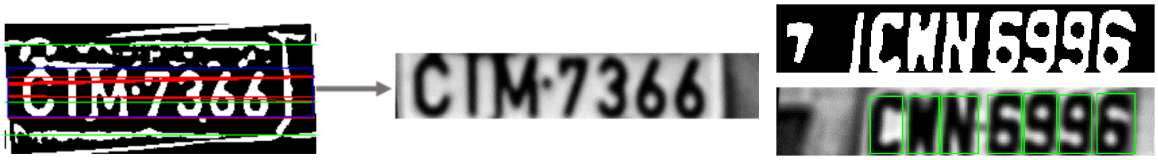
Fonte: Galindo *et al.* (2016)

diferente, maior e com várias inclinações de placa, esse resultado pode ter um desempenho ruim se não for incluída uma fase de correção do ângulo de inclinação, pois a inclinação prejudica a projeção vertical e horizontal.

No trabalho de Salazar *et al.* (2017) propõe-se primeiro enquadrar apenas uma região de interesse contendo o conjunto dos caracteres para depois segmentá-los a partir de uma classificação de qualidade das placas. Para o enquadramento, realiza-se uma binarização prévia com um limite adaptativo de Gauss, utiliza-se do método de CCA para eliminar componentes seguindo alguns critérios relativos ao tamanho da imagem da placa e, por fim, utiliza-se a transformada probabilística e progressiva de Hough, proposta por Matasyx e Kittlery (1998), para encontrar linhas horizontais e então cortar a região de interesse onde estão os caracteres. Após esse pré-processamento, realiza-se uma binarização com limiar adaptativo Gaussiano baseado na classificação das placas, construída a partir dos níveis de cinza (SALAZAR *et al.*, 2017). Para eliminar componentes ou partes de componentes indesejáveis, ruídos e suavizar contornos, aplica-se uma abertura e alguns critérios de tamanho e projeção horizontal são utilizados. Além disso, CCA é utilizada para encontrar e filtrar as *bounding boxes* dos componentes e caracteres unidos em um só componente. Esse processo de segmentação está exemplificado na 24.

Esse método conseguiu $97,03\%$ de acurácia considerando a segmentação completa das placas, ou seja, das 810 imagens do conjunto de teste, em 786 foram corretamente segmentados os 7 caracteres. Considerando cada caractere individualmente, o método conseguiu segmentar $96,7\%$ dos caracteres no conjunto de treino. Essa informação é importante porque esses caracteres segmentados serão utilizados na etapa seguinte: reconhecimento. Como a base

Figura 24 – Exemplos do processo de segmentação de caracteres



(a) Enquadramento da região de interesse (b) Resultado final da segmentação

Fonte: Salazar *et al.* (2017)

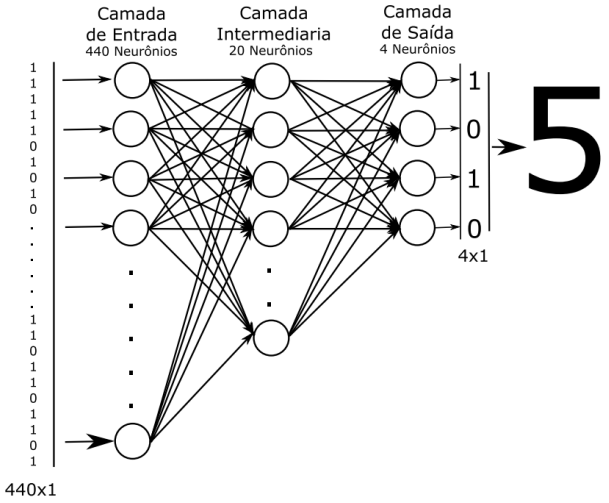
de imagens é mais variada e o método mais completo, pode-se considerar esses resultados como mais fortes que os resultados do trabalho de Galindo *et al.* (2016).

3.3 Leitura dos caracteres

A leitura dos caracteres, isto é, a classificação e contextualização dos caracteres obtidos pelas etapas anteriores, é a última etapa de um sistema ALPR. Nesta seção serão analisados os métodos de Galindo *et al.* (2016) e Salazar *et al.* (2017), que utilizam do resultado da segmentação para classificação, e também de Silva e Jung (2017) e Laroca *et al.* (2018), que em métodos parecidos utilizam a mesma técnica para detecção e classificação dos caracteres.

O método para reconhecimento dos caracteres proposto por Galindo *et al.* (2016) é composto por uma rede neural artificial do tipo Perceptrons de Múltiplas Camadas (MLP), utilizando de entrada uma transformação da matriz de saída da etapa de segmentação em vetor. A camada de entrada da rede tem, portanto, 440 neurônios, a camada intermediária tem 20 neurônios com ativação sigmoidal e a camada de saída tem 4 neurônios com função de ativação linear. A topologia da rede é mostrada na Figura 25.

Figura 25 – Topologia da rede MLP utilizada para classificar caracteres



Fonte: Galindo *et al.* (2016)

Esse método considera apenas reconhecimento de dígitos, devido ao tamanho pequeno da base de imagens utilizada, e conseguiu uma acurácia de 96% para o reconhecimento individual dos caracteres e 76% considerando a leitura completa das 43 placas detectadas na etapa de detecção. Os autores não entram em detalhes sobre o conjunto de treinamento e teste da rede MLP, o que compromete a importância desses resultados. Além disso, há uma inconsistência nas informações dos autores. Na acurácia da leitura completa é dito que 37 das 43 placas foram reconhecidas, o que resultaria em aproximadamente 86% de acurácia, e não 76%. Outra inconsistência é que os autores mencionam que 54 placas podem ser utilizadas para o reconhecimento, mas posteriormente apenas 43 são mencionadas.

A etapa de classificação dos caracteres proposta por Salazar *et al.* (2017) utiliza dois classificadores SVM, um para letras com 26 classes e outro para dígitos com 10 classes. A entrada dos classificadores é o HOG das imagens obtidas na etapa de segmentação e o tipo de treinamento utilizado foi o *one-vs-one*. Alguns exemplos de reconhecimento da placa estão contidos na Figura 26.

Figura 26 – Exemplos de reconhecimento dos caracteres da placa

Fonte: Salazar *et al.* (2017)

Considerando a classificação individual dos caracteres segmentados na etapa anterior, a acurácia desse método foi de 97%. O autor argumenta que as letras D, O e Q se confundem e, dessa forma, a classificação de letras tem a acurácia de $95,1\%$, mas, segundo pesquisa de Castelan (2009) essas letras não são levadas em consideração no cálculo da acurácia por sistemas comerciais e empresariais. Por causa disso, o autor não considera essas letras na acurácia do reconhecimento de todos os caracteres das placas, cujo resultado é de 96%. Dessa forma, a acurácia total do sistema de Salazar *et al.* (2017), definida como o produto das acurácias de cada etapa, é de $92,2\%$. É importante lembrar que a base de imagens utilizada pelo autor é uma base comercial com imagens de câmeras de luz infravermelha em posições de radar, então esses resultados podem ser considerados, ainda, muito específicos e, além disso, a omissão das letras D, O e Q compromete a acurácia obtida levando em consideração os sistemas de ALPR dos outros trabalhos estudados neste capítulo.

Os sistemas de ALPR propostos por Silva e Jung (2017) e Laroca *et al.* (2018)

utilizam a rede neural convolucional *CR-NET*, proposta por Silva e Jung (2017), com algumas peculiaridades em cada sistema. Essa rede é baseada na arquitetura YOLO, aproveitando as primeiras 11 camadas dessa arquitetura e acrescentando mais 4 camadas para melhorar a não-linearidade do modelo.

O trabalho de Silva e Jung (2017) considera a letra O e o dígito 0 como se fossem o mesmo objeto, totalizando 35 classes entre letras e dígitos para detecção, e propõe também algumas heurísticas para reconhecimento após o processamento da CR-NET:

- Se mais de sete caracteres são detectados, apenas os sete mais prováveis são mantidos como saída da rede;
- Se a saída da rede não for exatamente sete caracteres, o reconhecimento é dado como errado e encerra-se.
- Caso contrário, os três primeiros caracteres são presumidos como letras e os quatro últimos como dígitos, dessa forma são realizadas as seguintes trocas:
  - Para as letras são: $5 \Rightarrow S$, $7 \Rightarrow Z$, $1 \Rightarrow I$, $8 \Rightarrow B$, $2 \Rightarrow Z$, $4 \Rightarrow A$ e $6 \Rightarrow G$.
  - Para os dígitos são: $Q \Rightarrow 0$, $D \Rightarrow 0$, $Z \Rightarrow 7$, $S \Rightarrow 5$, $J \Rightarrow 1$, $I \Rightarrow 1$, $A \Rightarrow 4$ e $B \Rightarrow 8$.

A Figura 27 contém exemplos de leitura das placas utilizando o processamento descrito.

Figura 27 – Exemplos de leitura das placas

HJN2081 PUG7401 00J2834 HIF8352 00K5510 GMF2862 HAY5852

Fonte: Silva e Jung (2017)

Esse método resultou precisão e sensibilidade maiores que 99% para a segmentação, considerando $IoU = 0.5$ na base SSIG. Para o reconhecimento, considerando o sistema ALPR completo, o método reconheceu $63,18\%$ das placas do conjunto de teste corretamente. Considerando pelo menos 6 caracteres corretos, esse resultado sobe para $90,55\%$ e, considerando pelo menos 5 caracteres corretos, sobe para 97.33% (SILVA; JUNG, 2017). Em comparação com o sistema comercial Sighthound, detalhado em Masood *et al.* (2017), esse método obteve 13.9% de vantagem. Os autores ressaltam que enquanto o método reconheceu corretamente todos os quatro dígitos em $93,04\%$ dos casos, para as três letras esse resultado baixa para 63.43%. Isso mostra, portanto, que a maior dificuldade do sistema de Silva e Jung (2017) é o reconhecimento de letras. Além disso, o tempo de execução dessa etapa foi de $2,2$ ms em uma GPU *high-end* e $20,1$ ms em uma GPU *mid-end*. O tempo de execução do sistema ALPR completo em Silva e Jung (2017) foi de 13 ms em uma GPU *high-end* e $114,5$ ms em uma GPU *mid-end*, ou seja, a eficiência

do sistema depende do equipamento e pode não funcionar bem para algumas aplicações em máquinas menos robustas.

No lugar das heurísticas propostas por Silva e Jung (2017), o trabalho de Laroca *et al.* (2018) propõe que três modelos diferentes sejam criados: um para a segmentação, um para letras, com 26 classes, e um para dígitos, com 10 classes. As redes convolucionais dos modelos têm a mesma arquitetura, baseada na CR-NET de Silva e Jung (2017), com o modelo de dígitos removendo as quatro primeiras camadas. A base utilizada, UFPR-ALPR, contém imagens de placa de moto, que são redimensionadas para uma proporção similar às placas de carro, de forma que seja utilizado o mesmo modelo para os dois tipos de placa. Para melhorar o treinamento dos modelos, usa-se uma adição artificial de dados seguindo as regras:

- Inclui-se imagens negativadas das placas e seus caracteres, para aumentar o conjunto de treinamento e simular placas de outras categorias que podem variar de cor, como, por exemplo, placas de ônibus, que são vermelhas;
- Se mais de 7 caracteres são detectados, são mantidos apenas os 7 mais prováveis que não se sobrepõem ($IoU \geq 0,25$ para carros e $IoU \geq 0,25$ para motos);
- Para melhorar os resultados da segmentação e do reconhecimento, uma margem é adicionada tanto na *bouding box* da placa quanto nas *bounding boxes* dos caracteres;
- Os dígitos 0 e 1 também são usados como instâncias das letras I e O.
- Os caracteres podem ser refletidos para gerar novas instâncias da seguinte forma:
  - Verticalmente: 0, 1, 3, 8, B, D, E, H, I, K, O e X;
  - Horizontalmente: 0, 1, 8, A, H, I, M, O, T, U, V, W, X e Y;
  - Ambas os eixos: 0, 1, 6 (para gerar 9), 8, 9 (para gerar 6), H, I, N, O, S, X e Z.

Além disso, esse sistema implementa uma redundância temporal, que é utilizada para melhorar o reconhecimento considerando o caractere mais frequente da redundância, baseado nos trabalhos de Gonçalves *et al.* (2016a) e Donoser *et al.* (2007). Com essa implementação, a base UFPR-ALPR tem 60 placas a serem reconhecidas. Alguns exemplos de leitura das placas estão contidos na Figura 28.

Considerando o conjunto de teste da base UFPR-ALPR proposta pelos autores, esse método conseguiu uma sensibilidade de $95,97\%$ com tempo de execução de $1,6548$ ms na segmentação dos caracteres e acurácia de $90,37\%$ com tempo de execução de $11,6591$ ms no reconhecimento dos sete caracteres. O sistema ALPR teve uma acurácia de $64,89\%$ e, com redundância temporal, alcançou $78,33\%$. Para efeitos de comparação, os sistemas comerciais

Figura 28 – Exemplos de leitura das placas

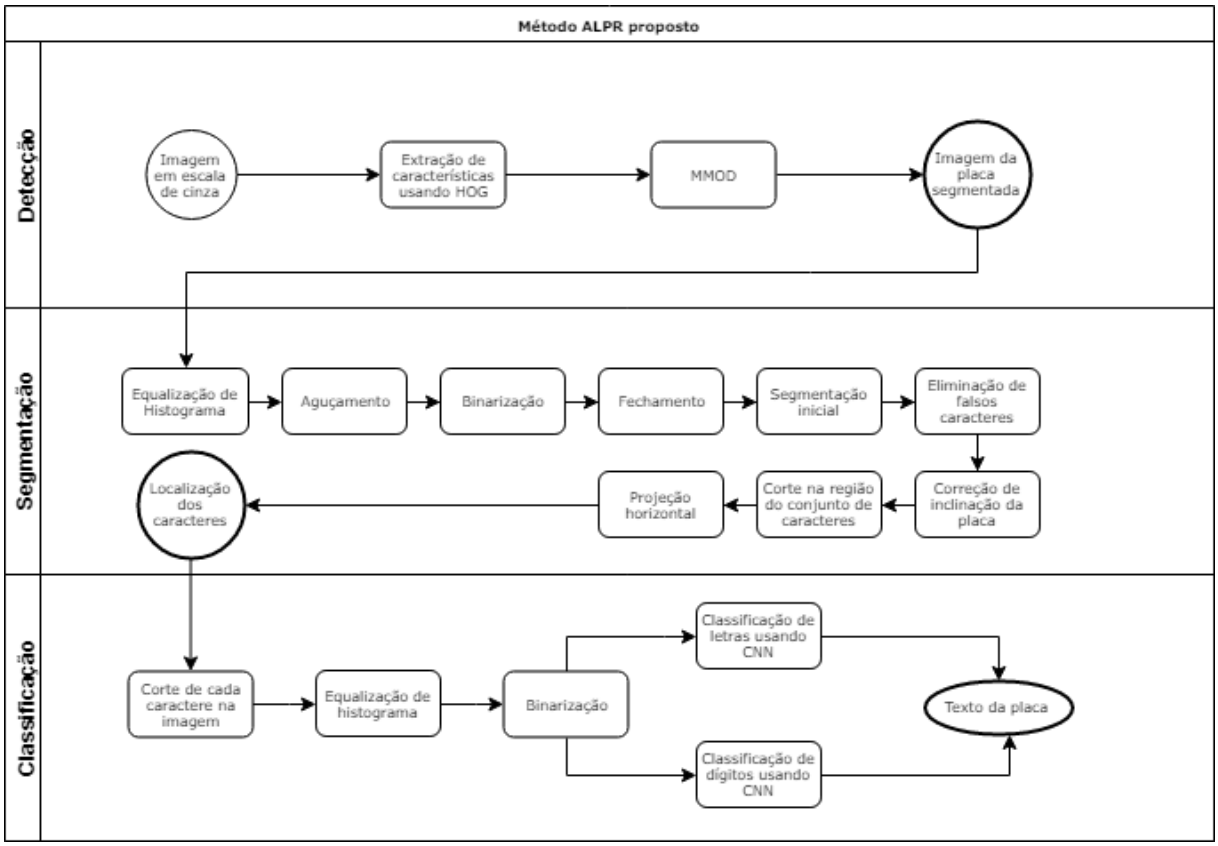
Fonte: Laroca *et al.* (2018)

Sighthound e OpenALPR (MATTHEW HILL, 2014) obtiveram 47,39% e 50,94% de acurácia, respectivamente. Com a redundância temporal, esses resultados sobem para 56,67% para o Sighthound e 70,00% para o OpenALPR. Desses resultados, pode-se concluir que a base UFPR-ALPR é a mais variada e desafiadora para os métodos de ALPR (LAROCA *et al.*, 2018). Considerando a base SSIG, o método proposto teve acurácia de 85,45% (93.53%, considerando a redundância temporal), uma melhora significativa em relação ao resultado original de Silva e Jung (2017).

4 PROPOSTA DE UM MÉTODO ALPR

Neste capítulo, a proposta de um método ALPR será detalhada minuciosamente, seguindo as etapas de detecção da placa veicular, segmentação dos caracteres da placa e reconhecimento. O método pode ser resumido no diagrama da Figura 29.

Figura 29 – Diagrama resumindo o método ALPR proposto



Fonte: Elaborado pelo autor.

4.1 Analisando a base de dados

As imagens utilizadas neste trabalho pertencem à base de imagens UFPR-ALPR (LAROCA *et al.*, 2018), que contém 4500 imagens com resolução 1920×1080 pixels. As câmeras utilizadas foram GoPro Hero4 Silver, Huawei P9 Lite e iPhone 7 Plus, cada uma produzindo 1500 imagens, seguindo a composição:

- 900 imagens de carros com placas cinza;
- 300 imagens de carros com placas vermelhas;
- 300 imagens de motocicletas com placas cinza.

A base é dividida como: 40% para treinamento, 40% para teste e 20% para validação. Todas as

imagens possuem as seguintes anotações disponíveis em um arquivo de texto: a câmera em que a foto foi tirada; a posição do veículo e informação como tipo (carro ou motocicleta), fabricante, modelo e ano; a identificação e posição da placa, bem como as posições dos caracteres. A Figura 30 mostra um exemplo de imagem da base.

Figura 30 – Amostra da base UFPR-ALPR (com placa borrada por questões de privacidade)

Fonte: Elaborado pelo autor, retirado da base UFPR-ALPR proposta por Laroca *et al.* (2018)

A base de dados pode ser considerada desafiadora (LAROCA *et al.*, 2018), pelo ambiente não controlado das imagens, variando inclinação, presença ou não de sombras, ruídos ou borrões, bem como pela variedade de tipos de placa encontradas. A Figura 31 mostra exemplos dessa variação.

Para o método proposto, é considerada apenas imagens de carro com placas cinza para o problema discutido, em especial na fase de segmentação, mas todas as imagens são usadas na fase de detecção e na classificação caracteres. Originalmente, as imagens da base UFPR-ALPR são coloridas, mas serão consideradas que as imagens são convertidas para escala de cinza.

4.2 Detecção de placas veiculares

A primeira etapa do método é a detecção de placas, realizada utilizando HOG, como apresentado na Seção 2.1.5, como descritor de características e o método de detecção MMOD

Figura 31 – Amostras da base UFPR-ALPR exemplificando a variação de veículos de ambientes

(a) (b) (c)

(d) (e) (f) (g)

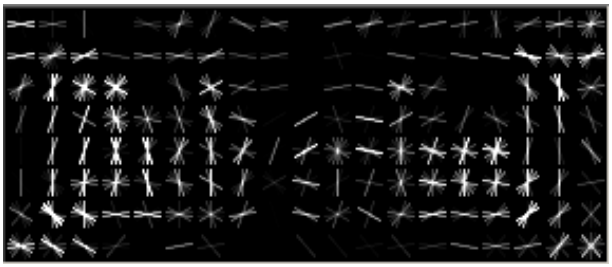
Fonte: Elaborado pelo autor, retirado da base UFPR-ALPR proposta por Laroca *et al.* (2018)

para encontrar as *bouding boxes* das placas, como descrito em 2.3.1.

4.2.1 Max-Margin Object Detection (MMOD) com HOG

A implementação do método MMOD utiliza a localização da placa nas imagens de treinamento como exemplos positivos e todas as janelas como exemplos negativos, de forma a encontrar os melhores parâmetros *w* que impliquem no menor número de erros de detecção para o conjunto de treinamento, utilizando como descritor de características o HOG dessas imagens. A Figura 32 mostra o HOG aprendido por esse método a partir das imagens do conjunto de treinamento e a Figura 33 mostra um exemplo de placa detectada após o aprendizado do método.

Figura 32 – HOG de placa



Fonte: Elaborado pelo autor.

Figura 33 – Exemplo de placa detectada pelo método MMOD (com a placa borrada por questões de privacidade dos dados)

Fonte: Elaborado pelo autor.

4.3 Segmentação de caracteres

A etapa de segmentação dos caracteres da placa é realizada através de uma série de processamentos na imagem da placa obtida pela fase de detecção, conforme apresentados na Seção 2.1. Para todos os processamentos seguintes, a imagem da placa está redimensionada para 300 pixels de largura e 200 de altura, sendo o suficiente para as os processamentos envolvendo vizinhança de pixel e formas sem aumentar muito a quantidade de pixels a serem processados.

4.3.1 Equalização de Histograma

Dado o objetivo de segmentar os caracteres da imagem de placa, o primeiro passo é aumentar o contraste da imagem, de forma a deixar os caracteres destacados na imagem. Para isso, utiliza-se a equalização de histograma. Dessa forma, o fundo cinza e os caracteres pretos tem uma diferença maior de intensidade. A Figura 34 exemplifica o processo de equalização de histograma numa placa.

4.3.2 Aguçamento

As imagens de placa tem baixa resolução, resultando em detalhes pouco perceptíveis e borrados. Então, para realçar esses detalhes, como os próprios caracteres, aplica-se a operação de aguçamento, através de filtragem espacial com a máscara H, resultante da soma da imagem

Figura 34 – Aplicação de equalização de histograma em uma imagem de placa

(a) Imagem original (b) Imagem com histograma equalizado

Fonte: elaborado pelo autor.

original com as arestas do operador de Laplace, multiplicada por um fator α, resultando na máscara H'.

$$H' = \alpha \times H = \alpha \times \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{4.1}$$

O resultado dessa filtragem com $\alpha = 3$ está exemplificado na figura 35.

Figura 35 – Aguçamento de uma imagem de placa

Fonte: Elaborado pelo autor

4.3.3 Binarização

Uma vez que a imagem está com contraste maior e com detalhes aguçados, utiliza-se o método de Otsu (OTSU, 1979) para binarizar a imagem, de forma que fiquem preto os pixels mais escuros da imagem original e brancos os mais claros. Da imagem resultante é então calculado seu complemento, ou seja, inverte-se os pixels pretos para branco e vice-versa. Desse processamento resulta uma imagem binária onde os caracteres estão em branco num fundo preto, mas além dos caracteres, outros objetos e ruídos mais escuros também podem ter sido destacados pelo processo, como pode ser observado na figura 36.

4.3.4 Fechamento

Como o processo de binarização não é perfeito, podem existir pequenos buracos nos objetos, em especial nos caracteres, o que não é desejável. Para eliminar esses buracos, aplica-se

Figura 36 – Imagem de placa binarizada através do método de Otsu

Fonte: Elaborado pelo autor

um fechamento, conforme mostrado na Figura 37.

Figura 37 – Imagem resultante de um fechamento em uma imagem binarizada de placa

Fonte: Elaborado pelo autor

4.3.5 Eliminação de falsos caracteres e correção da inclinação da placa

A partir do resultado do fechamento, realiza-se uma segmentação inicial, utilizando de qualquer heurística simples que retorne as *bounding boxes* dos objetos na imagem, como encontrar os contornos dos objetos (SUZUKI *et al.*, 1985) e retornar os menores retângulos que os contêm. Considerando H a altura da imagem da placa e h a altura do objeto, ele só permanece na lista de *boxes* se obedecer os critérios, obtidos com base na padronização da placa brasileira e no formato dos retângulos obtidos pela etapa de detecção:

$$H/h \leq 3 \qquad \text{e} \qquad H/h \geq 1.2 \tag{4.2}$$

Após essa filtragem das *boxes*, cujo resultado está ilustrado na Figura 38, é utilizada uma regressão linear, conforme apresentado na Seção 2.2.1, nas coordenadas (x,y) do ponto superior esquerdo das *boxes*. O resultado será uma função de primeiro grau da forma $f(x) = ax+b$, onde $f(x)$ é uma aproximação da coordenada y, ou seja, uma reta na imagem montada a partir das posições dos objetos. O erro dessa aproximação pode ser dado por $|y - f(x)|$. Enquanto houverem mais do que sete objetos, pode-se excluir aquele que tem o maior erro na regressão linear, ou seja, que mais se distancia da reta obtida, e então a regressão linear é calculada novamente. A Figura 39 mostra a reta obtida ao final desse processamento em um exemplo. A correção da inclinação é feita utilizando a função obtida pela regressão linear. A derivada $f'(x)$, ou seja, o termo a, é igual à tangente do ângulo de inclinação da reta. Desse

Figura 38 – Segmentação inicial em uma imagem de placa

Fonte: Elaborado pelo autor

Figura 39 – Resultado de uma regressão linear aplicada nas posições dos caracteres

Fonte: Elaborado pelo autor

modo, pode-se considerar esse ângulo, que é igual a $\arctan(a)$, como a inclinação da placa e corrigi-la rotacionando a imagem e as *boxes* no sentido oposto. A Figura 40 ilustra o resultado dessa correção.

Figura 40 – Imagem binária de placa com inclinação corrigida

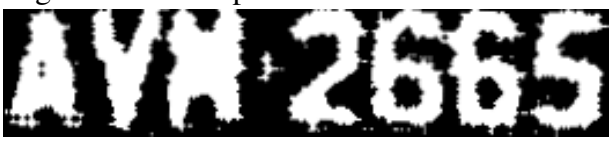
Fonte: Elaborado pelo autor

4.3.6 Segmentação final

Uma vez corrigida a inclinação da placa, é realizado um corte para segmentar apenas a região que contém os caracteres. No eixo horizontal, é utilizado como ponto inicial a coordenada do ponto esquerdo da *box* do primeiro caractere detectado e como ponto final a coordenada do ponto direito do último caractere detectado. No eixo vertical, é utilizado como ponto inicial a coordenada do ponto esquerdo da *box* do primeiro caractere detectado e como ponto final a coordenada da *box* de caractere de maior altura. Esse corte está ilustrado na Figura 41.

Depois desse corte na imagem, é calculado o histograma de branco dos pixels horizontais e sua média para então subtrair um quinto da média do histograma, formando a projeção horizontal. As regiões no eixo horizontal onde a projeção não é zero são as potenciais

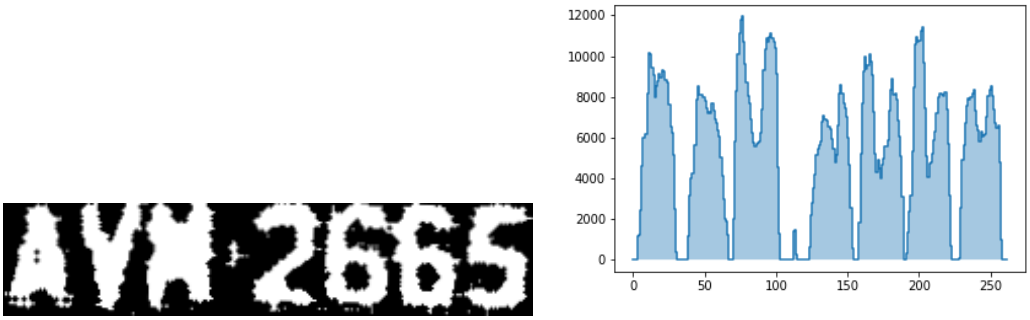
Figura 41 – Corte na imagem binária de placa



Fonte: Elaborado pelo autor

regiões de caracteres finais, como ilustra a Figura 42. Caso existam mais de sete regiões, elimina-se a de menor largura até que existam apenas sete. Esse método garante que uma vez que a inclinação é corrigida e pelo menos o primeiro e último caracteres são previamente encontrados, os sete caracteres podem ser segmentados. Por consequência, é possível extrair a região dos caracteres na imagem da placa original com inclinação corrigida, como mostrado na Figura 43.

Figura 42 – Imagem dos caracteres da placa e sua respectiva projeção horizontal



(a) (b)

Fonte: elaborado pelo autor.

Figura 43 – Imagem de placa com seus caracteres segmentados

Fonte: elaborado pelo autor

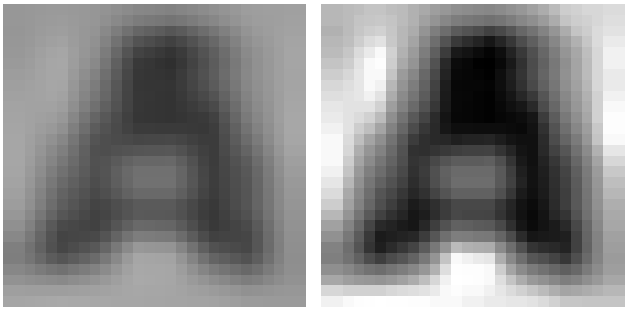
4.4 Classificação de caracteres e reconhecimento da placa

Depois que a placa foi localizada e a posição dos caracteres encontradas, obtêm-se imagens separadas dos caracteres. Essas imagens serão classificadas para completar o processo de leitura da placa veicular. Para o treinamento, foi utilizada uma marcação manual dos caracteres, na base UFPR-ALPR. O processamento para a classificação de cada caractere é descrito a seguir.

4.4.1 Equalização de histograma

O primeiro passo é aumentar o contraste da imagem para tornar os detalhes mais visíveis, através de uma equalização do histograma da imagem, semelhante à equalização do processo de segmentação, mas dessa vez levando em consideração apenas a subimagem correspondente da janela do caractere, como ilustra a Figura 44.

Figura 44 – Aplicação de equalização de histograma em uma imagem de caractere



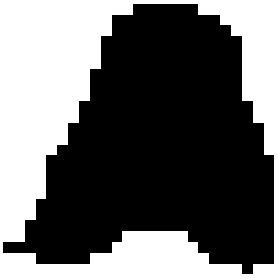
(a) Imagem original (b) Imagem com histograma equalizado

Fonte: elaborado pelo autor

4.4.2 Binarização

Levando em consideração a imagem resultante com contraste acentuado, aplica-se uma binarização utilizando o método de Otsu (OTSU, 1979) e então redimensiona-se a imagem para dimensão 28×28. A imagem binária deverá conter o caractere destacado em uma intensidade e o fundo em outra, como ilustra a Figura 45. A imagem binária é então utilizada tanto para o treinamento do classificador quanto para a inferência, isto é, predizer qual é o caractere contido na imagem.

Figura 45 – Imagem binária de um caractere



Fonte: elaborado pelo autor.

4.4.3 Classificação de dígitos e letras

A partir do pré-processamento composto pela equalização de histograma e binarização, utiliza-se dois classificadores separados: um para dígitos e outro para letras. Isso é possível graças ao formato da placa veicular brasileira, onde os três primeiros caracteres são letras e os quatro restantes são dígitos.

Ambos os classificadores são modelos de Redes Neurais Convolucionais, com arquitetura baseada na LeNet (LECUN *et al.*, 1998), como descrita na Seção 2.2.2, pois a LeNet tem uma arquitetura simples e é relativamente eficaz para o problema de leitura de dígitos. A Tabela 1 detalha o classificador de dígitos e a Tabela 2 o classificador de letras, cuja diferença na arquitetura é na última camada, tendo 26 saídas em vez de 10. Além disso, é adicionado

Tabela 1 – Arquitetura da rede para reconhecimento de dígitos

Camada	Tipo	Dimensões	Filtros	Passo	Tamanho da Máscara	Ativação
I	Imagem de entrada	30×30				
C1	Convolução	28×28	6	1	3×3	relu
P1	Average Pooling	14×14		2	2×2	
C2	Convolução	12×12	16	1	3×3	relu
P2	Average Pooling	6×6		2	2×2	
FC	Densamente conectada	120				ReLU
FC	Densamente conectada	84				ReLU
FC	Densamente conectada	10				softmax

Fonte: Elaborado pelo autor

Tabela 2 – Arquitetura da rede para reconhecimento de letras

Camada	Tipo	Dimensões	Filtros	Passo	Tamanho da Máscara	Ativação
I	Imagem de entrada	30×30				
C1	Convolução	28×28	6	1	3×3	relu
P1	Average Pooling	14×14		2	2×2	
C2	Convolução	12×12	16	1	3×3	relu
P2	Average Pooling	6×6		2	2×2	
FC	Densamente conectada	120				ReLU
FC	Densamente conectada	84				ReLU
FC	Densamente conectada	26				softmax

Fonte: Elaborado pelo autor

um *padding* de tamanho 2 na imagem de entrada, isto é, são preenchidas 2 faixas adicionais de zeros em cada borda da imagem. Desta forma, a imagem de entrada com dimensões 28×28 passa a ter dimensões 30×30. Outro detalhe de implementação relevante é que depois de cada camada convolucional é realizada a operação de normalização de *batch*. Um mini-batch é o subconjunto utilizado numa iteração do treinamento e a sua normalização é transformar cada

dado x em $BN(x)$ seguindo a seguinte fórmula:

$$BN(x) = \gamma \frac{x - \mu_B}{\sqrt{\sigma_B^2 - \varepsilon}} + \beta \tag{4.3}$$

onde μ_B e σ_B^2 são a média e a variância do mini-batch onde x está inserido, respectivamente, ε é uma constante e γ e β são parâmetros a serem aprendidos pela rede (IOFFE; SZEGEDY, 2015).

Uma vez reconhecidos as três letras e os quatro dígitos, na ordem de posição no eixo horizontal, tem-se a leitura completa da placa veicular.

5 RESULTADOS

Este capítulo discute os resultados do método proposto utilizando a base de imagens UFPR-ALPR (LAROCA *et al.*, 2018), conforme apresentada na Seção 4.1. Posteriormente, os resultados serão comparados com alguns métodos da literatura, como vistos no Capítulo 3.

O *hardware* utilizado para toda a implementação do método é composto por um processador AMD Ryzen 3 2200G, com frequência base de $3,5$ GHz e quatro núcleos, e 8 GB de memória DDR4, sendo apenas 6 GB utilizáveis devido à reserva de memória para a placa gráfica integrada. Em relação a *software* , o sistema operacional utilizado foi o Windows 10 e o método foi desenvolvido em Python, através do ambiente de desenvolvimento Spyder e utilizando as bibliotecas OpenCV, Dlib, Scikit-Learn e Keras.

5.1 Detecção de placas veiculares

O objetivo dessa etapa, como já mencionado anteriormente, é localizar a região da placa veicular e segmentá-la. Esse processo é realizado através do método MMOD e utilizando HOG como descritor de características.

A implementação do método MMOD utiliza uma quantidade considerável de memória dependendo do conjunto de dados de treinamento, principalmente considerando a resolução de 1090 por 1080 das imagens da base UFPR-ALPR. Portanto, para a detecção de placas será considerada subimagens de 50 pixels adicionais, horizontalmente e verticalmente, em relação à localização real da placa, formando uma base secundária para a etapa de detecção, visto que com o equipamento utilizado o treinamento seria impossível. Além disso, também serão consideradas imagens de placa vermelha, que mantêm uma série de características em comum com as placas que o trabalho propõe a detecção, apesar de não estarem no escopo do problema. A Figura 46 mostra um exemplo dessa base.

Figura 46 – Exemplo de imagem utilizada para analisar o método de detecção de placas

Fonte: Elaborado pelo autor.

Para avaliar os resultados obtidos, serão utilizados os conceitos de precisão e sensibilidade. Considerando tp como número de acertos (positivos verdadeiros), fp como número de falsos alarmes (positivos falsos) e fn como número de erros (negativos falsos), a precisão e a sensibilidade são definidas como nas seguintes equações.

$$\text{precisão} = \frac{tp}{tp+fp} \tag{5.1}$$

$$\text{sensibilidade} = \frac{tp}{tp+fn} \tag{5.2}$$

Dessa forma, a precisão mede a capacidade do modelo de acertar quando prediz um exemplo como positivo e a sensibilidade mede a capacidade do modelo de acertar quando prediz um exemplo como negativo.

Considerando apenas imagens da base secundária para os conjuntos de treinamento, validação e teste e considerando os parâmetros da função de otimização do MMOD, descrita na Equação 2.40, como $C = 10$ e $\varepsilon = 0,01$, são obtidos os resultados mostrados na Tabela 3. A janela deslizante obtida pela média das localizações no conjunto de treino é de 139 pixels de largura e 46 pixels de altura e as imagens têm a resolução aumentada pelo dobro uma vez para permitir a detecção de placas menores.

Tabela 3 – Resultados da detecção de placas veiculares considerando a base secundária de imagens

Conjunto	Precisão	Sensibilidade
Treinamento	100%	99,72%
Validação	100%	96,52%
Teste	97,71%	95,20%

Fonte: Elaborado pelo autor

Em outra análise, utilizando o mesmo modelo gerado nos resultados da Tabela 3 testado nas 1080 imagens de placas cinza do conjunto de teste da base UFPR-ALPR, tem-se precisão de $87,76\%$ e sensibilidade de $97,33\%$, resultado a ser discutido em comparação com outros métodos a seguir. Como as imagens utilizadas para treinamento contêm apenas poucos pixels ao redor da placa, os exemplos negativos são insuficientes e então muitos falsos positivos são obtidos, diminuindo a precisão, como exemplifica a Figura 47. Uma precisão maior pode ser obtida com imagens de treinamento com áreas maiores, proporcionalmente ao algoritmo de detecção mais exemplos negativos. Além disso, para melhorar o método como um todo, poderia ser utilizada uma Rede Neural Convolucional como extrator de características para o MMOD.

Figura 47 – Exemplo de imagem com falsos positivos detectados

Fonte: Elaborado pelo autor.

Em comparação com o método de Salazar *et al.* (2017), cujo escopo também é de placas cinza mas em imagens de câmeras de luz infravermelha, a precisão é inferior em $9,74\%$ e a sensibilidade em $0,07\%$. Além disso, o método de Galindo *et al.* (2016) tem 97% de precisão e 89% de sensibilidade numa base simples de apenas 100 imagens enquanto o método proposto neste trabalho possui uma sensibilidade melhor, numa base maior e mais variada. Isso significa que o método tem robustez similar ou melhor para uma base de imagens mais fidedigna a dados reais, considerando as limitações de treinamento, mostrando uma certa vantagem para o método MMOD com HOG em relação ao que foi utilizado nos trabalhos citados.

Em relação aos métodos de Silva e Jung (2017) e Laroca *et al.* (2018), que usam deep learning para detecção, o método com MMOD e HOG tem uma sensibilidade menor em $2,18\%$ na base SSIG com o método de Silva e Jung (2017) na pior das hipóteses, isto é, considerando uma baixa IoU igual a $0,5$, e em apenas 1% na base UFPR-ALPR completa com o método proposto por Laroca *et al.* (2018). A precisão do método proposto é inferior aos trabalhos citados em $7,31\%$ e $10,57\%$, respectivamente. É importante observar, como mencionado no Capítulo 3, que quanto maior a IoU menor o desempenho, podendo alcançar valores próximos e até menores que o o método proposto neste trabalho.

5.2 Segmentação dos caracteres

Esta etapa, como já mencionado, trata da segmentação da região individual dos sete caracteres numa imagem de placa. Este trabalho propõe um método que utiliza pré-processamento de imagens e regressão linear, sendo uma proposta mais simples devido aos objetivos do trabalho, que estão mais concentrados nas outras etapas. O método será comparado com os trabalhos vistos no Capítulo 3.

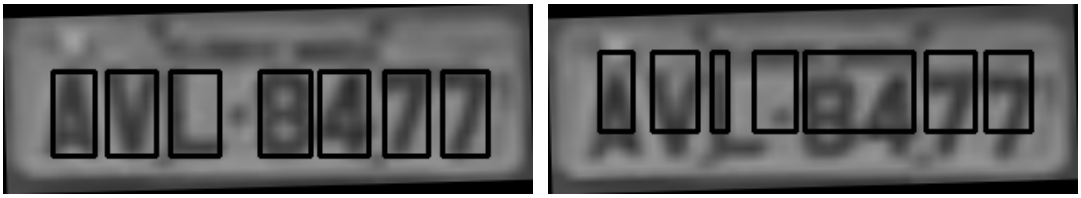
Para a análise do método foi considerada as marcações de placas dos conjuntos de

treino e teste com placas cinzas da base UFPR-ALPR, ambos com 1080 imagens. O método segmentou sete regiões de caracteres possíveis em 854 e 824 imagens do conjunto de treinamento e de teste, respectivamente. Isso resulta em uma acurácia máxima de $79,09\%$ no conjunto de treinamento e $76,29\%$ no conjunto de teste, considerando acurácia definida na equação a seguir.

$$\text{acurácia} = \frac{\text{\#placas com todos os caracteres segmentados}}{\text{\#total de placas}} \tag{5.3}$$

O trabalho de Salazar *et al.* (2017) reporta uma acurácia de $97,03\%$, um valor visivelmente maior, mas em base de imagens obtidas por câmeras de luz infravermelha em posição de radar. Os métodos propostos por Laroca *et al.* (2018), utilizando a rede CR-NET (SILVA; JUNG, 2017), obtêm uma sensibilidade de $95,97\%$ na segmentação utilizando a base UFPR-ALPR completa. Isso significa que o método proposto neste trabalho ainda deve ser melhorado para aplicações reais, servindo principalmente como ponte entre os objetivos do trabalho. A Figura 48 mostra dois exemplos de segmentação de caracteres, um correto e um errado.

Figura 48 – Exemplos de segmentação de caracteres



(a) Segmentação correta (b) Segmentação errada

Fonte: elaborado pelo autor.

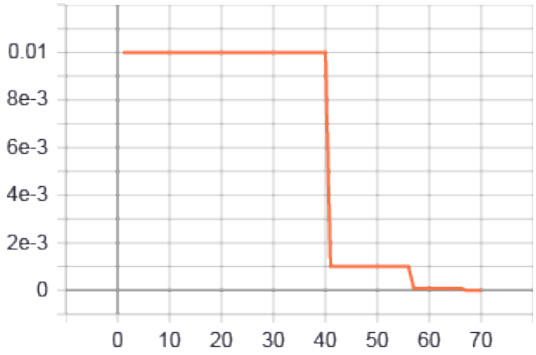
5.3 Classificação dos caracteres

A última etapa do sistema de ALPR é o reconhecimento (ou classificação) dos caracteres, como já mencionado. Neste trabalho é proposto que seja utilizado *deep learning* para essa tarefa, com auxílio de pré-processamento de imagens. O método proposto divide esta etapa em duas: classificação de dígitos e classificação de letras.

Em ambos os classificadores, considera-se a base UFPR-ALPR completa para treinamento, validação e teste. Por mais que placas de moto e placas vermelhas estejam fora do escopo, podem ser úteis para aumentar os dados utilizados. Além disso, a parametrização é similar, com uma taxa de aprendizagem que começa em $0,01$ e diminui em um fator de $0,1$ caso a perda

calculada não diminua por cinco épocas, como mostra a Figura 49 e caso não haja evolução em dez épocas, o treinamento automaticamente é encerrado.

Figura 49 – Ajuste automático de taxa de aprendizagem: taxa de aprendizagem por épocas



Fonte: Elaborado pelo autor.

Para entender a evolução da aprendizagem, a Figura 50 mostra as curvas de *loss* e acurácia para cada classificador. Como é possível observar, os dois classificadores tem uma coerência no aprendizado, evoluindo no conjunto de treino e também no de validação. As Tabelas 4 e 5 mostram o resultado final nos conjuntos de treinamento, validação e teste.

Tabela 4 – Avaliação do classificador de dígitos

Conjunto	Acurácia	Loss
Treinamento	100%	$6,85 \times 10^{-3}$
Validação	96,28%	0,1118
Teste	95,45%	0,1929

Fonte: Elaborado pelo autor

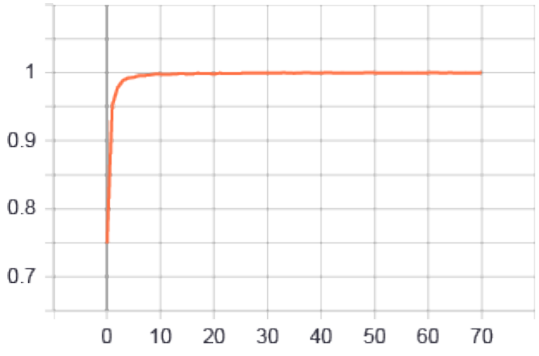
Tabela 5 – Avaliação do classificador de letras

Conjunto	Acurácia	Loss
Treinamento	99,95%	0,0383
Validação	85,09%	0,7506
Teste	85,04%	0,7047

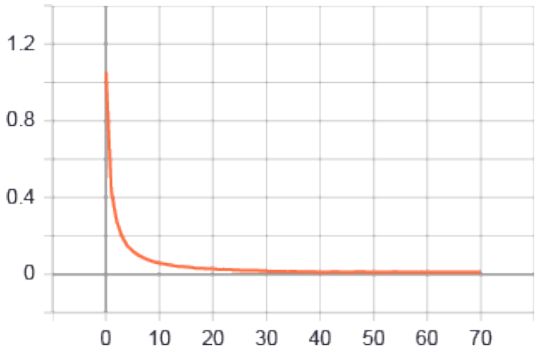
Fonte: Elaborado pelo autor

Esses resultados mostram que o classificador de dígitos tem um bom desempenho, próximo a trabalhos como o apresentado por Salazar *et al.* (2017), cuja acurácia para dígitos foi de $98,3\%$, e até mesmo mais forte ao resultado apresentado por Galindo *et al.* (2016), de 96%, levando em consideração as diferenças nas bases de dados. Entretanto, o classificador de letras tem um desempenho inferior, com $10,01\%$ de diferença em relação ao trabalho de Salazar *et al.* (2017). Para detalhar os resultados individuais de cada dígito e letra, a Tabela 6 e a e a Tabela

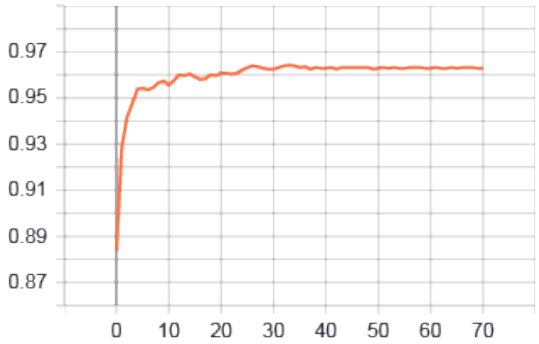
Figura 50 – Evolução da aprendizagem



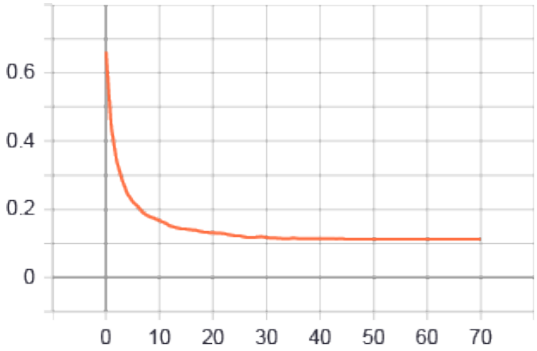
(a) Treinamento dos dígitos: Acurácia por épocas



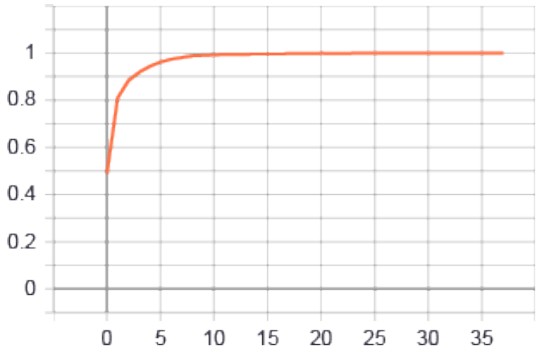
(b) Treinamento dos dígitos: Loss por épocas



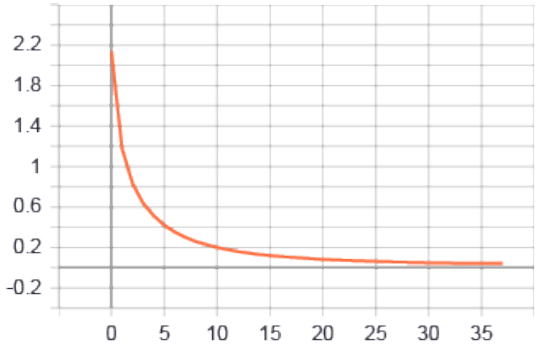
(c) Validação dos dígitos: Acurácia por épocas



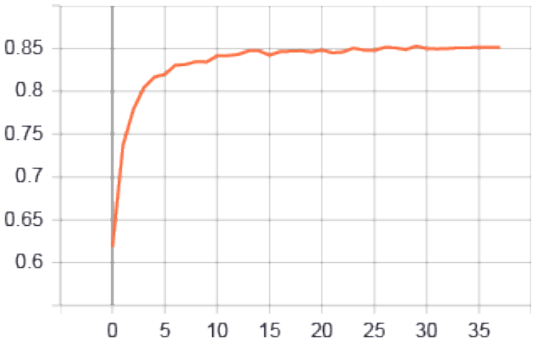
(d) Validação dos dígitos: Loss por épocas



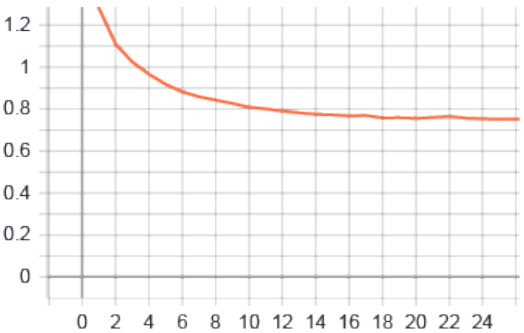
(e) Treinamento das letras: Acurácia por épocas



(f) Treinamento das letras: Loss por épocas



(g) Validação das letras: Acurácia por épocas



(h) Validação das letras: Loss por épocas

Fonte: elaborado pelo autor.

7 mostram a precisão e a sensibilidade para cada dígito e para cada letra, respectivamente, no conjunto de teste.

Tabela 6 – Resultados individuais por dígito (com valores de 0 a 1)

Dígito	Precisão	Sensibilidade	Número de exemplos
0	1.00	0.99	600
1	0.78	0.92	600
2	1.00	0.98	360
3	0.98	1.00	420
4	0.99	1.00	540
5	0.98	0.99	540
6	0.98	1.00	720
7	0.89	1.00	390
8	1.00	0.91	810
9	0.99	0.84	780

Fonte: Elaborado pelo autor

Tabela 7 – Resultados individuais por letra (de 0 a 1)

Letra	Precisão	Sensibilidade	Número de exemplos
A	0.98	0.99	1200
B	0.87	0.98	300
C	0.97	1.00	120
D	0.45	0.15	60
E	1.00	0.99	90
F	1.00	1.00	30
G	1.00	1.00	60
H	0.61	0.78	60
I	0.66	0.96	150
J	1.00	0.64	90
K	0.76	0.93	60
L	1.00	0.99	120
M	0.73	0.82	180
N	0.00	0.00	30
O	0.30	0.33	90
P	0.83	1.00	240
Q	0.43	0.67	60
R	1.00	0.25	120
S	1.00	1.00	180
T	0.20	0.08	60
U	0.65	1.00	90
V	0.51	0.92	150
W	1.00	0.29	240
X	1.00	0.64	90
Y	0.98	0.91	240
Z	0.99	1.00	210

Fonte: Elaborado pelo autor

5.4 Leitura da placa

Nesta seção, a leitura da placa será considerada como o resultado de todo o sistema de ALPR, utilizando os modelos obtidos e técnicas descritas nas seções anteriores. Para a análise dos resultados, mostrados na Tabela 8, é considerada a acurácia no conjunto de placas cinza da base UFPR-ALPR, onde a acurácia é a divisão do número de placas corretas pelo total de placas. Além da acurácia do texto inteiro da placa, também são levadas em consideração as acurácias para pelo menos cinco caracteres corretos e pelo menos seis caracteres corretos.

Tabela 8 – Resultado da leitura das placas com o sistema proposto

Conjunto	Acurácia (texto completo)	Acurácia ($\geq 6 caracteres$)	Acurácia ($\geq 5 caracteres$)
Treinamento	50,92%	63,42%	71,20%
Teste	29,53%	46,20%	56,11%

Fonte: Elaborado pelo autor

Os resultados mostram um desempenho visivelmente inferior aos resultados do método proposto por Laroca *et al.* (2018), onde a base UFPR-ALPR foi proposta, principalmente considerando que as placas vermelhas e placas de motocicleta estão fora do escopo deste trabalho. A Tabela 9 mostra uma comparação entre os resultados do método proposto, os trabalhos relacionados apresentados no Capítulo 3, levando em consideração as bases de dados utilizadas nos próprios trabalhos, e métodos comerciais na base UFPR-ALPR. O fator de maior relevância para esse resultado é a etapa de segmentação, que não é muito efetiva, já que utiliza técnicas muito simples. Então a Tabela 9 mostra também a acurácia do método proposto supondo uma segmentação perfeita, ou seja, o melhor resultado possível alterando apenas a etapa de segmentação.

Tabela 9 – Comparação dos resultados em diferentes métodos

Conjunto	Acurácia (texto completo)	Acurácia ($\geq 6 caracteres$)	Base de dados
Proposto	29,53%	46,20%	UFPR-ALPR[1]
Proposto (segmentação perfeita)	62,12%	87,12%	UFPR-ALPR[1]
OpenALPR (MATTHEW HILL, 2014)	50,94%	54,72%	UFPR-ALPR
Sighthound (MASOOD *et al.*, 2017)	47,39%	62,50%	UFPR-ALPR
Laroca *et al.* (2018)	64,89%	87,33%	UFPR-ALPR
Silva e Jung (2017)	63,18%	90,55%	SSIG
Salazar *et al.* (2017)	92,20%		comercial
Galindo *et al.* (2016)	76,00%		GNSS

Fonte: Elaborado pelo autor

[1] apenas placas de carro cinza no conjunto de teste.

6 CONCLUSÃO

6.1 Considerações gerais

Neste trabalho foi elaborada uma proposta de sistema para reconhecimento automático de placas veiculares brasileiras, abrangendo as três etapas comuns para a leitura de uma placa veicular: detecção da placa, segmentação dos caracteres e reconhecimento dos caracteres segmentados. Para desenvolver a proposta foram analisados trabalhos da literatura com objetivos similares, averiguando que os seus sistemas podem ter resultados com ambientes mais específicos ou técnicas muito complexas computacionalmente.

O sistema proposto consiste na utilização do método MMOD para detecção das placas, como descrito na Seção 4.2, aplicar um pré-processamento de imagens e regressão linear para segmentar os caracteres, como descrito na Seção 4.3 e o uso de redes neurais convolucionais para reconhecer os caracteres, como descrito na Seção 4.4. O método de detecção das placas veiculares se provou um método promissor, já que com uma limitação forte de equipamento conseguiu alguns resultados equiparáveis a outros trabalhos no tema, como mostrado na Seção 5.1. Além disso, a divisão do reconhecimento de caracteres entre um classificador para letras e outro para dígitos garantiu um bom resultado para a classificação de dígitos, conforme os resultados da Seção 5.3. Com o estudo realizado e os métodos propostos, os objetivos específicos deste trabalho foram cumpridos.

Algumas dificuldades foram encontradas, como a etapa de segmentação que, por ser um protótipo muito simples, apresentou resultados que implicaram em uma acurácia da leitura de $29,53\%$, como mostrado na Seção 5.4. Entretanto os resultados obtidos mostram um limite superior de $62,12\%$ caso a segmentação fosse perfeita, sem outras alterações, o que demonstra o potencial das técnicas discutidas.

6.2 Trabalhos Futuros

Para contornar as limitações de recursos encontradas na implementação do método de detecção, uma alternativa poderia ser uma etapa anterior a esta que realize a detecção do veículo. Isso diminui a quantidade de memória necessária para realizar os treinamentos e pode diminuir substancialmente a quantidade de falsos positivos. Além disso, para tornar o método mais robusto, podem ser explorados novos extratores de características, como, por exemplo,

algumas camadas convolucionais de uma rede neural convolucional.

A principal dificuldade deste trabalho foi a etapa de segmentação. Para obter resultados mais fortes, podem ser estudadas técnicas mais robustas, utilizando redes neurais artificiais, ou uma nova sequência de pré-processamentos na imagem.

Enfim, para melhorar a classificação de letras, que apresentou o pior resultado na etapa de classificação, podem ser estudadas alterações na arquitetura da rede neural convolucional ou a utilização de uma outra arquitetura conhecida na literatura.

REFERÊNCIAS

CASTELAN, L. **Reconhecimento Automático de Placas Veiculares (RAPV)**. [S.l.]: Universidade Federal de Santa Catarina, 2009.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE COMPUTER SOCIETY. **international Conference on computer vision & Pattern Recognition (CVPR'05)**. [S.l.], 2005. v. 1, p. 886–893.

DONOSER, M.; ARTH, C.; BISCHOF, H. Detecting, tracking and recognizing license plates. In: SPRINGER. **Asian Conference on Computer Vision**. [S.l.], 2007. p. 447–456.

DU, S.; IBRAHIM, M.; SHEHATA, M.; BADAWY, W. Automatic license plate recognition (alpr): A state-of-the-art review. **IEEE Transactions on circuits and systems for video technology**, IEEE, v. 23, n. 2, p. 311–325, 2012.

GALINDO, J. C. F.; CASTRO, C.; BRAGA, A. Sistema automático para reconhecimento de placas de automóveis baseado em processamento digital de imagens e redes perceptron de múltiplas camadas. In: . [S.l.: s.n.], 2016.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. [S.l.: s.n.], 2011. p. 315–323.

GONÇALVES, G. R.; MENOTTI, D.; SCHWARTZ, W. R. License plate recognition based on temporal redundancy. In: IEEE. **2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.], 2016. p. 2577–2582.

GONÇALVES, G. R.; SILVA, S. P. G. da; MENOTTI, D.; SCHWARTZ, W. R. Benchmark for license plate character segmentation. **Journal of Electronic Imaging**, International Society for Optics and Photonics, v. 25, n. 5, p. 053034, 2016.

GONZALEZ, R. C.; WOODS, R. E. **Digital image processing**. Upper Saddle River, N.J.: Prentice Hall, 2008.

HAHNLOSER, R. H.; SARPESHKAR, R.; MAHOWALD, M. A.; DOUGLAS, R. J.; SEUNG, H. S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. **Nature**, Nature Publishing Group, v. 405, n. 6789, p. 947, 2000.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015.

JOACHIMS, T.; FINLEY, T.; YU, C.-N. J. Cutting-plane training of structural svms. **Machine Learning**, Springer, v. 77, n. 1, p. 27–59, 2009.

KING, D. E. Max-margin object detection. **arXiv preprint arXiv:1502.00046**, 2015.

LAROCA, R.; SEVERO, E.; ZANLORENSI, L. A.; OLIVEIRA, L. S.; GONÇALVES, G. R.; SCHWARTZ, W. R.; MENOTTI, D. A robust real-time automatic license plate recognition based on the yolo detector. In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2018. p. 1–10.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Taipei, Taiwan, v. 86, n. 11, p. 2278–2324, 1998.

MASOOD, S. Z.; SHU, G.; DEHGHAN, A.; ORTIZ, E. G. License plate detection and recognition using deeply learned convolutional neural networks. **arXiv preprint arXiv:1703.07330**, 2017.

MATASYX, J.; KITTLERY, C. G. J. Progressive probabilistic hough transform. 1998.

MATTHEW HILL. **OpenALPR Cloud API**. 2014. Disponível em: <http://www.openalpr.com/cloud-api.html>.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. Activation functions: Comparison of trends in practice and research for deep learning. **arXiv preprint arXiv:1811.03378**, 2018.

OPENCV. **OpenCV: Histogram Equalization**. 2019. Disponível em: <https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html>. Acesso em: 30 de maio de 2019.

OPENCV. **OpenCV: Morphological Transformations**. 2019. Disponível em: <https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html>. Acesso em: 30 de maio de 2019.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 9, n. 1, p. 62–66, 1979.

RAJARAMAN, A.; LESKOVEC, J.; ULLMAN, J. D. **Mining Massive Datasets**. [s.n.], 2014. Disponível em: <http://www.mmds.org/>.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 7263–7271.

ROSENBLATT, F. **The perceptron, a perceiving and recognizing automaton Project Para**. [S.l.]: Cornell Aeronautical Laboratory, 1957.

ROSENFELD, A.; KAK, A. C. **Digital Picture Processing**. New York: Academic Press, 1982.

SALAZAR, J. A. D. *et al.* Reconhecimento automático de placas veiculares brasileiras em ambientes não controlados. 2017.

SHAFIEE, M. J.; CHYWL, B.; LI, F.; WONG, A. Fast yolo: a fast you only look once system for real-time embedded object detection in video. **arXiv preprint arXiv:1709.05943**, 2017.

SILVA, S. M.; JUNG, C. R. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In: IEEE. **2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.], 2017. p. 55–62.

SOBEL, I. An isotropic 3$\times$ 3 image gradient operater. **Machine vision for three-dimensional scenes**, Academic Press, p. 376–379, 1990.

SUZUKI, S. *et al.* Topological structural analysis of digitized binary images by border following. **Computer vision, graphics, and image processing**, Elsevier, v. 30, n. 1, p. 32–46, 1985.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. London; New York: Springer, 2010.

TIBSHIRANI, R.; WAINWRIGHT, M.; HASTIE, T. **Statistical learning with sparsity: the lasso and generalizations**. [S.l.]: Chapman and Hall/CRC, 2015.

VIOLA, P.; JONES, M. *et al.* Rapid object detection using a boosted cascade of simple features. **CVPR (1)**, v. 1, p. 511–518, 2001.

VOGEL, J. **Conheça todos os modelos de placas de veículos já usados no Brasil**. 2018. Disponível em: <https://oglobo.globo.com/economia/carros/conheca-todos-os-modelos-de-placas-de-veiculos-ja-usados-no-brasil-23063276>. Acesso em: 30 de maios de 2019.

WANG, Y.; FANG, B.; LAN, L.-J.; LUO, H.-W.; TANG, Y.-Y. Adaptive binarization: a new approach to license plate characters segmentation. In: IEEE. **2012 International Conference on Wavelet Analysis and Pattern Recognition**. [S.l.], 2012. p. 91–99.

ZAITOUN, N. M.; AQEL, M. J. Survey on image segmentation techniques. **Procedia Computer Science**, Elsevier, v. 65, p. 797–806, 2015.

APÊNDICE A – REPOSITÓRIO COM O CÓDIGO E MODELOS DO TRABALHO

O código desenvolvido neste trabalho e os modelos e pesos dos treinamentos estão disponíveis no link <https://github.com/lucasfernandes42/br-alpr-mmod>.