Centro Federal de Educação Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Guilherme Antônio Rezende

Sistema de monitoramento em tempo real de fadiga de motoristas

Divinópolis
2022

Guilherme Antônio Rezende

Sistema de monitoramento em tempo real de fadiga de motoristas

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Eletrônica e Programação.

Orientador: Prof. Doutor Thiago Magela Rodrigues Dias
Co-orientador: Eng. Lucas Dias Fonseca

Divinópolis
2022

Guilherme Antônio Rezende

Sistema de monitoramento em tempo real de fadiga de motoristas

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Eletrônica e Programação.

Comissão Avaliadora:

Prof. Doutor Thiago Magela Rodrigues Dias
CEFET/MG *Campus* V

Prof. Doutor Alan Mendes Marotta
CEFET/MG *Campus* V

Prof. Raulivan Rodrigo da Silva
CEFET/MG *Campus* V

Divinópolis
2022

"A grande tarefa humanista e histórica dos oprimidos é libertar a si mesmo e a seus opressores."

Paulo Freire

Resumo

A fadiga ou sonolência de condutores é uma das principais causas de acidentes de trânsito no Brasil, visto que a mesma incapacita o condutor para a realização das atividades necessárias para a condução de automóveis. Neste trabalho é apresentado um sistema de monitoramento e indicação de fadiga em motoristas com intuito garantir a segurança em ambientes onde a atenção é um fator de risco. Técnicas não intrusivas de identificação de fadiga são discutidas e implementadas através do emprego de visão computacional. O desenvolvimento desse projeto passa pelo treinamento de um modelo preditivo que rastreia e monitora o estados dos olhos do condutor como principal indicador de fadiga, sendo essa tarefa realizada através do uso de algoritmos de aprendizagem de máquina e redes neurais artificiais. O sistema é embarcado em um *Raspberry Pi*, e é composto por uma câmera de monitoramento, centrada na face do motorista, e um sistema de sinalização.

Palavras-chave: Visão computacional, Detector de fadiga, *Python*.

Abstract

Driver fatigue or drowsiness is one of the main causes of traffic accidents in Brazil, as it disables the driver to perform the activities necessary for driving cars. This work presents a monitoring system that indicates fatigue in drivers in order to ensure safety in environments where attention is a risk factor. Non-intrusive fatigue identification techniques are discussed and implemented using computer vision. The development of this project involves the training of a predictive model that tracks and monitors the state of the driver's eyes as the main indicator of fatigue, this task is carried out through the use of machine learning algorithms and artificial neural networks. The system is embedded in a *Raspberry Pi* microcontroller, and is composed of a monitoring camera, centered on the driver's face, and a signaling system.

Keywords: Computer Vision, Fatigue Detector, *Python*.

Sumário

Lista de Figuras **xi**

Lista de Tabelas **xii**

Lista de Acrônimos e Notação **xiii**

1 Introdução **1**
1.1 Definição do Problema . . . 1
1.2 Motivação . . . 2
1.3 Objetivos do Trabalho . . . 2
1.3.1 Objetivos Gerais . . . 2
1.3.2 Objetivos Específicos . . . 2
1.4 Organização do Documento . . . 3

2 Revisão bibliográfica **4**
2.1 Estado da Arte . . . 4
2.2 Fundamentação Teórica . . . 5
2.2.1 Visão Computacional . . . 5
2.2.2 Processamento de Imagens . . . 7
2.2.3 Otimização . . . 12
2.2.4 Aprendizado de Máquina . . . 17
2.2.5 Detecção Facial . . . 18
2.2.6 Redes Neurais Artificiais . . . 22
2.2.7 Raspberry pi . . . 27

3 Metodologia **28**
3.1 Escopo do Trabalho . . . 28
3.2 Linguagem de programação utilizada . . . 31
3.3 Bibliotecas utilizadas . . . 31
3.4 Datasets utilizados . . . 32
3.5 Microcontrolador empregado . . . 35
3.6 Câmera empregada . . . 36

4 Desenvolvimento **38**
4.1 Treinamento HOG . . . 38
4.2 Treinamento CNN para Detecção e Reconhecimento Ocular . . . 39

4.3 Implementação do Algoritmo de Detecção de Fadiga 41
4.4 Sistema Embarcado 41

5 Resultados e Discussões **44**
5.1 Treinamento 44
5.2 Testes Não-Embarcados 45
5.3 Testes Embarcados 48
5.3.1 Limitações - Inclinação da Face 49
5.3.2 Limitações - Ocultação da Face 50
5.3.3 Limitações - Distância Entre Face e Câmera 53
5.3.4 Limitações - Iluminação 54

6 Conclusões Finais **56**
6.1 Conclusão 56
6.2 Tabela de Custos 56
6.3 Trabalhos Futuros 57

A Algoritmos **58**
A.1 Algoritmo de determinação de hiperparâmetro de regularização para detecção facial 58
A.2 Algoritmo de modelo de detecção ocular 58
A.3 Algoritmo de *data augmentation* 59
A.4 Algoritmo de modelo de classificação ocular 60
A.5 Algoritmo de sinalização de fadiga 60

Referências **62**

Lista de Figuras

2.1 Elementos de processamento de imagens. (FONTE: MARQUE FILHO O.; NETO (1999)) . . . 6
2.2 Representação de imagem nas escalas de cores colorida (RGB) e cinza. (FONTE: SCIKIT-IMAGE (2020)) . . . 8
2.3 Histograma em escala cinza. (FONTE: CARPENTRY (2021)) . . . 9
2.4 Convolução de matrizes. . . . 10
2.5 Aplicação de *Data Augmentation*. . . . 12
2.6 Deslocamento do gradiente no espaço solução da função custo. (FONTE: LAMBLET VAZ (2020)) . . . 14
2.7 Histograma de gradientes orientados de uma imagem. . . . 19
2.8 Soluções possíveis, sendo a imagem a direita produto a otimização, e consequentemente apresenta maior margem. (FONTE: L. BRUNTON; KUTZ (2017)) . . . 20
2.9 Método Non Maximum Suppression. (FONTE: PRAKASH (2021)) . . . 21
2.10 Arquitetura genérica de uma ANN. (FONTE: L. BRUNTON; KUTZ (2017)) 22
2.11 Rede neural com apenas uma camada intermediária. (FONTE: L. BRUNTON; KUTZ (2017)) . . . 23
2.12 Arquitetura de rede neural convolucional. (FONTE: L. BRUNTON; KUTZ (2017)) . . . 25
2.13 A função sigmoid encapsula números reais dentro de uma faixa de [0, 1]. (FONTE: STANFORD (2021)) . . . 26
2.14 Camada de regularização dropout. (FONTE: BUDHIRAJA (2021)) . . . 26
2.15 Raspberry pi. (FONTE: FOUNDATION (2022)) . . . 27

3.1 Fluxograma do trabalho. . . . 28
3.2 Diagrama de fluxo de dados . . . 30
3.3 Amostras positivas do dataset LFW. . . . 33
3.4 Amostras negativas do dataset LFW. . . . 33
3.5 Landmark dataset. . . . 34
3.6 Dataset empregado na classificação dos olhos. . . . 35
3.7 Câmera para aquisição de imagens. . . . 36

4.1 Pinagem do *Raspberry pi 3 B. (FONTE: GAY (2018))* . . . 42
4.2 Sistema de Alerta . . . 43

5.1 Teste para detecção facial . . . 46

5.2 Teste para detecção ocular 46
5.3 Recorte do *frame* para olho aberto (a) e fechado (b) 47
5.4 *frame* que indica se os olhos estão aberto (a) ou fechados (b) 47
5.5 Recorte do *frame* para olho aberto (b), (c) e (d) e fechado (a) 48
5.6 Teste embarcado em condições ideias com olhos abertos 49
5.7 Teste embarcado em condições ideias com olhos fechados 49
5.8 Teste embarcado com inclinação acentuada do rosto. 50
5.9 Teste embarcado com leve inclinação do rosto. 50
5.10 Teste embarcado para condutor com máscara 51
5.11 Teste embarcado para condutor com boné 51
5.12 Teste embarcado para condutor com óculos 52
5.13 Teste embarcado para condutor com gorro 52
5.14 Teste embarcado para condutor com capuz 53
5.15 Teste embarcado para condutor a cerca de 1m metro da câmera 54
5.16 Teste embarcado com baixa iluminação com olhos abertos 55
5.17 Teste embarcado com baixa iluminação com olhos fechados 55

Lista de Tabelas

3.1 Tabela de especificações *Raspberry pi* 36

5.1 Desenvolvimento das métricas perda e precisão ao longo do treinamento . . 45

Lista de Acrônimos e Notação

ABRAMET	Associação Brasileira de Medicina no Tráfego
AdaGrad	Adaptive Gradient
ANN	Artificial Neural Networks
CNN	Convolucional Neural Networks
FPS	Frames Per Second
GD	Gradient Descent
HOG	Histograma de Gradiente Orientados
MIT	Massachusetts Institute of Technology
MSE	Mean Squared Error
NMS	Non Max Supression
OMS	Organização Mundial da Saúde
PPP	Pixels Por Polegada
PRF	Polícia Rodoviária Federal
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RMSProp	Root Mean Squared Propagation
SGD	Stochastic Gradient Descent
SUS	Sistema Único de Saúde
SVM	Support Vector Machine

α	Taxa de aprendizado
λ	Comprimento de Onda
γ	Taxa de decaimento RMSProp
β	Taxa de decaimento Adam
In	Polegada
vc	Viés de correção

Capítulo 1

Introdução

Segundo a Associação Brasileira de Medicina no Tráfego (ABRAMET), a falta de atenção ao volante englobam a maioria dos casos catalogados de acidente de trânsito em estradas e rodovias supervisionadas pela Polícia Rodoviária Federal (PRF) (ABRAMET, 2020). Essa causa se relaciona a situações clínicas como fadiga, stress, cansaço e déficit de atenção, e a sonolência . Motoristas profissionais e portadores de carteiras de habilitação C, D e E, são responsáveis por 60% dos acidentes rodoviários, 42% relacionados ao sono e 18% a fadiga (PERKONS, 2018).

O financiamento proveniente de instituições estaduais de trânsito para o aumento da fiscalização, programas de conscientização, melhorias das condições de trabalho de motoristas profissionais, ou aplicação de infrações mais severas são exemplos de abordagens para a prevenção de acidentes relacionados à fadiga. Entretanto, com o crescimento exponencial da capacidade computacional na última década (MIT, 2020), concatenada as teorias de engenharia de dados desenvolvidas ao longo do século XX e XXI (GUPTA; EFROS; HEBERT, 2010; PAPERT, 1966; MARR, 1970), soluções mais robustas podem ser elaboradas.

Nesse contexto, as áreas de tratamento de dados e visão computacional ganharam capilaridade na produção científica e industrial. Consequentemente, modelos de alta complexidade para a construção de sistemas de monitoramento e alerta em tempo real da condição de direção de motoristas podem ser desenvolvidos.

1.1 Definição do Problema

Para o desenvolvimento do sistema de monitoramento de fadiga de condutores, o problema pode ser interpelado de diversas maneiras (DEVI; BAJAJ, 2008). O método fisiológico, ainda que acurado, requer uma abordagem invasiva, sendo dispostos eletrodos no corpo do motorista (ERIKSSON; PAPANIKOTOPOULOS, 1997) para o sensoriamento de sinais como ondas cerebrais, frequência cardíaca, pulsação e respiração.

Já o método físico é não intrusivo, e monitora sinais como postura corporal, inclinação da cabeça do condutor e estado dos olhos, abertos ou fechados. Dessa maneira, não há interferência nas condições de direção, visto que os sinais monitorados podem ser capturados e analisados através de uma câmera.

O processamento de imagens é necessário para a implementação eficiente deste último método, que é desenvolvido tal que o erro seja otimizável e a resposta dos sistema seja a mais rápida possível, considerado o hardware empregado para a tarefa. Em relação aos erros admitidos pelo projeto, são consideradas as condições de operação do processo, como iluminação, ângulo de visão da câmera e ocultação do objeto (POKHREL, 2020).

Portanto, o projeto trata de solucionar o problema de desenvolvimento de uma metodologia não invasiva e de baixo custo computacional, para aplicação em tempo real, de detecção de fadiga em condutores. Deste modo, são investigadas as principais tecnologias para detecção de fadiga através do monitoramento facial de um indivíduo.

1.2 Motivação

Devido aos números elevados de acidentes de trânsito apresentados pela Organização Mundial da Saúde (OMS) (SBMT, 2019), o trabalho proposto se mostra relevante no âmbito social. A aplicação deste trabalho visa garantir que o condutor se mantenha atento ao tráfego de veículos e alerta aos primeiros sinais de sonolência, o que consequentemente atenuaria a quantidade de acidentes de trânsito relacionados à fadiga do motorista.

Além disso, o trabalho tem potencial de comercialização devido a demanda de países subdesenvolvidos, como o Brasil, por uma solução para o problema crônico de mortes no trânsito. Deste modo, é possível onerar os gastos públicos do Sistema Único de Saúde (SUS) para esse tipo de ocorrência (CREMEB, 2019).

1.3 Objetivos do Trabalho

1.3.1 Objetivos Gerais

O objetivo geral deste trabalho é o desenvolvimento de um sistema de detecção de fadiga de condutores em tempo real através de monitoramento de imagem. Técnicas de visão computacional, como processamento de imagens, e a modelagem via aprendizado de máquina são empregados para a tarefa.

1.3.2 Objetivos Específicos

O sistema de monitoramento, embarcado em um microcontrolador, deve ser capaz de processar e manipular em tempo real os dados recolhidos pela câmera. Os métodos

utilizados no desenvolvimento do algoritmo são sequenciais, de modo que o projeto segue as etapas:

- Capturar e processar imagens;
- Implementar código de detecção de face;
- Implementar código de reconhecimento da geometria facial;
- Implementar código de reconhecimento dos olhos;
- Parametrizar o estado dos olhos e utilizá-lo como indicador de fadiga;
- Testar e analisar resultados obtidos pelo algoritmo;
- Embarcar algoritmo em *Raspberry Pi*;
- Elaborar alerta visual para o sistema;
- Testar e analisar resultados finais;

1.4 Organização do Documento

O trabalho foi dividido nas seções:

- Capitulo 1: Introdução;
- Capitulo 2: Revisão Bibliográfica;
- Capitulo 3: Metodologia;
- Capítulo 4: Desenvolvimento;
- Capítulo 5: Resultados e Discussões; e
- Capítulo 6: Conclusões Finais;

Capítulo 2

Revisão bibliográfica

Este capítulo se dispõe a apresentar, no tópico *Estado da Arte*, os trabalhos mais recentes e relevantes na elaboração das tecnologias empregadas nesse estudo. Enquanto a seção *Fundamentação teórica* aborda conceitos e fundamentos teóricos necessários para o desenvolvimento das ferramentas utilizadas nesse projeto.

2.1 Estado da Arte

O número de acidentes de trânsito causados por fadiga do condutor tem provocado o surgimento de dispositivos auxiliares de condução, desenvolvidos principalmente por companhias de automóveis. No âmbito das investigações acadêmicas sobre o área de detecção de fadiga de motoristas, a última década promoveu diversos avanços impostos pela propagação de tecnologias relacionadas à inteligência artificial.

A *Nissan* implementou um sistema adaptativo ao comportamento do condutor em seus veículos, que identifica o padrão de direção, e a detecção de qualquer desvio desse gera um sinal de alerta, e se necessário freia o carro (NISSAN, 2021). Já a *Volkswagen* desenvolveu o auxiliar de descanso, que monitora o estado da pista, o uso do pedal e movimentos erráticos do volante para determinar se o condutor apresenta fadiga (VOLKSWAGEN, 2021).

O monitoramento do estado de atenção do condutor é o parâmetro crítico para a condução veicular, visto que a fadiga diminui o tempo de resposta do motorista. Sintomas de fadiga incluem bocejo (YANG *et al.*, 2020), tempo de reação lento (BINIAS *et al.*, 2020), pálpebras fechadas (KIM *et al.*, 2017), pegada frouxa no volante (BALASUBRAMANIAN; BHARDWAJ, 2018), dentre outros.

O método dos histogramas de gradientes orientados (HOG) é um dos mais utilizados e populares para a detecção facial. Este é empregado por PAULY; SANKAR (2016) associado a técnica *Support Vector Machine* (SVM) para a classificação das imagens.

CHOU; HUANG; HO (2019) interpelam o problema de detecção facial estimando a

distância entre os cílios, sendo necessária a implementação de uma rede neural que otimize através de regressão linear o modelo geométrico facial, de modo que são obtidos diversas coordenadas do rosto referentes a seus traços mais relevantes.

Os trabalhos na área que apresentam caráter mais prático tendem a embarcar tais soluções no microcontrolador *Raspberries pi*, devido a alta capacidade de processamento do dispositivo. (JAIN A. K.; SHARMA, 2018) utiliza em seu projeto esse tipo de hardware, aplicado a identificação de intrusos em domicílios.

Por fim, JORDAN *et al.* (2020) e DUA *et al.* (2021), elaboram projetos de detecção de fadiga, com os olhos como indicadores, que empregam redes neurais convolucionais (CNN) para a obtenção de parâmetros ótimos. A vantagem desse tipo de trabalho é o desenvolvimento offline do modelo, na etapa de treinamento, condicionada pela escolha com parcimônia entre custo computacional e precisão.

2.2 Fundamentação Teórica

O sistema elaborado deve ser capaz de adquirir e processar as informações recolhidas pelo sensor tal que alerte o motorista em caso de sonolência. Esta seção se dedica a explorar técnicas, métodos e processos para o desenvolvimento do trabalho, sendo dividido nos seguintes tópicos:

- Visão Computacional
- Processamento de Imagens
- Otimização
- Aprendizado de Máquina
- Detecção facial
- Redes Neurais Artificiais
- Raspberry pi

2.2.1 Visão Computacional

As aplicações da área de visão computacional crescem substancialmente com a viabilidade recente de utilização de modelos baseados em aprendizado de máquina. Esse tipo de sistema pode ser entendido através de quatro principais elementos de processamento de imagens: aquisição, processamento, saída e armazenamento, como aponta (MARQUE FILHO O.; NETO, 1999) na figura 2.1.

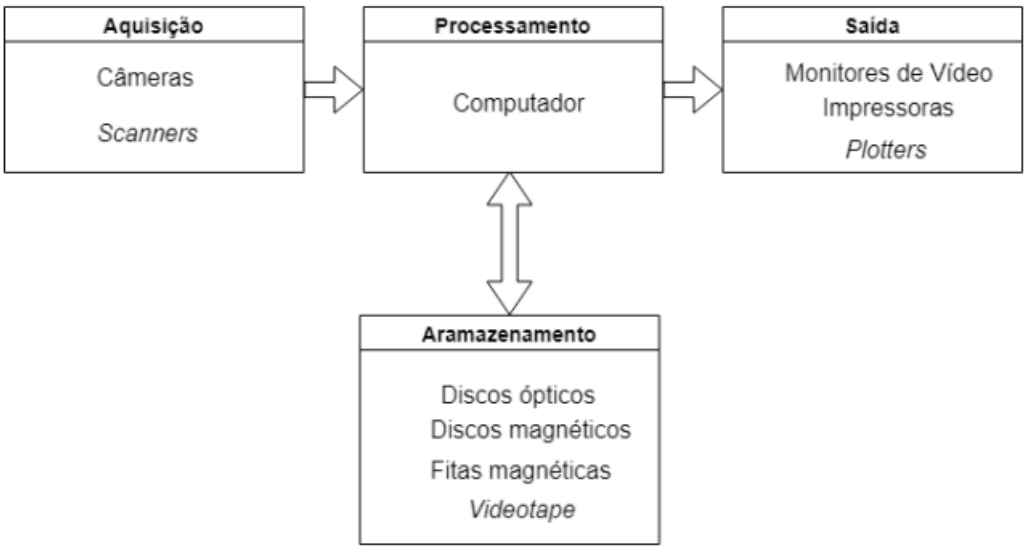


Figura 2.1: Elementos de processamento de imagens. (FONTE: MARQUE FILHO O.; NETO (1999))

Os elementos descritos acima são explicados em detalhe a seguir:

- **Aquisição:** Para a realização dessa etapa é necessário que o sistema tenha um dispositivo físico fotossensível que produza um sinal de saída proporcional ao de entrada, e um conversor analógico-digital para a transformação de um sinal elétrico em uma sequência binária. Desta maneira, essa parte do processamento se encarrega de converter a imagem em uma representação numérica adequada.

- **Armazenamento:** Esse elemento é essencial para o desempenho do sistema projetado, principalmente quando são empregados modelos baseados em aprendizado profundo. O armazenamento do sistema pode ser categorizado em três categorias: de curta duração, que utiliza a memória RAM do computador principal ou de periféricos dedicados, de massa de recuperação rápida, que se refere ao tempo de acesso de arquivos, e arquivamento para recuperação futura, que se caracteriza por arquivar imensas quantidade de dados.

- **Processamento:** Essa etapa é implementada através da utilização de software. As técnicas e tecnologias empregadas são direcionadas de acordo com os objetivos do projeto.

- **Saída:** Responsável pela transmissão e exibição do resultado do processamento. Técnicas de compressão e descompressão são usualmente propostas para melhorar a velocidade da transmissão.

2.2.2 Processamento de Imagens

Conceito de Imagem Digital

Os dados recolhidos no processo de aquisição de imagens, através de uma câmera de monitoramento, sob forma de sinal analógico $I(x,y)$, são submetidos a uma discretização espacial. Este procedimento consiste no recolhimento de amostras da função contínua $I(x,y)$ de mapeamento da intensidade de luz, que representa uma imagem digital (MARQUES FILHO; VIEIRA NETO, 1999). Esta assume a forma de uma matriz de dimensão MxN, descrita abaixo:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,N-1) \\ f(1,0) & f(1,1) & ... & f(1,N-1) \\ \vdots & \vdots & ... & \vdots \\ f(M-1,0) & f(M-1,1) & ... & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

A imagem digital é composta de um número finito de elementos, cada um atribuído de localização e intensidade, sendo estes denominados elementos pictóricos ou pixels (C. GONZALEZ; E. WOODS, 2010). Estes são manipulados e adaptados aos diferentes propósitos de um projetista através de transformações, seja de intensidade ou filtragem espacial.

A resolução de uma imagem é o seu número de pixels por polegada (PPP), sendo esta uma métrica utilizada na avaliação da nitidez de uma imagem. As categorias a seguir são consideradas na determinação da resolução de uma imagem digital (BOTTER MARTINS, 2020):

- Espacial: Para uma mesma região no espaço (1 in), quanto maior o número de pixels, maior será a resolução da imagem;
- Profundidade da Imagem: Número de bits, b, da imagem. O número de níveis de cinza de uma imagem $I(x,y)$ é 2^b;
- Espectral: Em uma mesmo intervalo de espectro de luz, a maior resolução espectral está associada ao maior número de bandas, ou canais, que são componentes primários na formação do espaço cor; e
- Temporal: Referente a frame de vídeo, de modo que a resolução aumente em relação direta com o número de quadros por segundo (FPS);

Espaço de Cores

Os conceitos de processamento de imagem são intuitivamente construídos, de modo que a sua interpretação provenha da analogia com a biologia do olho e da percepção

sensorial humana. Uma cor apresenta três componentes, geralmente, representados pela combinação de cores monocromáticas nos comprimentos de onda (λ), azul, vermelho e verde (BOTTER MARTINS, 2020), sendo este chamado de canal RGB. As categorias abaixo qualificam estes canais de cores:

- Intensidade: Responsável pela sensação de brilho;
- Matiz: Sensação de cor, relacionada ao comprimento de onda (λ); e
- Saturação: Grau de pureza da cor em relação ao branco;

Imagens em escala cinza apresentam apenas o componente intensidade e, geralmente, são empregadas na manipulação de dados por ocuparem pouco espaço de memória, sendo mais comum o armazenamento de 8 bits. A transformação do espaço cor RGB para a escala cinza ocorre através da relação abaixo (POYNTON, 1997):

$$Y = 0{,}2125R + 0{,}7154G + 0.0721B \tag{2.2}$$

A figura 2.2 mostrada abaixo é submetida a transformação para escala de cores cinza.

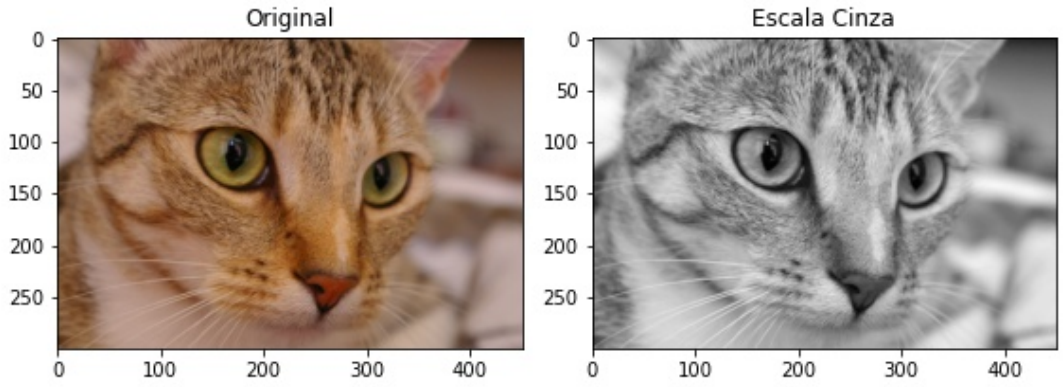


Figura 2.2: Representação de imagem nas escalas de cores colorida (RGB) e cinza. (FONTE: SCIKIT-IMAGE (2020))

O valor de intensidade de cada pixel de uma imagem em escala cinza é determinado através da soma ponderada de cada um dos componentes da imagem RGB (SCIKIT-IMAGE, 2020). Desse modo, é obtida, com suporte da biblioteca *sklearn*, a figura acima, proveniente da transformação da imagem original.

Histograma

O histograma é uma representação gráfica de uma imagem que fornece informação referente a frequência de ocorrência de um evento ou categoria, nesse caso, de uma intensidade de cor dentro de uma faixa de valores (CARPENTRY, 2021). Geralmente, os dados manipulados estão em escala cinza, de modo que este gráfico representa a distribuição de pixels em k níveis de cinza em uma imagem.

Segundo ANDRADE; ALBUQUERQUE; ALBUQUERQUE (2021), o histograma obedece a teorema e axiomas da teoria de probabilidades, visto que pode ser considerado uma função distribuição de probabilidades. A figura 2.3 exemplifica o processo de obtenção do histograma em escala cinza referente a imagem 2.2:

Figura 2.3: Histograma em escala cinza. (FONTE: CARPENTRY (2021))

As principais aplicações do histograma para o processamento de imagens é a segmentação e a detecção de objetos. Esse trabalho se dedica a utilizar o conceito na detecção de contornos, tal que seja possível identificar características faciais.

Filtragem Espacial

A filtragem de uma imagem, no domínio espacial, é um procedimento que visa a classificação de um conjunto de pixels sobreposto por uma máscara, de forma que seja realizado um mapeamento das características de interesse do projetista. Os filtros, ou *Kernel*, empregados na identificação de bordas, linhas e manchas são não-lineares e tem como objetivo a detecção do objeto através da suavização ou aguçamento dos pixels (C. GONZALEZ; E. WOODS, 2010).

O *Kernel* utilizado para o reconhecimento de contornos é calculado através do gradiente do pixel de interesse e seus vizinhos imediatos. Os filtros usados são descritos

abaixo:

$$Gx = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}; Gy = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T \tag{2.3}$$

Uma outra maneira de representar a operação é apresentada abaixo, sendo i e j os índices de linhas e colunas, respectivamente:

$$\begin{aligned} Gx &= I(i,j+1) - I(i,j-1) \\ Gy &= I(i-1,j) - I(i+1,j) \end{aligned} \tag{2.4}$$

Convolução

Convolução é uma operação que permite a multiplicação de matrizes de tamanhos diferentes, mas de mesma dimensionalidade, o que produz uma terceira matriz. Esse conceito é empregado no processamento de imagens e essencial para a aplicação de diversos operadores, dentre eles os filtros espaciais (CORNELL, 2013). A figura 2.4 mostrada a seguir ilustra a convolução de duas matrizes:

$$\begin{pmatrix} 4 & 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 4 & 0 & 0 & 4 & 4 & 0 & 8 \\ 9 & 1 & 1 & 0 & 5 & 0 & 7 \\ 1 & 1 & 0 & 0 & 5 & 8 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 8 & 2 & 4 & 3 & 3 \\ 2 & 5 & 7 & 4 & 1 \\ 5 & 4 & 4 & 1 & 4 \\ 3 & 3 & 2 & 1 & 4 \end{pmatrix}$$

Figura 2.4: Convolução de matrizes.

Esse conceito é recorrente na detecção de objetos, tal que a janela de operação da convolução realiza o rastreamento do objeto ao longo da imagem. O passo que essa janela efetua, tanto na horizontal quanto na vertical, é definido pelo projetista, e também é conhecido como *stride*.

Normalização

A normalização dos dados é um procedimento importante na prevenção de distorções no processo de aprendizado, visto que atributos em diferentes escalas são interpretados com ponderações distintas. Desta maneira os dados são submetidos a normalização tal que estejam dentro de uma mesma faixa de valores.

Além disso, a normalização dos dados evita redundâncias das amostras recolhidas, visto que dados duplicados não são relevantes na análise e distorcem os resultados finais

(IMPORT.IO, 2019). O custo computacional também é influenciado, posto que dados não-normalizados comprometem o processo de otimização, ao dificultar a convergência e aumentar o tempo de processamento do método (BOTTOU, 2021).

As normas L1 e L2, geralmente, são empregadas para a normalização de dados. O primeiro é calculado através da distância *Manhattan*, enquanto o último usa a distância euclidiana. Considerando um vetor v composto de três elementos, as normas L1 e L2 são mostradas, respectivamente, na sequência:

$$\begin{aligned} ||v||_1 &= |v_1| + |v_2| + |v_3| \\ ||v||_2 &= \sqrt{v_1^2 + v_2^2 + v_3^2} \end{aligned} \tag{2.5}$$

Com o crescimento da capacidade computacional, problemas com uma grande base de dados passaram a empregar a norma L1, que resulta em uma solução robusta. Isto porque esta norma é insensível a *outliers* e gera uma solução com um número mínimo de parâmetros (L. BRUNTON; KUTZ, 2017).

Data Augmentation

Data Augmentation é o método desenvolvido na área de machine learning que busca tornar mais robusto um banco de dados, além de melhorar seu desempenho. Isto se dá através da manipulação dos dados obtidos a priori, de modo a artificialmente utilizar dados nas mais diversas condições para o aprendizado (GANDHI, 2021).

Algumas das transformações mais comum nessa ferramenta é o recorte de pedaços de uma imagem, a rotação em determinado ângulo, a mudança de intensidade, a inversão vertical e horizontal da imagem, etc. *Data Augmentation* é bastante usado em problemas onde o banco de dados é pequeno, o que gera muita variância. A figura 2.2 é submetida ao método *Data Augmentation* e o resultado é exposto na figura 2.5:

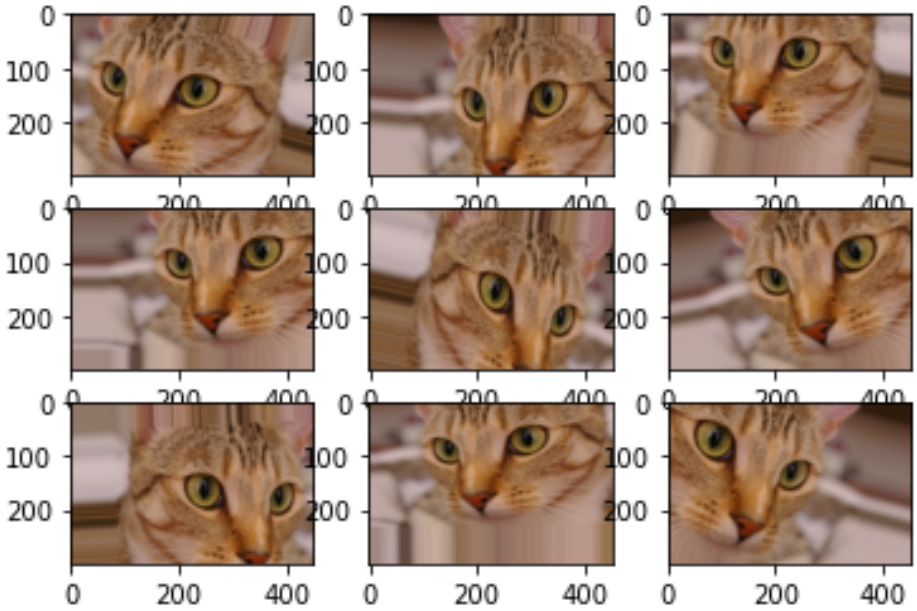


Figura 2.5: Aplicação de *Data Augmentation*.

A figura 2.5 exemplifica o funcionamento de diversas das transformações usualmente empregadas nesse tipo de técnica, sendo as nove imagens acima a representação da variedade de condições que o método pode artificialmente introduzir ao conjunto de dados. As transformações aplicadas foram: rotação de 20 radianos; zoom de 15 %; translação percentual da imagem na horizontal e na vertical em 20 %; e inversão na horizontal.

2.2.3 Otimização

O problema de otimização, geralmente aplicado na resolução de problemas de aprendizado de máquina e redes neurais, busca aprender e generalizar dados históricos, tal que consiga realizar predições sobre novos dados (MASTERY, 2021). As principais funções de custo, ou função objetivo, empregadas são a do método do mínimos quadrados e a entropia cruzada, para regressão e classificação, respectivamente.

Método dos Mínimos Quadrados

A otimização consiste de uma função de mapeamento parametrizada que visa a minimização do erro, que é denominada de função custo. Dado o conjunto de dados $\{(x1,y1)...(x_m,y_m)\}$, o erro associado pode ser calculado pela função custo MSE (*Mean Squared Error*), empregado em problemas de regressão, apresentada abaixo (LAMBLET VAZ, 2020):

$$J(\theta) = \frac{1}{2m}\sum_{i=0}^{m}(y_i - h_\theta(x_i))^2 \qquad h_\theta(x_i) = \theta_0 + \theta_1 x_i \tag{2.6}$$

Como descrito na equação 2.6, a otimização acontece através da minimização da diferença entre dados observados, y_i, e a predição realizada pelo modelo, $h_\theta(x_i)$. A definição do último cabe a análise que o projetista extrair do comportamento dos dados de entrada.

Entropia Cruzada

O método da Entropia Cruzada, ou Binary Cross-Entropy, fundamentado na teoria da probabilidade, é utilizado em problemas de classificação binária (GODOY, 2018). Deste modo, a equação para esse tipo de problema, considerando um conjunto de dados de tamanho n, assume a seguinte forma:

$$J = -\frac{1}{n}\sum_{i=1}^{n} y_i log(p(\overline{y_i})) + (1 - y_i) log(1 - p(\overline{y_i})) \tag{2.7}$$

A equação 2.7 é um tipo de função custo empregada em problemas binários, geralmente de classificação. Esta tende a penalizar mais que a equação 2.6, o que gera gradientes maiores e melhora a taxa de convergência da otimização.

Gradiente Descendente

O algoritmo usualmente empregado na obtenção dos parâmetros θ_i, tal que $J(\theta)$ seja minimizado, é o método do gradiente descendente (GD). Este se desloca iterativamente, dentro do espaço solução da função custo, na direção da descida mais íngreme, o ponto ótimo do modelo (LAMBLET VAZ, 2020), a figura 2.6 ilustra este comportamento:

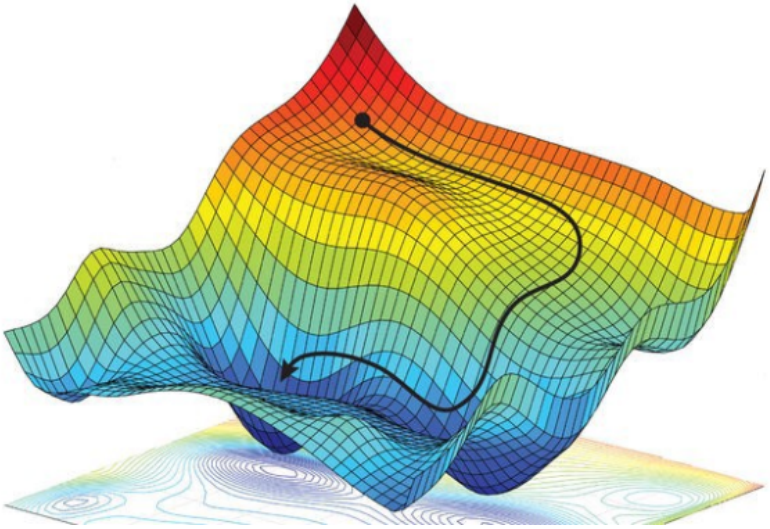


Figura 2.6: Deslocamento do gradiente no espaço solução da função custo. (FONTE: LAMBLET VAZ (2020))

Assim sendo, a implementação do método GD busca encontrar, através do gradiente da função custo, as regiões mais íngremes para se deslocar até o ponto mínimo do espaço solução do problema. Para apresentar o modo de operação do GD é exemplificado o problema de regressão que utiliza a função custo MSE:

- Obter o gradiente dos parâmetros;

$$\begin{aligned} \nabla_{\theta_0} J &= \frac{dJ(\theta_0,\theta_1)}{\theta_0} = \frac{1}{m}\sum_{m}^{i=1}(y_i - h_\theta(x_i)) \\ \nabla_{\theta_1} J &= \frac{dJ(\theta_0,\theta_1)}{\theta_1} = \frac{1}{m}\sum_{m}^{i=1}(y_i - h_\theta(x_i))x_i \end{aligned} \tag{2.8}$$

- Definir taxa de aprendizado (α) e atualizar os parâmetros:

$$\begin{aligned} \theta_0 &= \theta_0 - \alpha\nabla_{\theta_0} J \\ \theta_1 &= \theta_1 - \alpha\nabla_{\theta_1} J \end{aligned} \tag{2.9}$$

- Repetir as etapas acima até a função custo convergir, ao satisfazer a condição:

$$\nabla_{\theta_i} J = 0 \tag{2.10}$$

Posto que os coeficientes do modelo são atualizados por meio do gradiente, cabe ao projetista determinar a que passo acontece esse incremento. Como descrito na equação 2.9, a variável α que define este passo, sendo este termo denominado de taxa de aprendizado.

Gradiente Descendente Estocástico

O método do gradiente descendente estocástico (SGD) se apresenta como uma solução para o alto custo computacional do método gradiente descendente tradicional, visto que este utiliza toda a base de dados para o processamento. Já SGD realiza a otimização em um subconjunto da base de dados, escolhidos de forma randômica a cada iteração (BOTTOU, 2021).

Em contrapartida a vantagem de obter menor tempo de processamento durante a otimização, o método SGD apresenta *overshoot*, o que compromete a convergência da otimização. Para contornar esse problema é necessário um ajuste da taxa de aprendizado.

Gradiente Descendente com Momento

Em determinadas regiões do espaço solução da função custo, onde a taxa de variação em uma dimensão é muito maior que em outra, ou em inflexões da curva, o método GD apresenta ruído, de modo que o algoritmo não tende diretamente a região mais íngreme da função. Além disso, o SGD também é problemático por apresentar grandes oscilações. Assim sendo, o momento é introduzido como uma extensão aos métodos acima com intuito de aumentar a inércia do processo, acelerando a taxa de convergência (QIAN, 1999). Essa abordagem considera o passo de atualização dos parâmetros tomado em iterações anteriores (v_i). A relação abaixo descreve a técnica:

$$\begin{aligned} v_i &= \nabla_{\theta_i} J + \rho v_{i-1} \\ \theta_i &= \theta_{i-1} - v_i \end{aligned} \tag{2.11}$$

Tendo em vista a equação 2.11, as atualizações dos parâmetros ao longo do processo de otimização são acumulados e considerados nas próximas iterações. Sendo ρ o termo referente ao momento, o valor configurado a este termo depende do projetista. Para valores elevados de ρ, existe grande influência de iterações anteriores na atualização dos coeficientes, o que não acontece ao contrário.

Aprendizado Adaptativo

Os algoritmos AdaGrad, ou gradiente adaptativo, RMSProp, ou propagação do quadrado médio da raiz, são projetados para aplicação em problemas voltados para taxas de aprendizado adaptativo. Ambos os algoritmos tratam o problema de diferentes variações em dimensões distintas da base de dados, o que dificulta a escolha da taxa de aprendizado (DUCHI; HAZAN; SINGER, 2011).

Essas rotinas numéricas são muito similares, posto que ambas se propõem a determinar de maneira adaptativa a taxa de aprendizado. A principal diferença destas técnicas reside

no fato de que AdaGrad tende a diminuir em demasiado a taxa de aprendizado, o que faz com que os parâmetros deixem de atualizar e a otimização fique presa em uma região do espaço solução da função custo. A equação apresentada abaixo descreve o procedimento iterativo:

$$\theta_i = \theta_{i-1} - \frac{\alpha \nabla_{\theta_i} J}{\sqrt{\sum_{i=0}^{m} (\nabla_{\theta_i} J)^2}} \tag{2.12}$$

Como mostra a equação 2.12, a definição do passo realizado pelos parâmetros a cada iteração depende do histórico de atualização do processo. Se o gradiente acumulado for muito pequeno, isso significa que o espaço solução é bem comportado ao longo de suas dimensões, e o método permite passos longos. Entretanto, grandes variações são restringidas por pequenos passos, de modo que o algoritmo não tenda a uma região mal condicionada do espaço solução.

Como descrito anteriormente, o método AdaGrad tende realizar pequenos passos ao longo do tempo, o que tende a estagnar os parâmetros em uma região do espaço solução. Posto isso, é introduzido o método RMSProp, que contorna o problema da atenuação da taxa de aprendizado ao empregar o hiperparâmetro taxa de decaimento (γ) (BUSHAEV, 2018). O trecho abaixo exemplifica o método iterativo:

$$\begin{aligned} g &= \gamma g + (1-\gamma)(\nabla_{\theta_i} J)^2 \\ \theta_i &= \theta_{i-1} - \frac{\alpha \nabla_{\theta_i} J}{\sqrt{g}} \end{aligned} \tag{2.13}$$

Segundo a equação 2.13, o termo γ serve de contrapeso para o passo de atualização passado. Desse modo, cabe ao projetista definir a taxa de decaimento, de modo que o algoritmo dê passos mais longos em regiões que o AdaGrad realizaria passos curtos, e consequentemente estagna. O termo g é um auxiliar no armazenamento dos gradientes quadráticos acumulados.

Adam

O algoritmo Adam, ou estimador de momento adaptativo, consiste na união do SGD com o momento e os métodos de aprendizado adaptativo, RMSProp e AdaGrad. Esse método é bastante popular por apresentar flexibilidade de escolhas de projeto, sendo o projetista responsável por ponderar qual benefício proveniente das técnicas de otimização descritas acima favorecem.

O método Adam calcula a média móvel dos gradientes e os gradientes quadráticos acumulados, referentes ao momento e a aprendizagem adaptativa, respectivamente, sendo controladas as taxas de decaimento destes através dos hiperparâmetros $\beta 1$ e $\beta 2$. As

expressões exibidas na sequência deixam explícito com ocorre na rotina numérica, onde i contabiliza o número de iterações:

$$g = \left[\beta 1 g[0] + (1-\beta 1)\nabla_{\theta_i} J \quad \beta 2 g[1] + (1-\beta 1)(\nabla_{\theta_i} J)^2\right]; \tag{2.14}$$

$$vc = \left[\frac{g[0]}{(1-\beta 1(i-1))} \quad \frac{g[1]}{(1-\beta 2(i-1))}\right]; \tag{2.15}$$

$$\theta_i = \theta_{i-1} - \frac{\alpha vc[0]}{\sqrt{vc[1]}} \tag{2.16}$$

Entretanto, Adam apresenta o problema de começar as iterações com uma taxa de aprendizado muito alta, que pode comprometer a convergência do método. Posto isso, é introduzido o termo viés de correção (vc) aos hiperparâmetros $\beta 1$ e $\beta 2$, tal que seja evitado o mal condicionamento desses (RUDER, 2016).

2.2.4 Aprendizado de Máquina

Validação Cruzada

A validação cruzada é um método amplamente utilizado na modelagem estatística, de modo que diferentes modelos baseados em aprendizado de máquina possam ser comparados. O único parâmetro desse tipo de ferramenta é o k, que significa a quantidade de vezes que o dataset é subdividido (BROWNLEE, ????).

A principal vantagem que esse método introduz ao processo é a capacidade de generalização dos dados, ou seja, acurácia na predição de novos dados submetidos ao modelo. A metodologia desse tipo de procedimento segue os itens mostrados abaixo:

- Embaralhar o conjunto de dados aleatoriamente
- Dividir o dataset em k grupos
- Recursivamente utilizar um dos grupos com dados de teste e os outros como treinamento, avaliar o desempenho do modelo
- Escolher o modelo de melhor desempenho

A biblioteca *sklearn* auxilia o projetista na implementação da validação cruzada através da função *train_test_split*. Essa função tem como principais parâmetros os dados de entrada do problema e a definição do tamanho percentual dos dados de teste. Geralmente, de 70% a 80% das imagens são separadas para treinamento.

Aprendizado de Máquina Supervisionado

Problemas de aprendizado de máquinas podem ser categorizados em supervisionados e não-supervisionados. O que discerne tais categorias é o resultados dos dados observados, sendo que o primeiro tipo de solução sabe qual o resultado esperado de cada amostra do conjunto de dados, ou seja, os dados são rotulados, ou contrário do aprendizado de máquina não-supervisionado (ZHU; GOLDBERG, 2009).

Para o desenvolvimento do sistema de detecção de fadiga são empregados técnicas de modelagem baseadas na premissa de que os dados são rotulados. Esse tipo de abordagem foi escolhida para o projeto devido a disponibilidade de aquisição desse tipo de dataset e a ampla literatura desse campo de estudo.

2.2.5 Detecção Facial

A tarefa de detecção facial consiste em determinar a presença de algum rosto em um frame e retornar a localização deste. Multiplicidade de faces no mesmo frame, inclinação, expressão, cor da pele, óculos, pelos faciais, condições de iluminação e resolução da imagem são alguns exemplos da variabilidade de empecilhos enfrentados na resolução do problema (HEMALATHA; SUMATHI, 2014).

A extração de características de uma imagem é uma das formas mais populares de abordar o problema, sendo esta realizada através do mapeamento dos traços mais relevantes do objeto de interesse. O método histograma de gradientes orientados é uma das rotinas numéricas mais conhecidas na implementação desse tipo de solução.

Histograma de Gradientes Orientados

O método do histograma de gradientes orientados requer a priori um pré-processamento das imagens que compõem o banco de dados do problema e a configuração do tamanho de cada uma delas para $64x128$. As imagens reconfiguradas são subdivididas em células de tamanho $8x8$, e cada célula é submetida a convolução por meio de um filtro espacial diferencial (DALAL; TRIGGS, 2005), como o apresentado nas equações 2.3. Posto isso, são obtidas as magnitudes e a orientações de cada um dos gradientes, esta operação é executada através da transformação de coordenadas retangulares para polar, como mostra as expressões abaixo:

$$\begin{aligned} mag &= \sqrt{gx^2 + gy^2} \\ ang &= tan^{-}1\frac{gy}{gx} \end{aligned} \tag{2.17}$$

A equação 2.17 rearranja os dados resultantes em um histograma que varia em orientação de 0° até 180° com passo de 20°, e acumula os valores de magnitude referentes ao

intervalo de ângulos que abriga, de modo que se forme um vetor de 9 posições (MITTAL, 2020). Por fim, de maneira que a rotina seja eficiente e robusta a variações, o vetor de gradientes deve ser normalizado como aponta as equações 2.5. A figura 2.7 apresentada abaixo exemplifica o procedimento sobre a imagem 2.2:

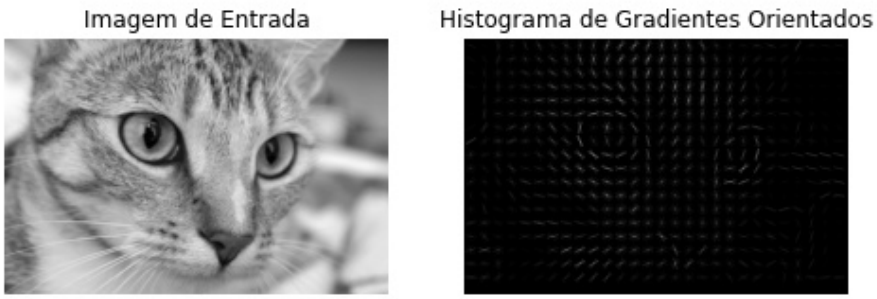


Figura 2.7: Histograma de gradientes orientados de uma imagem.

Como apontado anteriormente, este tipo de método é ideal para a detecção de objeto por ressaltar contornos, bordas e linhas. A figura acima, por exemplo, expõe essa característica, visto que regiões com contorno destacado, como olhos e nariz, são muito relevantes e apresentam maiores gradientes, como mostra o HOG.

Support Vector Machine

Support vector machine (SVM) é uma técnica de aprendizado de máquina supervisionado que visa a classificação e detecção de outliers. O objetivo desse algoritmo é determinar um hiperplano que discrimine os dados recolhidos.

SVM é um problema de otimização que visa separar diferentes classes de forma linear, de modo que a rotina numérica implementada encontre a solução com a maior margem de separação entre as amostras mais próximas do modelo (L. BRUNTON; KUTZ, 2017). A figura 2.8 abaixo apresenta duas soluções possíveis para a classificação de um conjunto de dados x:

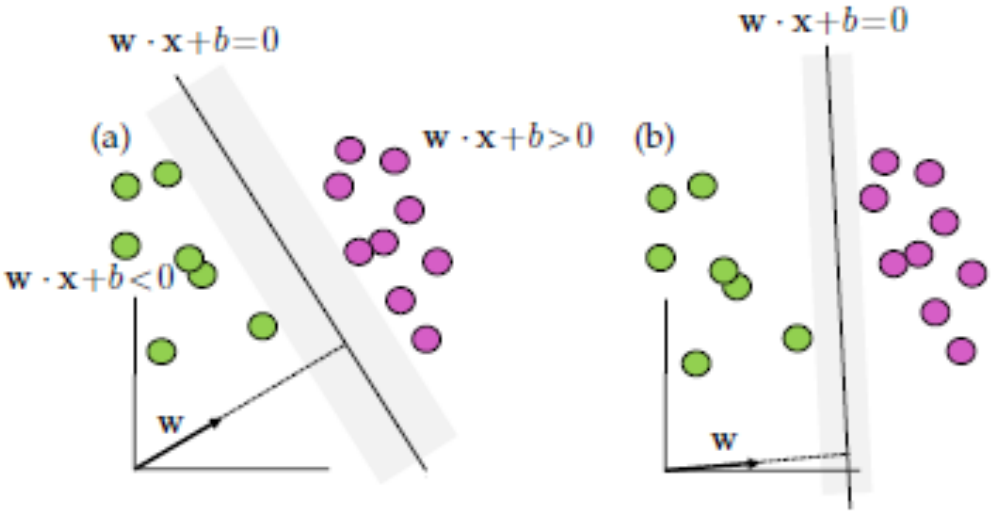


Figura 2.8: Soluções possíveis, sendo a imagem a direita produto a otimização, e consequentemente apresenta maior margem. (FONTE: L. BRUNTON; KUTZ (2017))

Como exposto na figura 2.8, os dois objetos de transformação desse tipo de otimização são a parametrização de um hiperplano que separe as classes e a maximização da margem. Entretanto essas condições são divergentes, de modo que uma prejudica o desempenho da outra, dessa maneira recai sobre o projetista balancear esses parâmetros para conciliar os objetivos. As equações abaixo apresentam as expressões do hiperplano e a função custo para a otimização, sendo y_j o rótulos verdadeiros para cada amostra e $\overline{y_j}$ a predição desses rótulos.

$$wx_j + b = 0 \tag{2.18}$$

$$l(y_j,\overline{y_j}) = l(y_j,sign(wx_j + b)) = \begin{cases} 0 & \text{se } y_j = sign(wx_j + b) \\ +1 & \text{se } y_j \neq sign(wx_j + b) \end{cases} \tag{2.19}$$

A função $sign()$ retorna o sinal resultante da operação realizada. Posto isso, a função custo é interpretada como o erro acumulado de classificações errôneas. Assim sendo, a otimização se propõe a treinar os dados e obter um modelo com o menor erro possível (L. BRUNTON; KUTZ, 2017).

Entretanto, o processo de otimização é condicionado pela obtenção dos gradientes da função custo, o que é inviável para a função $sign()$. Logo, para a minimização do erro é necessária a escolha de uma função aproximação para $sign()$ que seja diferenciável. A função *hinge* (H(z)) geralmente é empregado para a implementação computacional do problema (CHOWDHURY, 2019).

$$H(z) = max(0,1 - z) \tag{2.20}$$

Sendo considerada uma base de dados de tamanho m, a minimização do erro de classificação segue a expressão 2.21. Os dois termos que compõem a equação são o erro acumulado, $h(y_j,\overline{y_j})$, e a regularização, como proposto abaixo:

$$\begin{aligned} \underset{\text{w,b}}{\text{argmin}} \sum_{j=1}^{m} h(y_j,\overline{y_j}) + \frac{1}{2}||w||_2 \\ \text{restrição}: \ \min_{\text{j}}|wx_j| = 1 \end{aligned} \tag{2.21}$$

O termo de regularização é responsável por maximizar a margem. Quanto maior a margem, maior é a generalização do modelo, o que traz flexibilidade e adaptabilidade ao projeto quando sujeito a novas amostras.

Non Maximum Supression

Durante o processo de convolução para a detecção de imagens, diversas janelas de diferentes tamanhos são elencadas como potenciais áreas de detecção de face. Em consequência disso, é empregada a ferramenta *Non Maximum Supression* (NMS).

Esse método avalia o desempenho de cada uma das janelas de acordo com o modelo e a ocorrência de sobreposição de janelas. A métrica de sobreposição é chamada IoU (*Intersection over Union*), já o desempenho de cada janela é obtido através da predição do modelo.

$$IoU(Janela1,Janela2) = \frac{\text{Intersecção}(Janela1,Janela2)}{\text{União}(Janela1,Janela2)} \tag{2.22}$$

A equação 2.22 descreve IoU, essa métrica calcula a razão entre a área de intersecção entre duas janelas e a área de união entre as mesmas (PRAKASH, 2021). A ferramenta aponta quais janelas que se sobrepõem e escolhe a de maior desempenho para ser a janela de detecção (PRAKASH, 2021). A figura 2.9 descreve o processo:

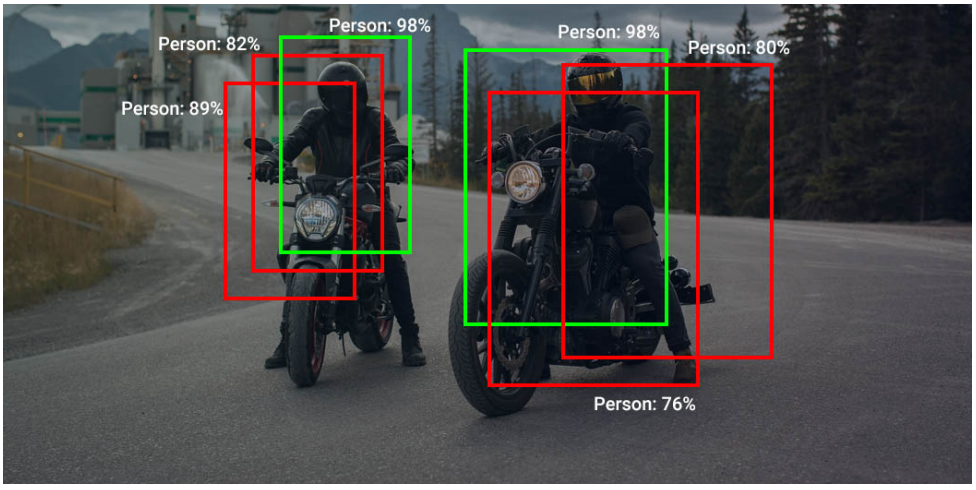


Figura 2.9: Método Non Maximum Suppression. (FONTE: PRAKASH (2021))

Como mostrado na figura 2.9 o método é funcional para classificação de múltiplos objetos. Além disso, previne redundância de informação e a presença de falsos positivos na detecção.

2.2.6 Redes Neurais Artificiais

O objetivo de implementação de redes neurais artificiais (ANN) é realizar o mapeamento de dados de entrada para classificação. Isto ocorre com o treinamento de um conjunto de dados x_j rotulados através de y_j (STANFORD, 2021). A estrutura de uma ANN, descrita na figura 2.10, é intercalada de várias camadas ponderadas por uma matriz de coeficientes A_j, de modo que sua arquitetura assume a forma abaixo:

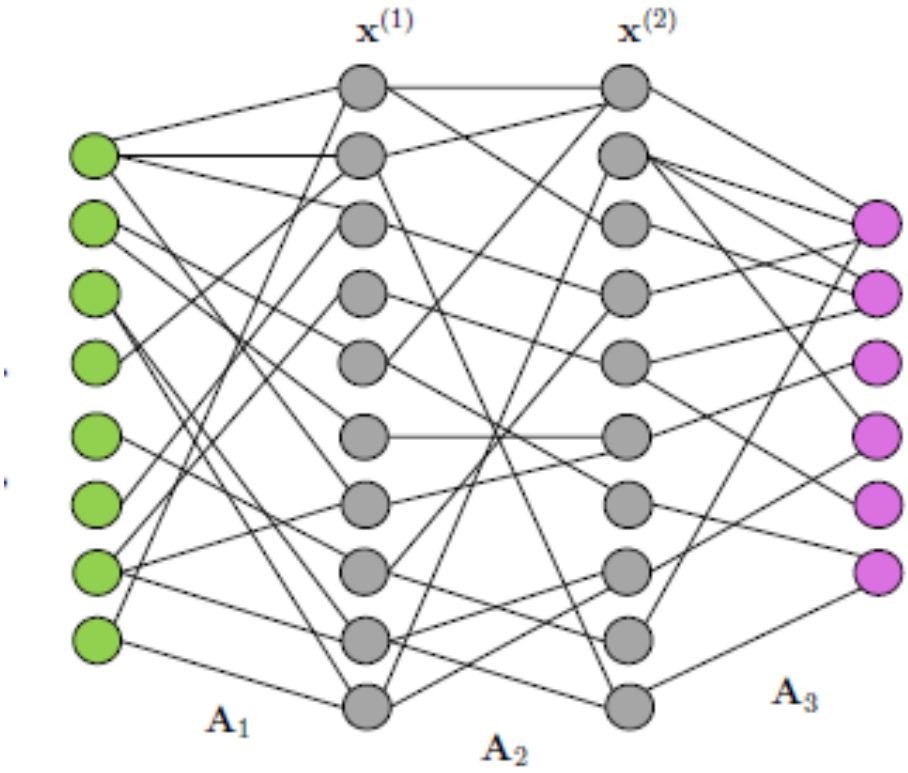


Figura 2.10: Arquitetura genérica de uma ANN. (FONTE: L. BRUNTON; KUTZ (2017))

Esse tipo de arquitetura é também chamado de rede neural completamente conectada. Cada nó da estrutura é conhecido como perceptron, sendo este uma unidade de operação linear. Deste modo a relação entre camadas subsequentes de uma ANN é descrita abaixo:

$$x_{j+1} = A_{j+1}x_j \tag{2.23}$$

A matriz A_{j+1} abriga os coeficientes sujeitos a otimização. Apesar da intuição linear para o problema, geralmente, as camadas intermediárias são preenchidas com funções de ativações que impõem não-linearidades ao modelo, e permite que esse obtenha uma

abrangência de respostas, o que traz flexibilidade para o projeto (STANFORD, 2021). A principal função de ativação utilizada é a Unidade Linear Retificada, ou ReLU.

$$f(x) = \begin{cases} 0 \ x \leq 0 \\ x \ x > 0 \end{cases} \tag{2.24}$$

A função ReLU é muito usada para aprendizado de máquina, já que possui baixo custo computacional e seu gradiente não é atenuado com o tempo, como acontece com outras funções ativação. Apesar disso, a arquitetura que utiliza a ReLU precisa de normalização, visto que esta não é centrada em zero.

BackPropagation

O procedimento de otimização em aprendizado de máquina visa a minimização do erro, sendo este calculado pela derivada da função custo. No caso de redes neurais, o método *BackPropagation* se aproveita da estrutura cascateada desta e utiliza o princípio da regra da cadeia para determinar o erro propagado ao longo das camadas (L. BRUNTON; KUTZ, 2017). A figura 2.11 mostrada abaixo ilustra o efeito em cascata desse tipo de modelagem:

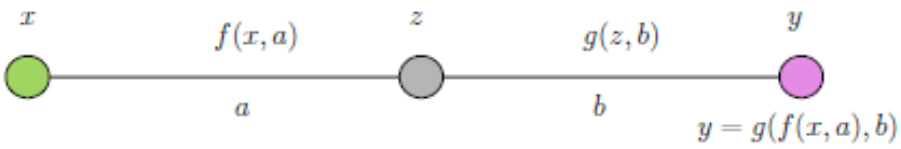


Figura 2.11: Rede neural com apenas uma camada intermediária. (FONTE: L. BRUNTON; KUTZ (2017))

BackProp alivia o custo computacional do cálculo do gradiente por envolver apenas operações simples como soma e multiplicação. Após a compilação do modelo, iniciado com coeficientes aleatórios, o método emprega a regra da cadeia de modo iterativo da última camada até primeira, tal que sejam ajustados os parâmetros do modelo. Para exemplificar o procedimento de otimização através do *BackProp* é utilizada a figura 2.11. Os termos yo e y representam, respectivamente, o rótulo correto de uma amostra e a aproximação definida pela ANN, de modo que a função custo seja:

$$E = \frac{1}{2}(yo - y)^2 \tag{2.25}$$

Assim como no GD, o método *BackProp* busca determinar, através do gradiente, o passo dado pelos parâmetros a cada atualização, sendo computado o gradiente da função custo em relação a cada parâmetro através da regra da cadeia. De modo que o processo seja facilmente interpretado, é suposto o problema de otimização de uma *ANN* com dois parâmetros, a e b, como expõe as equações na sequência:

$$\begin{aligned} \frac{dE}{da} &= -(yo - y)\frac{dy}{dz}\frac{dz}{da} \\ \frac{dE}{db} &= -(yo - y)\frac{dy}{db} \end{aligned} \tag{2.26}$$

Como mostra a equação 2.26, posterior a definição das funções de mapeamento entre as camadas da ANN, sendo estas diferenciáveis, o erro é encontrado por meio da sua propagação ao longo da topologia. Posto isso, a atualização dos parâmetros é similar a do GD, como mostra as expressões a seguir:

$$\begin{aligned} a_{k+1} &= a_k + \alpha\frac{dE}{da_k} \\ b_{k+1} &= b_k + \alpha\frac{dE}{db_k} \end{aligned} \tag{2.27}$$

A definição do passo aplicado aos parâmetros a cada iteração depende do projetista, sendo α o termo referente a taxa de aprendizado do modelo. Além disso, para não comprometer a eficiência e custo computacional do algoritmo é relevante a inicialização dos coeficientes, sendo, geralmente, escolhidos valores aleatórios (L. BRUNTON; KUTZ, 2017).

Deep Learning

Deep Learning é ramo de aprendizado de máquina que desenvolve modelos com estruturas profundas, ou seja, diversas camadas, e consequentemente elevado número de parâmetros. Um dos modelos mais usados em visão computacional é o de redes neurais convolucionais (CNN), que é interpretado com um extrator de características (L. BRUNTON; KUTZ, 2017).

A CNN pode ser dividida em duas partes, a primeira é responsável pela extração de características e a redução da dimensionalidade do problema, através da convolução e da subamostragem, ou pooling, respectivamente. E a segunda parte utiliza a saída da etapa de convolução para a estruturação de uma rede completamente conectada, como a ANN. A figura 2.12 apresentada abaixo ilustra a etapa de convolução:

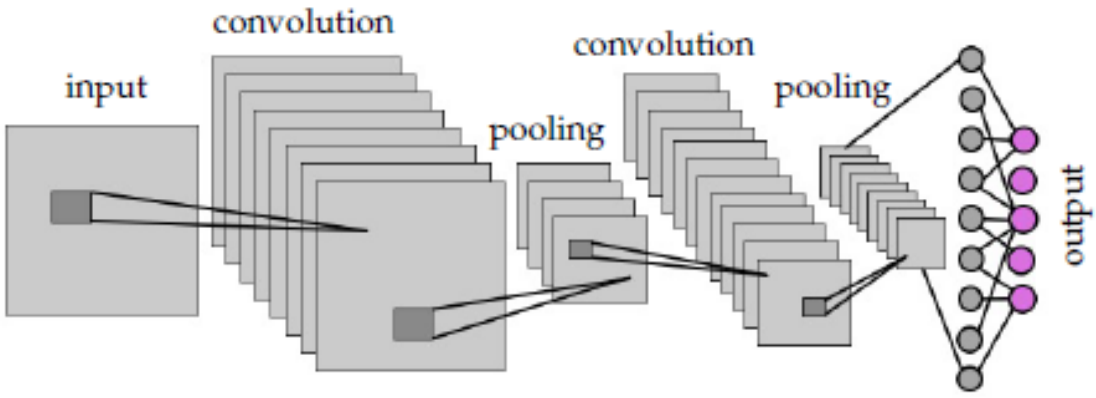


Figura 2.12: Arquitetura de rede neural convolucional. (FONTE: L. BRUNTON; KUTZ (2017))

A figura 2.12 realiza a operação de convolução e subamostragem, além de efetuar o mapeamento das características dos dados de entrada e sua redução dimensional. Vale ressaltar que os filtros empregados nesse mapeamento são os objetos de otimização do processo, de modo que são modificados a cada iteração, o que impõe complexidade ao sistema.

Posta a elevada dimensionalidade dessas redes neurais, o procedimento de convolução é acompanhado de camadas de subamostragem, sendo essa etapa denominada de *pooling*. A função empregada nessa tarefa, geralmente, é a *maxpooling*, que extrai o maior elemento de um bloco matricial $2x2$ (NETWORK, 2020).

CNN utilizam dois importantes hiperparâmetros no processo de convolução, *stride* e *padding*, definidos de acordo com escolhas de projeto. O primeiro se refere ao passo realizado na translação do filtro ao longo da imagem, na vertical e na horizontal. Já *padding* é o aumento da dimensionalidade da imagem, formando um frame de valor zero ao redor de toda a figura. A ausência de *padding* pode prejudicar a detecção ou classificação de objetos localizados nas extremidades do frame. Isto ocorre pois as extremidades são pouco influenciadas pela convolução comparadas a outras regiões (LEARNING, 2021).

Todas as camadas da rede neural são intercaladas por funções de ativação, como a ReLU já mostrada anteriormente. Outra função de ativação importante em problemas de aprendizado de máquina é a função *sigmoid*. Esta encapsula a última camada da rede neural dentro faixa de 1 e 0, de modo que esta solução se mostre conveniente de ser aplicada em problemas de classificação (STANFORD, 2021). A expressão da função *sigmoid* é descrita na sequência:

$$\sigma(x) = \frac{1}{1 + e^{-}x} \tag{2.28}$$

O emprego da equação 2.28 está relacionado ao fato dela ser diferenciável, ao contrário da uma função *step*. Entretanto, essa função apresenta a desvantagem de saturar, o que

pode prejudicar o processo de otimização, visto que o gradiente nessa região é muito pequeno, de modo que a atualização dos parâmetros fique estagnada. A figura 2.13 abaixo descreve o comportamento da função *sigmoid*:

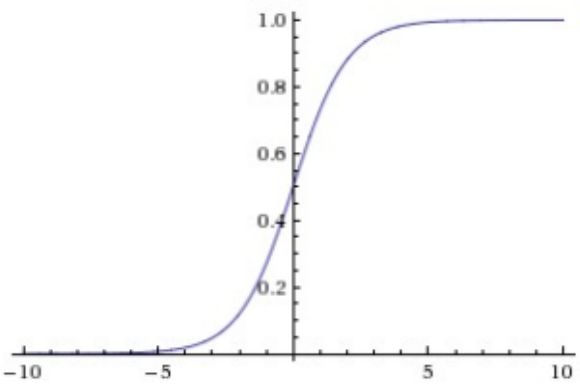


Figura 2.13: A função sigmoid encapsula números reais dentro de uma faixa de [0, 1]. (FONTE: STANFORD (2021))

Por fim, a estrutura de uma rede neural ainda pode ser composta de camadas intercaladas de regularização e normalização dos dados, denominadas *Dropout*. Essa técnica permite a generalização do aprendizado e previne que o modelo apenas memorize os dados, visto que isto compromete a predição de novas amostras. A figura 2.14 na sequência ilustra esse procedimento:

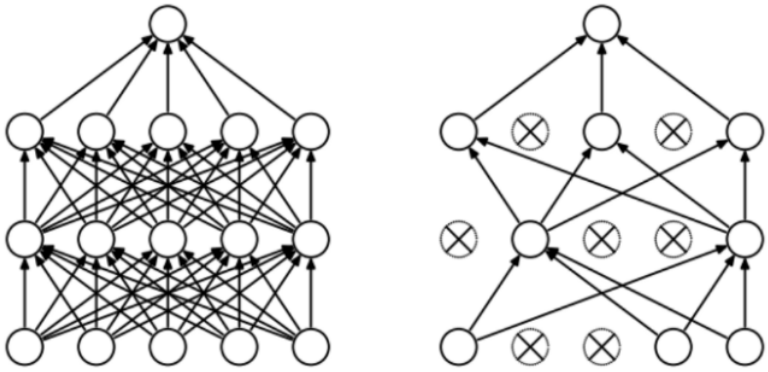


Figura 2.14: Camada de regularização dropout. (FONTE: BUDHIRAJA (2021))

Como mostra a figura 2.14, esse método consiste em remover de maneira aleatória determinados nós da rede neural (BUDHIRAJA, 2021). A finalidade do *Dropout* é restringir o número de parâmetros do modelo para que não ocorra sobreajuste *overfitting*. Sobreajuste é quando um modelo se ajusta perfeitamente a um conjunto de dados, mas não consegue aprender a partir de novas amostras.

2.2.7 Raspberry pi

O *Raspberry pi* é um dispositivo de baixo custo e de dimensões físicas que permitem a portabilidade do aparelho, 85mm x 56 mm. Além disso, o microcontrolador apresenta uma elevada capacidade computacional, emprega distribuição Linux como sistema operacional e tem *python* como linguagem nativa, o que o torna viável a aplicações relacionadas à visão computacional (FOUNDATION, 2022). A figura 2.15 mostra o microcontrolador:

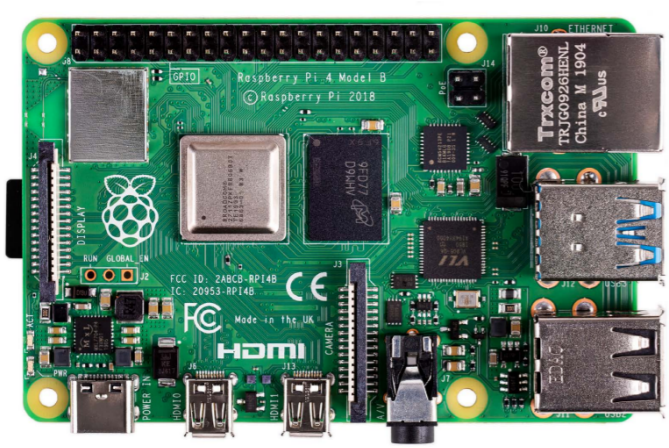


Figura 2.15: Raspberry pi. (FONTE: FOUNDATION (2022))

A placa mais recente disponível tem 1,5GHz de clock do processador e pelo menos 1GB de memória RAM. Além disso, o dispositivo apresenta adaptador Wifi (802.11ac), Bluetooth 5.0 BLE, 4 porta USB, conexão HDMI e Ethernet, slot para cartão de memória, 40 pinos GPIO, e interface para câmera (FOUNDATION, 2022).

Capítulo 3

Metodologia

Essa seção trata em detalhes da metodologia aplicada na elaboração do sistema proposto. As ferramentas aplicadas, as técnicas implementadas, e as decisões referentes às especificações de projeto tomadas ao longo do trabalho são as categorias abordadas neste tópico.

3.1 Escopo do Trabalho

Esse trabalho desenvolve uma solução embarcada que empregue uma algoritmo de aprendizado de máquina para detecção de fadiga em motoristas. A rotina numérica busca otimizar os parâmetros de um modelo que realiza o reconhecimento e classificação de olhos e face. Abaixo é apresentado o esquemático 3.1 para o projeto:

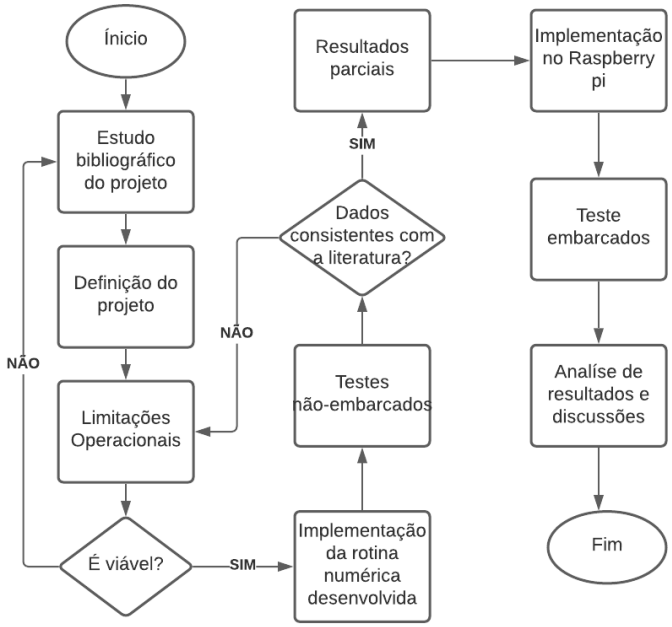


Figura 3.1: Fluxograma do trabalho.

A primeira etapa do desenvolvimento desse projeto foi a busca por referências bibliográficas para expor a viabilidade e atualidade da proposta. Desse modo, foram definidos os métodos para implementação, sendo considerados as limitações impostas pelo condicionamento do objeto de investigação.

Os principais empecilhos na implementação dos métodos são a iluminação, a ocultação e a inclinação do objeto. Posto que os métodos aplicados são baseados na extração de características faciais e oculares, a ocultação da face ou dos olhos se mostra um problema incontornável, visto que os algoritmos se fundamentam na inspeção espacial do objeto. Quanto a inclinação do objeto, é possível desprezar essa condição, já que a rotação da face, independente do eixo, não está no contexto do trabalho, visto que o condutor mantém o rosto alinhado com a via com pequenas inclinações que são admitidas pelo sistema, sem prejudicar a detecção e reconhecimento dos objetos de interesse. Por fim, a iluminação do ambiente pode prejudicar a detecção de fadiga, principalmente se o hardware que embarca a aplicação apresentar baixa taxa de amostragem e resolução ruim.

Os métodos escolhidos para o projeto foram: histograma de gradientes orientados (HOG), para a detecção facial; *landmark detection* para a detecção ocular; e redes neurais convolucionais (CNN), para o reconhecimento ocular. O primeiro foi preterido devido a simplicidade teórica, a robustez dos resultados e a velocidade de resposta, sendo conhecido por realizar predições em tempo real (MIZUNO *et al.*, 2012). Já os dois últimos métodos são empregados com o uso de redes neurais, com referenciais bibliográficos recentes e baixas taxas de erro (HAMID; RAZALI; IBRAHIM, 2019). Posto que todos os métodos escolhidos são fundamentados em aprendizado de máquina, cabe ao projetista definir uma base de dados com variabilidade de condições para o treinamento dos modelos. O algoritmo responsável pela detecção, em tempo real, de fadiga em condutores, é estruturado na forma sequencial mostrada no diagrama 3.2 a seguir:

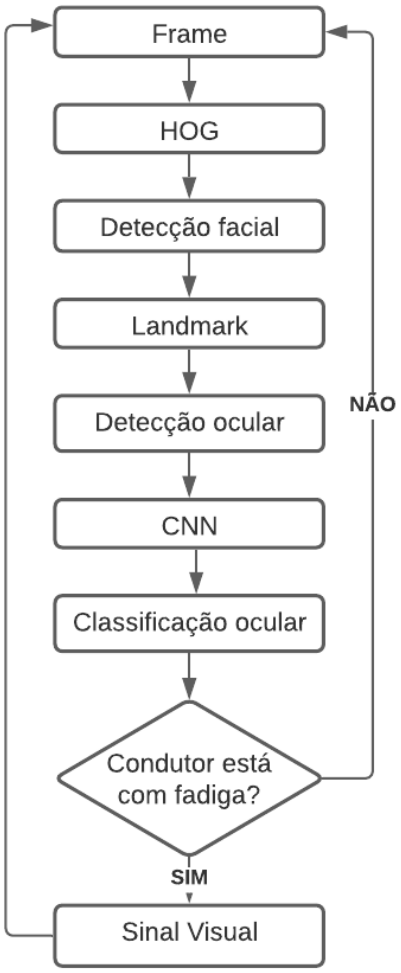


Figura 3.2: Diagrama de fluxo de dados

A figura 3.2 apresenta o diagrama de fluxo de dados do algoritmo por meio da metodologia empregada na sua elaboração. Os métodos expostos neste tópico manipulam os dados adquiridos por uma câmera de monitoramento para determinar a condição de direção do motorista. Os indicadores e métricas levantados, referentes ao desempenho das fases de treinamento e a velocidade de processamento da fase de teste, são relevantes para a viabilidade do projeto. O sistema apresenta como resposta sinal sonoro, e é iterativo, até ser desabilitado manualmente pelo condutor.

Por fim, após a realização dos primeiros testes sobre o software desenvolvido, a solução implementada é embarcada num dispositivo *Raspberry pi*. A análise de desempenho do projeto passa pela comparação entre os testes realizados durante a construção do algortimo e a solução embarcada, sendo consideradas as limitações inerentes ao projeto, provenientes das técnicas empregadas no software e das condições de operação do sistema.

3.2 Linguagem de programação utilizada

Esse tópico se dedica a explicar a escolha da linguagem de programação utilizada. Esse tipo de sistema apresenta em sua bibliografia aplicações em diversas linguagens, como *Java*, *C++* e *Python*.

Esse projeto foi implementado em *Python*, visto que este é amplamente empregado em tarefas de tratamento de imagens e aprendizado de máquina. Além disso, a linguagem é conhecida por ter sintaxe simples e concisa, e preza pela interpretabilidade do código.

Outro fator de influência é a disponibilidade da plataforma em código aberto e discussão das técnicas em comunidades como *Github* e *Stackoverflow*. Como ambiente de simulação, devido ao fato de que os modelos construídos requerem grande capacidade de processamento para o treinamento, o ambiente de simulação utilizado foi a plataforma *Jupyter Notebook*, de modo que o algoritmo possa ser compilado em uma GPU, que é uma unidade de processamento gráfico.

Depois de embarcado não existe a necessidade de utilização de GPU para o processamento do software. Isto se deve ao fato de que o tipo de modelo empregado no projeto, redes neurais, podem ser treinados offline. Vale lembrar, que o algoritmo é compilado através do sistema operacional do microcontrolador, uma distribuição linux.

3.3 Bibliotecas utilizadas

A definição das bibliotecas está diretamente associada à escolha da linguagem de programação, sendo consideradas bibliotecas voltadas para o tratamento de imagens e aprendizado de máquina. As principais bibliotecas empregadas no algoritmo foram: *numpy*, *pandas*, *matplotlib*, *sklearn*, *dlib*, *keras* e *OpenCV*.

A biblioteca *numpy*, ou *Numerical Python* é canônica na implementação de algoritmos de aprendizado de máquina. Esta é adotada na manipulação de matrizes e vetores, sendo relevante na análise algébrica de dados (SANTIAGO JR., 2018). Já *pandas* é dedicada a análise e visualização de dados, de modo que facilita a interpretação e o manuseio de grandes conjuntos de dados, visto que é recorrente a presença de dados redundantes e corrompidos. Posto isso, é fundamental a limpeza do banco de dados em tarefas de aprendizado de máquina para o desempenho e tempo de processamento do sistema (FIGUEIREDO, 2018). A biblioteca *matplotlib* é também utilizada na visualização de dados (SOLOMON, 2021).

Quanto ao tratamento de imagens, seja na filtragem espacial ou na utilização de *data augmentation* para tornar o banco de dados mais robusto, é utilizada a biblioteca *sklearn*. Esta é também responsável pela implementação da detecção facial, através de funções para obtenção de HOG, e a posterior classificação por meio de SVM (SCIKIT, 2021a,b).

O modelo definido é aplicado através da biblioteca *dlib*.

A biblioteca *keras* se dedica à construção de uma arquitetura para redes neurais, sendo consideradas diversos fatores, ou hiperparâmetros, como quantidade de camadas, de filtros na etapa de convolução, determinação do tipo de função custo de acordo com o problema abordado e do tipo de função ativação, definição da necessidade de utilização de camadas de normalização de dados, como dropout, e escolha da taxa de aprendizado para o processo de otimização.

De posse do modelo treinado, cabe a biblioteca *OpenCV* dar o suporte na área de visão computacional, por meio da aquisição de imagens provenientes da câmera de monitoramento e a manipulação dessa informação para a definição da janela que contém o objeto rastreado.

3.4 Datasets utilizados

Os datasets são os conjuntos de imagens empregados no treinamento dos algoritmos, e que influenciam o processo de aprendizado do modelo devido a seu tamanho ou características inerentes como mal condicionamento dimensional, ou pouca variabilidade das amostras recolhidas. Posto isso, é necessária uma análise discriminatória do conteúdo dos dados pelo projetista, de modo a se determinar a necessidade de um pré-processamento e como tratar essas informações.

Os datasets foram escolhidos por serem código aberto pela plataforma *kaggle* e embutidos a biblioteca *sklearn*. Os três datasets definidos são referentes às aplicações: HOG para detecção facial, *landmark* para detecção ocular, e CNN para a classificação ocular.

Os dados utilizados no treinamento do algoritmo de detecção facial é um módulo que faz parte da biblioteca *sklearn* e é denominado LFW, ou *the labeled faces in the wild* (SCIKIT, 2021c; LFW, 2021). O dataset contém 13323 amostras positivas, sendo estas figuras de rostos centralizados, e 13233 amostras negativas, sendo estas compostas por imagens de relógios, moedas, xícaras de café, etc. Todas as imagens que fazem parte desse conjunto de dados apresentam as mesmas dimensões de $62x47$. As figuras 3.3 abaixo mostram exemplos de amostras positivas e negativas, respectivamente, que compõem o dataset:

Figura 3.3: Amostras positivas do dataset LFW.

Como descrito na figura 3.3, o conjunto de imagens apresenta grande variabilidade de condições, como inclinação do rosto, geometria facial, rostos de diferentes gêneros e idades e diversas expressões como espanto, alegria. A escolha de um dataset com essa alteração visa a construção de um modelo robusto e insensível a *outliers*. A figura 3.4 dispõe de imagens de diversos objetos, e servem de contraste às características que o modelo busca aprender.

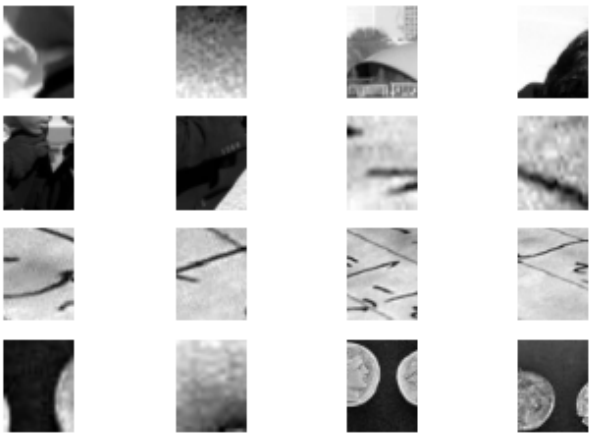


Figura 3.4: Amostras negativas do dataset LFW.

O banco de dados *LFW* é comumente empregado em aprendizado de máquina para a tarefas de detecção ou reconhecimento facial, além disso, foi escolhido devido a facilidade de implementação e por já ser o módulo de uma biblioteca utilizada no projeto.

Quanto ao dataset empregado na detecção ocular, retirado da plataforma *kaggle* (AMARANG, 2021), este é composto por 5847 imagens com 68 coordenadas demarcando a posição de características faciais como sobrancelhas, olhos, nariz, boca e maxilar. A figura 3.5 a seguir ilustra uma imagem que faz parte desse dataset:

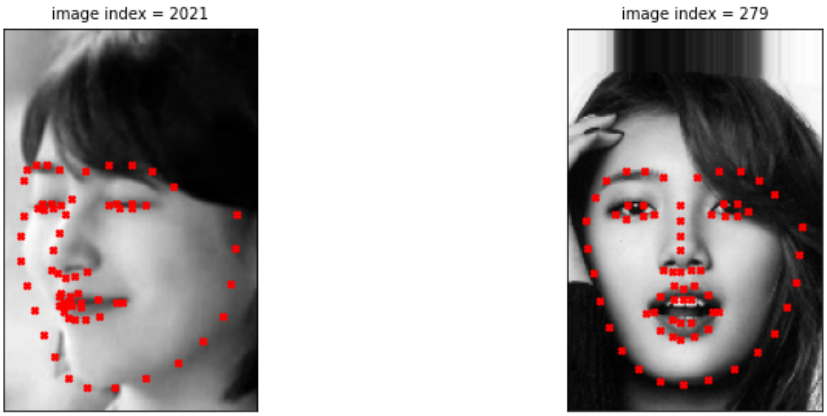


Figura 3.5: Landmark dataset.

Assim como o dataset escolhido para a detecção facial, o banco de imagens empregado na detecção ocular apresenta uma variabilidade de condições de apresentação das faces, como descrito pela figura 3.5. As informações provenientes desse banco de dados são: um vetor que armazena a localização dos 68 pontos, ou *landmark*; e as matrizes que armazenam as imagens.

Por fim, para a tarefa de classificação ocular, também obtido via plataforma *kaggle* (BABJAK, 2020), foi usado um dataset composto por amostras positivas e negativas, ou de olhos fechados e olhos abertos. que somadas contabilizam um total de 2874 figuras de dimensões $26x34$. A figura 3.6 na sequência exemplifica esse banco de dados com algumas imagens:

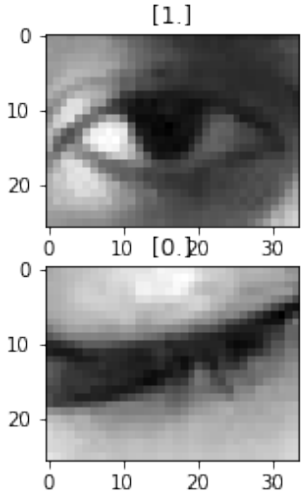


Figura 3.6: Dataset empregado na classificação dos olhos.

A figura 3.6 apresenta as duas classes que compõem o conjunto de dados, olhos abertos, rotulado com 1, e olhos fechados, rotulados com 0. Devido à pequena quantidade de amostras desse *dataset*, a técnica *data augmentation* é utilizada, de modo que a saída do método seja mais acurada e robusta.

3.5 Microcontrolador empregado

Uma das ferramentas mais escolhidas por projetistas da área de visão computacional é o microcontrolador *Raspberry pi*. Isto se deve a facilidade de implementação que o dispositivo oferece, sendo a solução embarcada em uma distribuição linux onde é possível construir uma rotina numérica sobre *Python*, linguagem de programação nativa da ferramenta.

Outro fator decisivo na escolha é a capacidade computacional deste controlador, o que garante robustez, velocidade e veracidade a resposta de código desenvolvidos a partir de tecnologias de aprendizado de máquina.

Posta às demandas técnicas requeridas de tal dispositivo, o modelo do controlador escolhido também foi baseado na disponibilidade de mercado. As especificações do produto definição são apresentadas na sequência:

Dispositivo	Raspberry pi 3 B
Clock do Processador	1.2GHz
RAM	1GB
GPIO	40
Wi-fi	Sim
Bluetooth	Sim
HDMI	1
USB	4
Ethernet	1
Armazenamento	MicroSD
Preço	R$550

Tabela 3.1: Tabela de especificações *Raspberry pi*

O sistema operacional do controlador é chamado de *Raspian*, este é similar ao *Debian*, uma distribuição linux. O cartão de memória de 16GB é utilizado, mais que necessário, para o armazenamento de 2,4GB proveniente dos arquivos associados ao algoritmo elaborado. Além disso, são empregadas duas portas GPIO para a sinalização do sistema.

3.6 Câmera empregada

A aquisição de dados do sistema depende da escolha de uma câmera que seja robusta nas adversidades condicionais inerentes ao trabalho. O principal vetor de variabilidade de resultados do projeto é a iluminação do ambiente, de modo que a resolução da câmera influencia a veracidade da resposta elaborada para o problema. Posto isso, foi definida para o projeto a implementação da câmera descrita na figura 3.7:

Figura 3.7: Câmera para aquisição de imagens.

Além disso, é preciso que o dispositivo escolhido tenha suporte do sistema operacional do *Raspberry pi*. Dentre outras especificações deste temos: faixa de foco de 200cm, conexão USB 2.0, e resolução de 1920x1080. O valor desse tipo de sensor é de cerca de R$100.

Capítulo 4

Desenvolvimento

O desenvolvimento do projeto é dividido nas etapas de treinamento e implementação. Primeiramente, são treinados os modelos de detecção facial e ocular e de classificação ocular, sendo utilizados os datasets referentes a cada tarefa.

Visto que todos os modelos elaborados admitem um determinado tipo de dado de entrada, as informações provenientes dos datasets são manipuladas numa etapa de processamento de imagens, de modo a conformar as informações digitalizadas das imagens em um determinado padrão. Os tópicos abaixo explicam em detalhes o treinamento de cada uma dos algoritmos que compõem o trabalho, e por fim, os modelos encontrados são concatenados para atender o objetivo do projeto de usar os olhos do condutor como indicador de fadiga.

A etapa de implementação visa embarcar a solução em um controlador, onde um sistema de alerta é elaborado para a sinalização do estado do condutor. Led's, resistores, jumpers e protoboard são empregados na construção deste.

4.1 Treinamento HOG

O treinamento do modelo de detecção facial se dá por meio do método do histograma de gradiente orientados (HOG). O banco de imagens empregado na tarefa é o LFW, presente na biblioteca *sklearn*, um conjunto de imagens, com amostras positivas e negativas, que são rotuladas, ou seja, cada imagem é associada a um valor, 0 ou 1, que aponta se ela contém uma face ou não. O HOG é recolhido de cada imagem no banco de dados, sendo essas informações usadas para realizar a classificação. Desse modo, o algoritmo busca aprender qual a classificação de cada imagem através do seu rótulo, o classificador SVM elabora um modelo linear e visa através de otimização obter os parâmetros que melhor generalizem o conjunto de dados.

Posto isso, cabe ao projetista determinar o hiperparâmetro de regularização C que multiplica a função custo do classificador SVM. Esse hiperparâmetro força o modelo,

um hiperplano, a diminuir a margem em favor de melhorar o desempenho na classificação das amostras (SAVAL, 2017; LUO, 2018). O objetivo de otimização desse método é obter a maior margem e o maior número de classificações corretas possíveis, de modo que a escolha do hiperparâmetro de regularização penalize o desempenho da detecção facial.

Assim sendo, é desenvolvida uma rotina numérica, com auxílio da biblioteca *sklearn*, para testar uma faixa de valores de C, de modo que seja determinado o hiperparâmetro de melhor desempenho para o modelo. O apêndice **A.1** descreve o trecho de código responsável por essa busca.

As duas primeiras linhas se referem a importação de bibliotecas e funções para a criação do classificador linear SVM e a aplicação de uma faixa de valores para o hiperparâmetro C, através da ferramenta $GridSearchCV$. Posteriormente tais funções são definidas e treinadas, sendo X_train um pedaço do banco de dados LFW designado para o treinamento do modelo, e y_train o rótulo das amostra referentes a este pedaço do dataset. Por fim, o trecho do algoritmo retorna o parâmetro C com melhor avaliação.

4.2 Treinamento CNN para Detecção e Reconhecimento Ocular

Os modelos de detecção e classificação ocular são construídos baseados na implementação de redes neurais convolucionais. Ambos os modelos são de aprendizado de máquina supervisionado, e os conjuntos de dados são divididos para que um pedaço seja utilizado no treinamento e o restante possa ser introduzido como imagens teste para averiguar o desempenho do modelo.

Outro ponto em comum dessas duas soluções construídas é o método de otimização *adam*, que tem um custo computacional reduzido e sem comprometimento do desempenho, já que utiliza apenas subconjuntos aleatórios do dataset para atualizar dos parâmetros, é implementado em ambos os modelos. Como escolha de projeto, este subconjunto é definido com 512 amostras.

Os hiperparâmetros referentes ao método de otimização *adam* são padronizados pela biblioteca *keras*, e não houve necessidade de alterações. O hiperparâmetro taxa de aprendizado (α) é definido como 0,001, enquanto as taxas de decaimento $\beta 1$ e $\beta 2$ assumem os valores de 0,9 e 0,99, respectivamente.

Além disso, para que as extremidades da imagem não sejam negligenciadas é utilizado o conceito de *padding* que introduz uma camada de zeros ao redor da imagem. Ambas as redes neurais também apresentam entre suas camadas, o método de regularização e normalização *Dropout*.

A estrutura genérica dessas redes neurais são compostas de três camadas referentes

a convolução, onde são usados filtros para o mapeamento de características das imagens, sendo a dimensão desses de 32x64x128. A etapa de convolução é permeada por camadas de sub amostragem os dados, o que visa aliviar o custo computacional e acelerar a taxa de convergência do algoritmo. Subsequente a etapa de convolução tem-se duas últimas camadas referentes aos neurônios completamente conectados.

Visto que redes neurais apresentam *BackPropagation*, ou seja, o erro se propaga ao longo das camadas, o processo de otimização tira vantagem dessa característica estrutural para atualizar os parâmetros. Entretanto, cabe ao projetista definir em quantos ciclos o erro se propaga, como uma condição de encerramento do procedimento de otimização, essa quantidade de ciclos é denominada como época. Usualmente, o projetista simula a arquitetura concebida e define poucas épocas para analisar desempenho, erro e convergência do modelo, caso a tendência apresentada seja de piora dessa métricas, o projetista pode alterar a CNN, caso seja de melhora consistente ele pode aumentar as épocas do treinamento até que sejam atingidas métricas satisfatórias.

O dataset utilizado para o primeiro problema é discutido no capítulo anterior (AMARANG, 2021), e é dividido em dois conjuntos, o primeiro composto por uma variedade de fotos com rosto centralizado, e o último é o vetor com as 68 coordenadas que informam a posição de traços faciais relevantes em uma pessoa, que funciona como um rótulo para as imagens e é o objeto de otimização do procedimento. O trecho do algoritmo que mostra a construção da arquitetura da CNN de detecção ocular é apresentada no apêndice **A.2**:

O trecho do algoritmo acima é referente a construção estrutural da rede neural. As primeiras linhas do código se dispõe a importar bibliotecas e funções empregadas na elaboração dos métodos necessários para a implementação do modelo.

Posteriormente, ao longo da estrutura da rede neural são determinados parâmetros específicos que diferenciam os problemas de detecção e reconhecimento, como a função custo e a ativação, visto que as tarefas propostas são, respectivamente, de regressão e classificação. A rotina numérica de detecção é submetida a um processo de otimização particular, onde o objetivo é determinar qual o valor ótimo para todas as 68 coordenadas que definem um rosto, logo temos um problema de regressão com a função custo MSE. A função ativação escolhida nessa regressão seguiu os referenciais teóricos recentes e empregou a $ReLU$.

Para o modelo de reconhecimento ocular, que também utiliza redes neurais convolucionais, temos o dataset apresentado na seção anterior (BABJAK, 2020). Posto que são apenas 2874 imagens neste conjunto de imagens, é empregada a técnica de *data augmentation* para induzir maior variabilidade de condições as amostras, o que visa a melhor generalização dos dados. Estes são submetidos às seguintes tranformações: normalização de intensidade, rotação em 10 radianos, translação horizontal e vertical, e distorção. O

algoritmo empregado para implementação do método é mostrado no apêndice **A.3**.

A primeira função aplicada, *ImageDataGenerator*, cria uma instância para a determinação das manipulações às quais o conjunto de dados é sujeitado. Desse modo, a função *flow* se encarrega da aplicação das operações de transformação. Assim sendo, a rede neural de classificação ocular é estruturalmente similar a apresentada anteriormente, sendo esta descrita no trecho de algoritmo apresentada no apêndice **A.4**:

A função custo e ativação são definidas para serem empregadas em um problema de classificação, de modo que o modelo faça a predição se o olho está fechado, 0, ou aberto, 1. Desta maneira, são empregadas as funções entropia cruzada, como objetivo de otimização, e a função sigmoid, para ativação.

4.3 Implementação do Algoritmo de Detecção de Fadiga

O sistema de detecção de fadiga em condutores é produto da concatenação dos três algoritmos apresentados anteriormente: detecção facial, detecção e reconhecimento ocular. Através do frame capturado essas operações são efetuadas.

Primeiramente, a face é encontrada ao deslizar-se uma janela de tamanhos distintos ao longo de todo o frame, o resultado desse procedimento é utilizado na sequência para se detectar os olhos. A detecção dos olhos ocorre através do método landmark, que aponta os traços mais relevantes do rosto através de pontos que determinam a localização desses na geometria facial.

Por fim, de posse da localização dos 12 pontos que definem a localização dos olhos, estes são usados para recortar a região dos olhos, de modo que esta seja introduzida ao modelo de classificação ocular. Os dados adquiridos pela câmera são normalizados antes de submetidos ao modelo.

4.4 Sistema Embarcado

O controlador é inicializado através da instalação do sistema operacional *Raspian*, uma distribuição linux. Este hardware é empregado para abrigar o software desenvolvido, e sua barra de entradas e saída, denominada GPIO, é configurada para implementação do sistema de alerta. Este é composto por uma protoboard, led's de sinalização, vermelho para condutor fadigado e verde para o contrário, resistores de 100 Ωe jumpers macho-fêmea para realizar a ligações entre a placa de contatos e o controlador (GAY, 2018). O modelo do *Raspberry pi* apresenta 40 pinos, a disposição desses na placa é descrita na sequência:

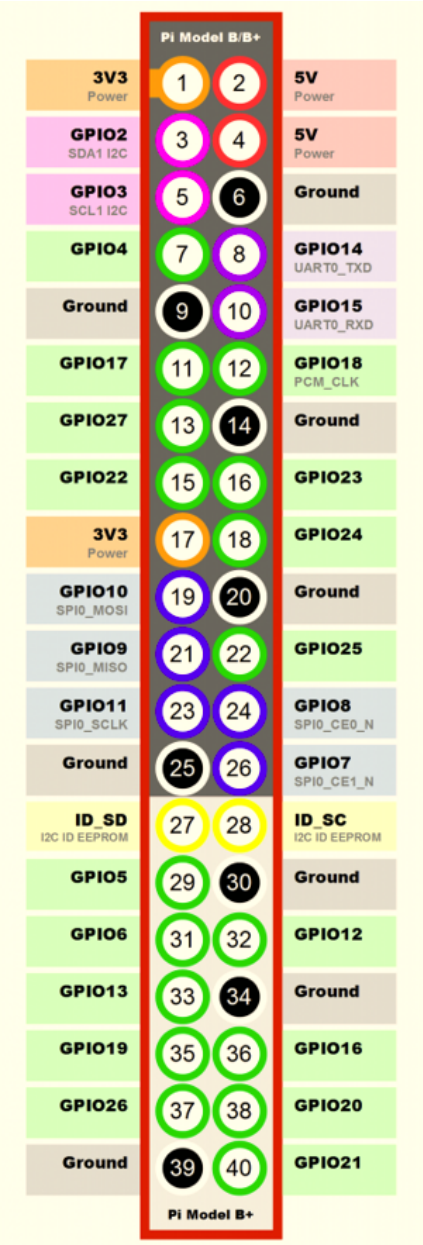


Figura 4.1: Pinagem do *Raspberry pi 3 B. (FONTE: GAY (2018))*

A configuração dos pinos do controlador ocorre através da instalação do pacote *rpi.gpio*, que auxilia na definição do pino, entrada ou saída, e nível de sinal. O trecho de algoritmo do apêndice **A.5** descreve a configuração como pinos de saída para os led's verde e vermelho e o loop de determinação de sinal destes.

Os pinos GPIO13 e GPIO19 são usados para acionar a sinalização, enquanto GND é o pino terra. O led vermelho é acionado apenas se ambos os olhos do condutor permanecerem fechados por mais de 4 segundos, caso contrário o led verde é mantido constantemente em nível de sinal alto. A imagem 4.2 mostrada abaixo expõe a construção física do sistema de alerta:

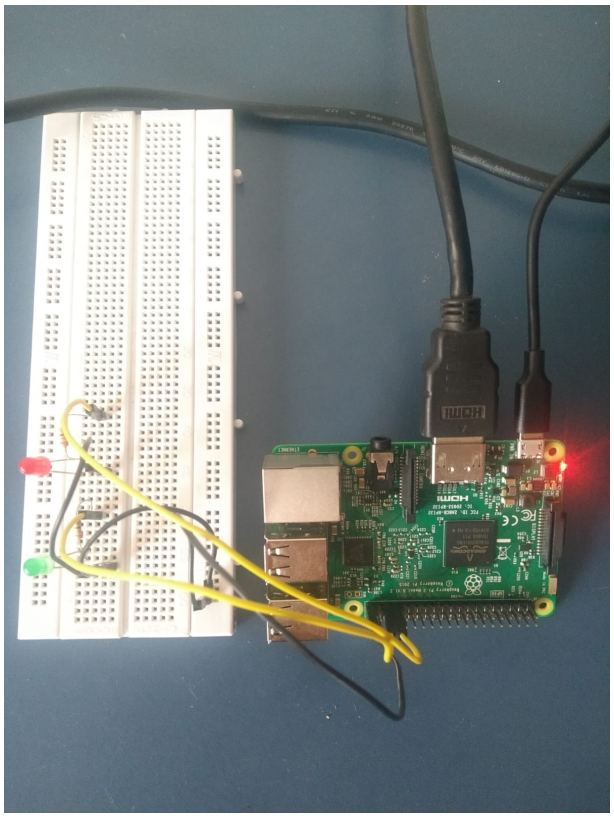


Figura 4.2: Sistema de Alerta

No capítulo posterior, são apresentados os testes para o software construído e a solução embarcada implementada. Métricas para avaliação de cada um dos modelos são elencadas, e são comparados os resultados entre as etapas.

Capítulo 5

Resultados e Discussões

5.1 Treinamento

O treinamento de um modelo para problemas de aprendizado de máquina possui a vantagem de ser realizado offline. Deste modo, o tempo de processamento para a realização dos testes não é comprometido pelo processo de treinamento.

Posto isso, uma das principais métricas elencadas para a avaliação do modelo treinado é a sua acurácia, responsável pela avaliação geral do desempenho do modelo. A acurácia quantifica a quantidade de classificações corretas do modelo:

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Total de predições realizadas}} = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

Os testes realizados são avaliados de acordo com os erros e acertos do modelo em rotular amostras positivas e negativas:

- Verdadeiros Positivos (VP): Classificações corretas das amostras positivas;
- Falsos Negativos (FN): Erro de rotulação do modelo na predição de classe negativa quando o valor real é de classe positiva;
- Falsos Positivos (FP): Erro de rotulação do modelo na predição de classe positiva quando o valor real é de classe negativa; e
- Verdadeiros Negativos (VN): Classificações corretas das amostras negativas.

A avaliação da modelagem da detecção facial através do método HOG ocorre durante a procura pelo melhor hiperparâmetro de regularização C. Isto porque o hiperparâmetro é definido através de uma busca otimizada , desse modo o melhor desempenho encontrado para um faixa de valores de C é de 98,9225%.

Já a avaliação de desempenho dos modelos de detecção e reconhecimento ocular, ambos desenvolvidos a partir de CNN, é feita por meio da evolução gradativa dos parâmetros de

perda e acurácia ao longo das épocas de treinamento. Posto isso, a tabela abaixo mostra os valores dessas métricas ao longo do treinamento:

	Detecção		Reconhecimento	
Época	Perda	Precisão	Perda	Precisão
1	6170,6245	0,00%	0,2157	90,97%
5	628,1575	6,38%	0,0311	99,31%
10	723,7112	86,27%	0,0067	99,65%
20	705,9379	86,27%	0,0086	99,65%
30	678,0162	86,27%	0,0112	99,65%
40	684,5080	86,27%	0,0096	99,65%
50	694,1927	86,27%	0,0116	99,65%

Tabela 5.1: Desenvolvimento das métricas perda e precisão ao longo do treinamento

Como mostra a tabela 5.1, a métrica perda, que é relacionada ao erro acumulado pela função custo, apresenta valores muito distintos nos dois métodos. Isto é devido a utilização de funções de custo distintas para cada uma das aplicações. Para o método de detecção ocular, por exemplo, é empregada a função custo MSE, de modo que a sua saída é o erro quadrático.

Quanto à taxa de convergência, foi simulado o treinamento para um grande número de épocas, de maneira que fosse analisado o comportamento do modelo a longo prazo. Posto isso, é observado que após 10 épocas, período suficiente para o método atingir seu ponto ótimo, os parâmetros estagnam em uma região do espaço solução do problema. Já para a métrica precisão dos modelos, foram atingidos valores satisfatórios, sendo 86,27% e 99,65% para detecção e reconhecimento ocular, respectivamente.

5.2 Testes Não-Embarcados

Os testes foram realizados ao longo do desenvolvimento do algoritmo. Desta maneira, foram primeiramente realizados testes para os métodos HOG e *landmark*, referentes a detecção facial e ocular, respectivamente.

Todavia, diferente do algoritmo de detecção de fadiga que utiliza os dados em vídeo provenientes do sensor, os testes realizados nessa primeira fase de desenvolvimento do software foram aplicados a uma imagem. A figura 5.1 a seguir apresenta teste realizado para a detecção facial:

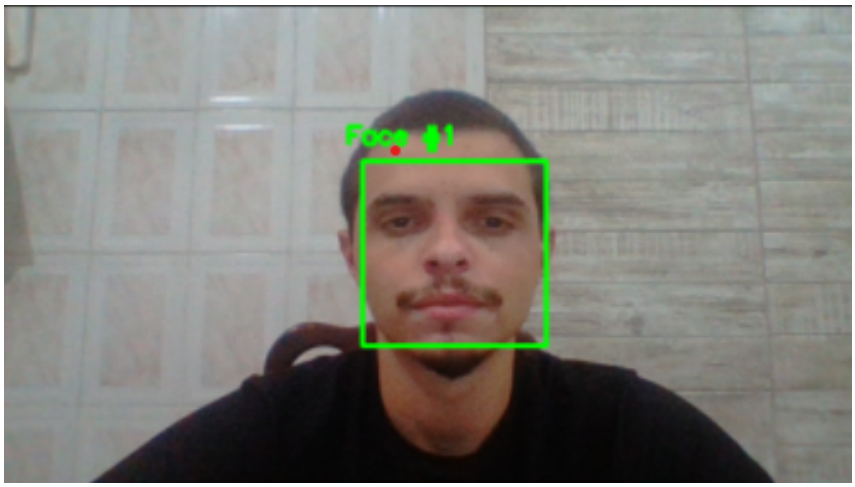


Figura 5.1: Teste para detecção facial

Como descrito pela figura 5.1, a janela de detecção obtida engloba a face disposta no *frame*, de tal modo que esse resultado possa ser empregado sequencialmente como entrada do método de detecção ocular. Sendo assim, a figura 5.2 abaixo mostra o teste do algoritmo *landmark*:

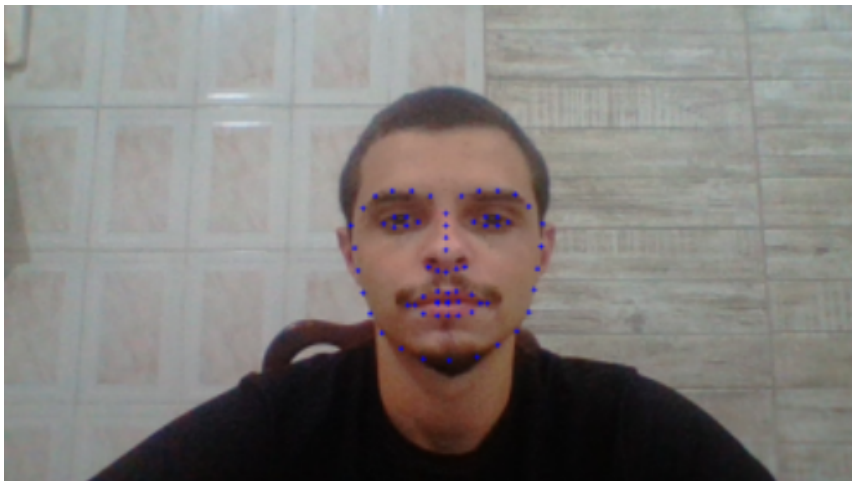
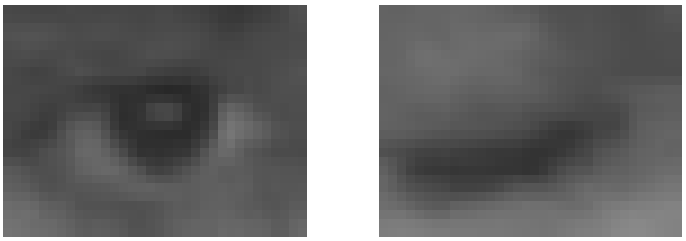


Figura 5.2: Teste para detecção ocular

Para a figura 5.2, o método constrói o mapeamento da geometria de uma face, sendo dispostas 68 coordenadas ao longo do rosto de um indivíduo. Essa técnica permite a identificação de traços relevantes na feição humana, entretanto o trabalho se interessa, particularmente, pela localização dos olhos, de modo que estes possam ser recortados da imagem original, e posteriormente analisados como indicadores de fadiga.

Por conseguinte, após o treinamento do modelo de classificação ocular, e a concatenação dos métodos, o software de detecção de fadiga é testado. Segue na sequência a figura 5.3 com os resultados obtidos do procedimento:



(a) olho aberto (b) olho fechado

Figura 5.3: Recorte do *frame* para olho aberto (a) e fechado (b)

A figura 5.3 é o recorte realizado sobre as coordenadas de detecção ocular após o processamento imposto pelos métodos *HOG* e *landmark*. De posse da captura de imagem dos olhos, estes são submetidos ao modelo *CNN* de reconhecimento ocular. O produto final desse sistema é exposto através da figura 5.4 a seguir:

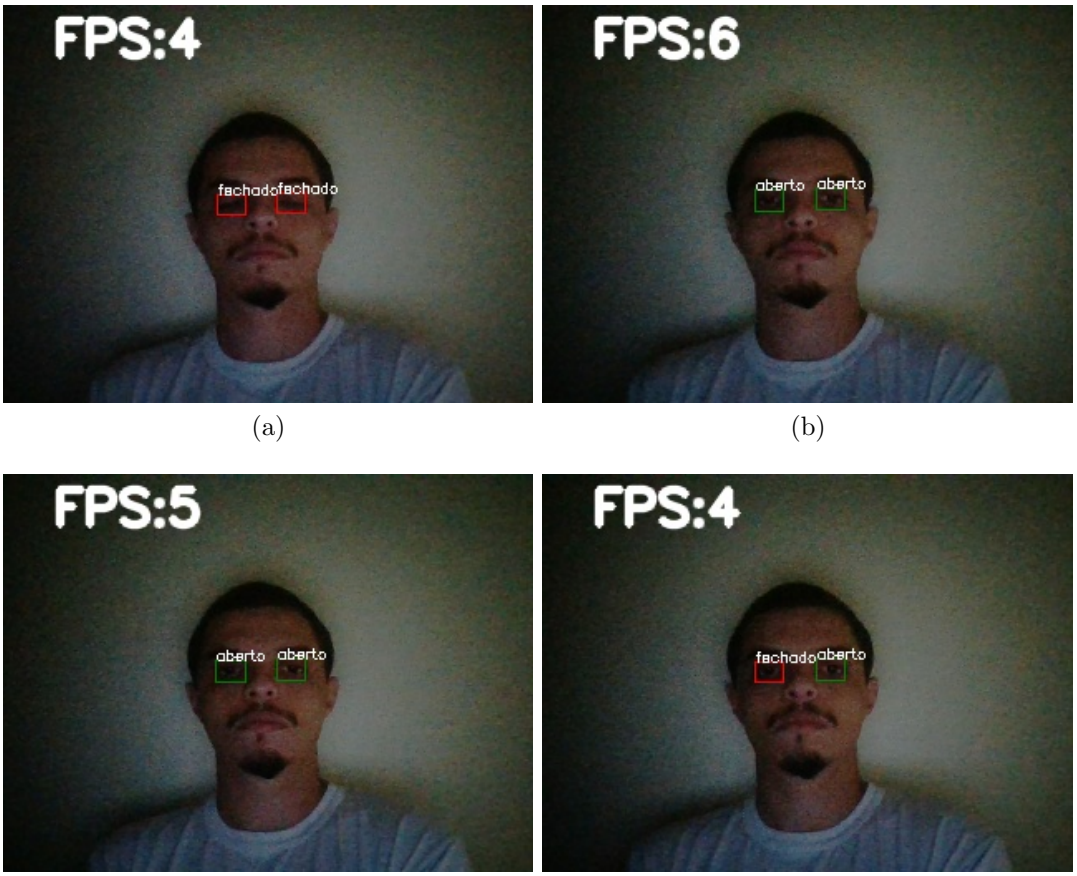
(a) olho aberto 1 (b) olho fechado

Figura 5.4: *frame* que indica se os olhos estão aberto (a) ou fechados (b)

A principal métrica de avaliação do processo é a velocidade de processamento, isso por que o sistema busca ter aplicabilidade em tempo real. Desse modo, essa aferição acontece através da definição de quantos *frames* por segundo, *FPS*, são processados pelo software. Visto que uma pessoa pisca em média 20 vezes por minuto (ROCHA, 2018), é necessário que o software desenvolvido atue pelo menos duas vezes mais rápido que esse valor, de modo que a informação recolhida não seja distorcida.

O desempenho do código é consideravelmente afetado devido ao tamanho dos modelos implementados. Entretanto, o sistema atua com 5 *FPS*, mais que o suficiente para captura de forma acurada da variação do estado dos olhos.

Além disso, um dos principais empecilhos para a implementação desse trabalho é a condições de iluminação do ambiente em que o sistema está inserido. Posto isso, foram realizados testes sob condições de baixa iluminação, tal que foram obtidas as figuras:



(a) (b)

(c) (d)

Figura 5.5: Recorte do *frame* para olho aberto (b), (c) e (d) e fechado (a)

Como mostrado pela figura 5.5, o algoritmo apresenta boa operabilidade em ambientes pouco iluminados. Ainda assim, o algoritmo se mostra momentaneamente instável sob condições de iluminação adversas, como descrito pela figura (d). Diversos fatores podem influenciar esse tipo de comportamento como: a baixa resolução da câmera de testes não-embarcados e o valor relativamente baixo de FPS.

5.3 Testes Embarcados

A realização dos testes da solução embarcada visa a análise de desempenho do trabalho sujeito a diversas condições adversas, como inclinação e ocultação da face, variação da distância do rosto para a câmera, e iluminação do ambiente. Os principais parâmetros de avaliação de desempenho do projeto são a detecção e reconhecimento do estado dos olhos e a taxa de processamento de imagens, indicada pelo FPS.

Em condições ideais de operação, onde o rosto do condutor é alinhado com a câmera e a distância entre estes é de no máximo 60 cm, o software embarcado se mostra robusto e aponta corretamente o estado dos olhos. As figuras 5.6 e 5.7 descritas abaixo exemplificam estes testes:

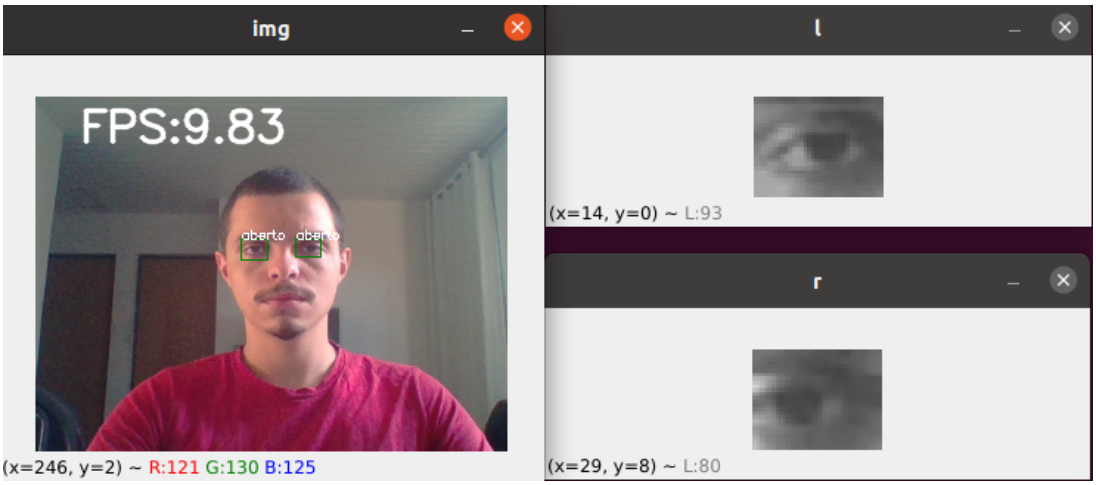


Figura 5.6: Teste embarcado em condições ideias com olhos abertos

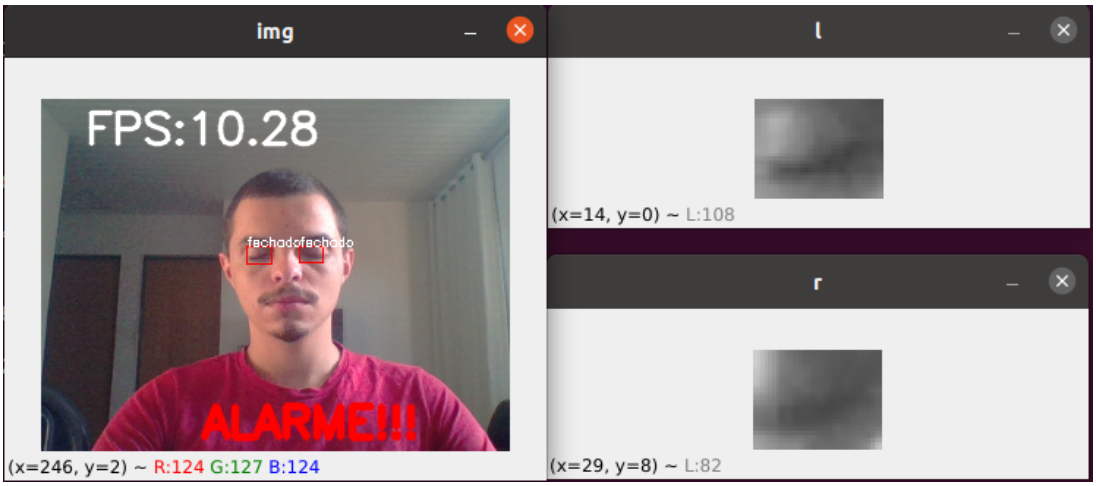


Figura 5.7: Teste embarcado em condições ideias com olhos fechados

O FPS do sistema embarcado varia na faixa de 9,5 até 11, consideravelmente maior que os valores encontrados nos testes não-embarcados. Este fator garante que as variações mais bruscas no estado dos olhos sejam captadas.

5.3.1 Limitações - Inclinação da Face

A inclinação da face do motorista pode implicar na não detecção do olhos e face no *frame*, isso se deve a incapacidade do modelo de generalizar esse tipo de situação, o que está relacionado a inexistência de amostras de treinamento que submetam o modelo a essa condição. As figuras 5.8 e 5.9 abaixo ilustram o comportamento do sistema quando o indivíduo inclina o rosto:

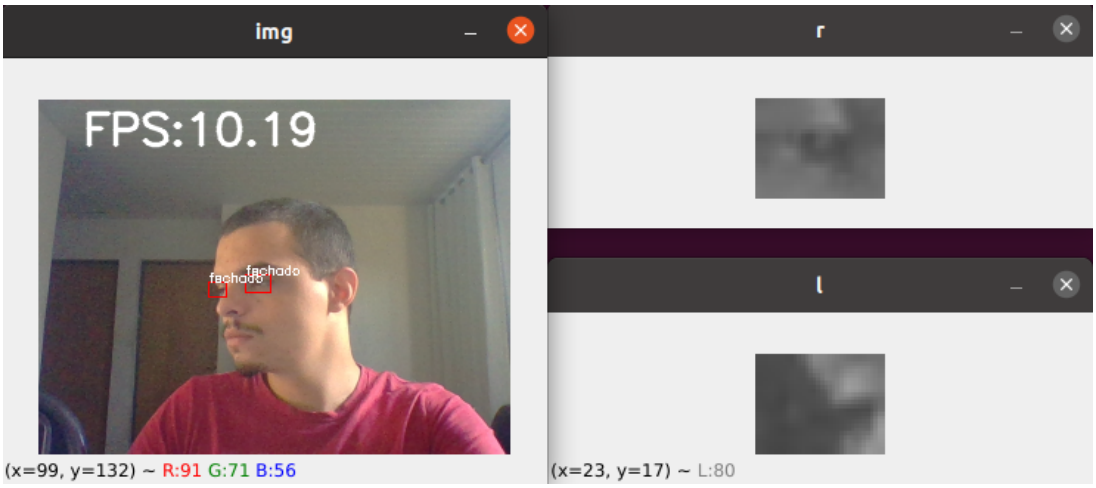


Figura 5.8: Teste embarcado com inclinação acentuada do rosto.

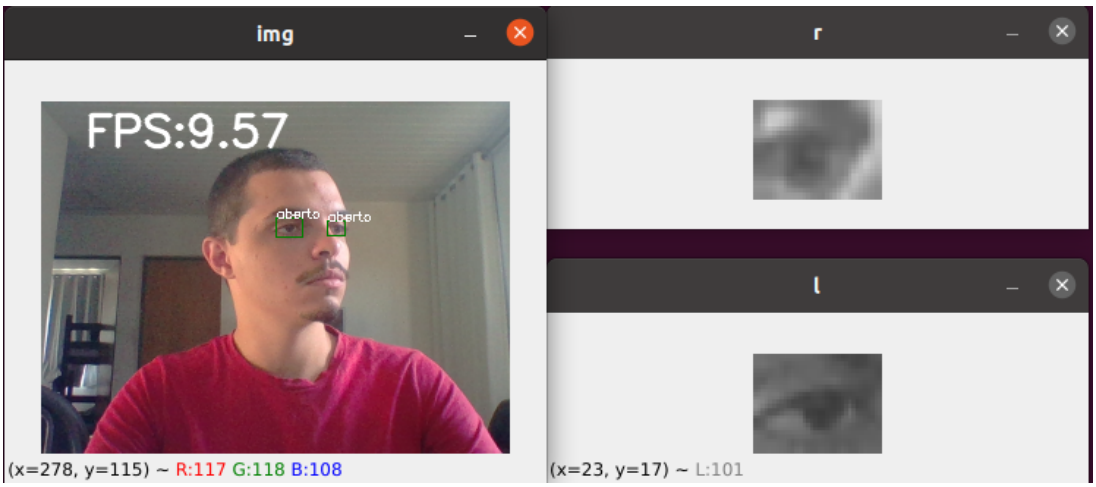


Figura 5.9: Teste embarcado com leve inclinação do rosto.

Através das imagens acima é possível afirmar que o software não é funcional para situações onde existe uma inclinação acentuada do rosto do condutor, enquanto leves inclinações não comprometem a atuação do sistema. Ainda assim, cabe ressaltar que o trabalho não se propõe a captar esse tipo de situação.

5.3.2 Limitações - Ocultação da Face

A ocultação da face é a maior limitação enfrentada pelo sistema de detecção de fadiga. Na sequência são mostrados os testes embarcados para motorista com máscara e de boné atavés das figuras 5.10 e 5.11:

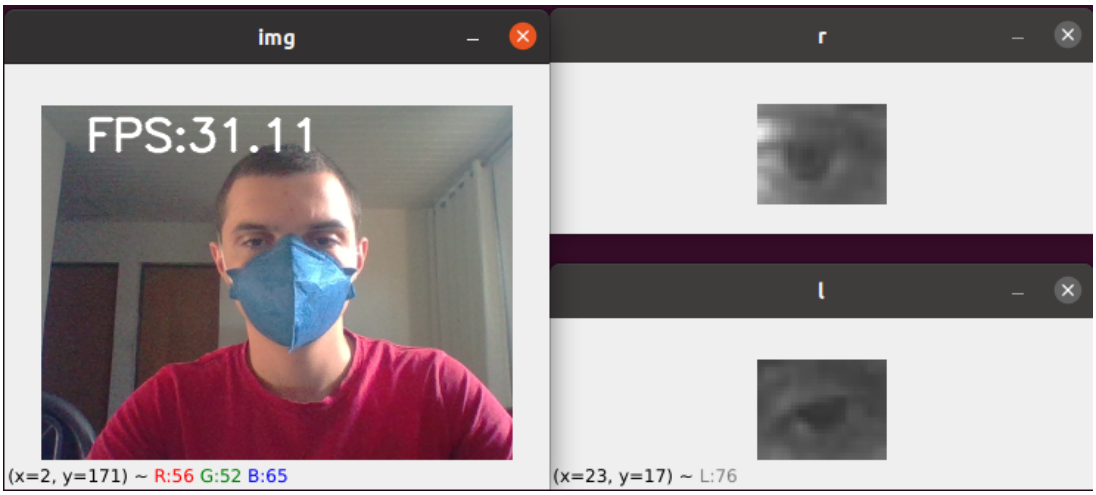


Figura 5.10: Teste embarcado para condutor com máscara

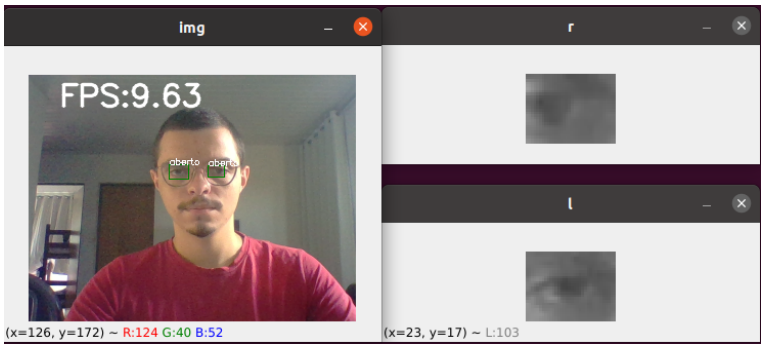
(a) (b)

(c) (d)

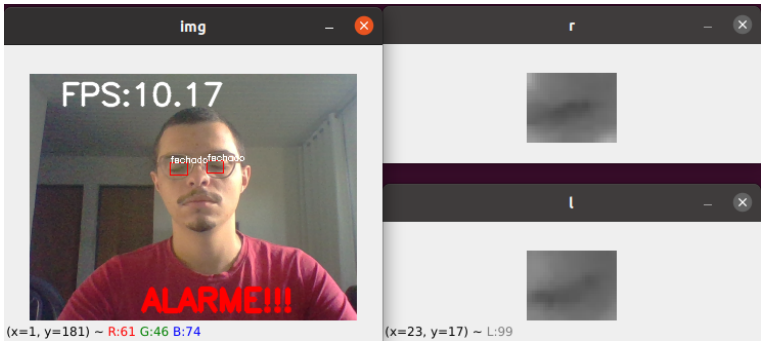
Figura 5.11: Teste embarcado para condutor com boné

Ambas as condições de ocultação de face do motorista estão fora do escopo do trabalho, sendo aconselhável a não utilização desses acessórios durante o funcionamento do sistema. Vale destacar, a recorrência do uso de máscaras por condutores veiculares profissionais por causa da pandemia de covid-19, e que tal problema seria contornado com o desenvolvimento de um algoritmo que fosse treinado com um banco de dados adequado, de rosto em diversas condições utilizando máscaras.

Alguns outros acessórios que dificultam a detecção e classificação de rosto e olhos são óculos, gorro e capuz. Testes também foram efetuados nessas condições e as figuras 5.12, 5.13 e 5.14 apresentam os resultados:

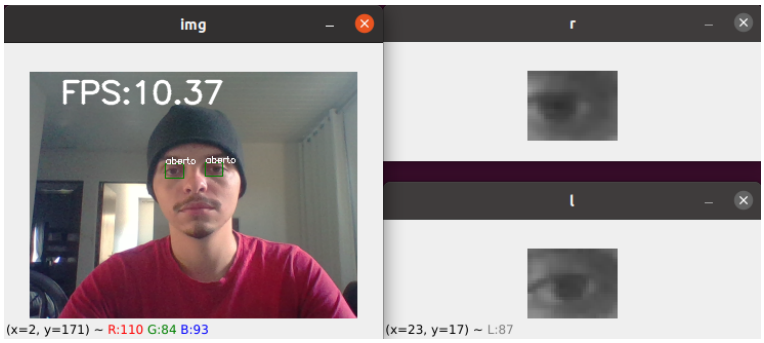


(a)

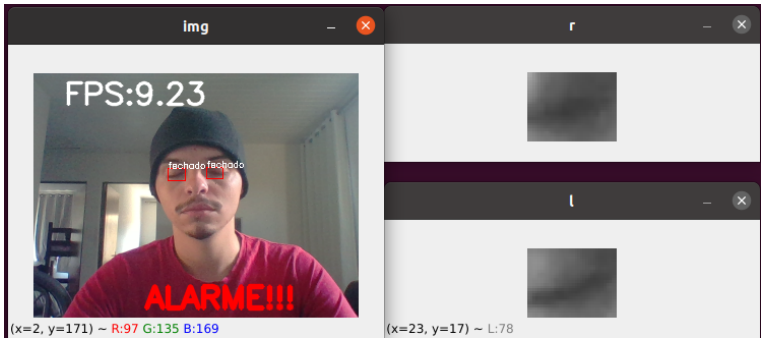


(b)

Figura 5.12: Teste embarcado para condutor com óculos

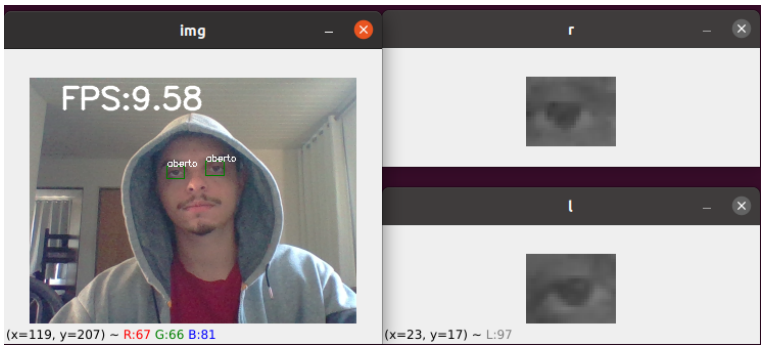


(a)

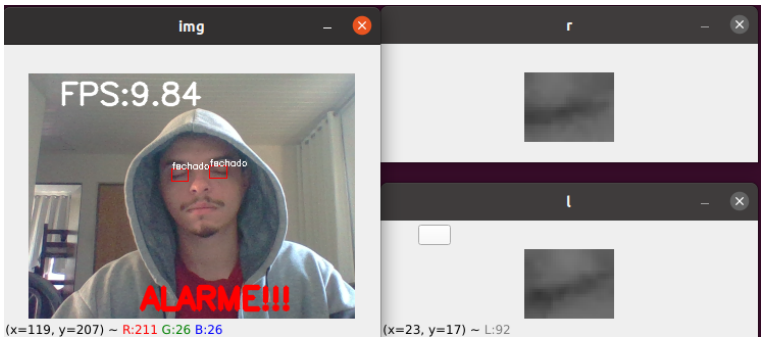


(b)

Figura 5.13: Teste embarcado para condutor com gorro



(a)



(b)

Figura 5.14: Teste embarcado para condutor com capuz

Em todas as situações descritas acima o software se comportou da maneira esperada, sendo identificados olhos e face sem problemas.

5.3.3 Limitações - Distância Entre Face e Câmera

A distância ideal do rosto do condutor para a câmera influencia o desempenho do software quando a distância da face atinge 1 metro ou mais. A figura 5.15 abaixo explicita essa situação:

Figura 5.15: Teste embarcado para condutor a cerca de 1m metro da câmera

Este fato está relacionado com a metodologia do algoritmo de detecção facial *HOG*, que opera através de convolução de filtros diferencial de dimensão 8x8, de modo que um rosto muito pequeno, ou distante da câmera, perde relevância para o classificador *SVM*.

5.3.4 Limitações - Iluminação

Os teste não-embarcados, quando condicionados sob baixa iluminação, apresentavam variações na resposta do estado dos olhos, sendo a resolução da câmera e a capacidade de processamento do computador empregada nesses testes apontados como insuficientes para garantir veracidade e velocidade de resposta. Assim sendo, testes embarcados sob estas condições de operabilidade foram realizados e as figuras 5.16 e 5.17 obtidas são apresentadas a seguir:

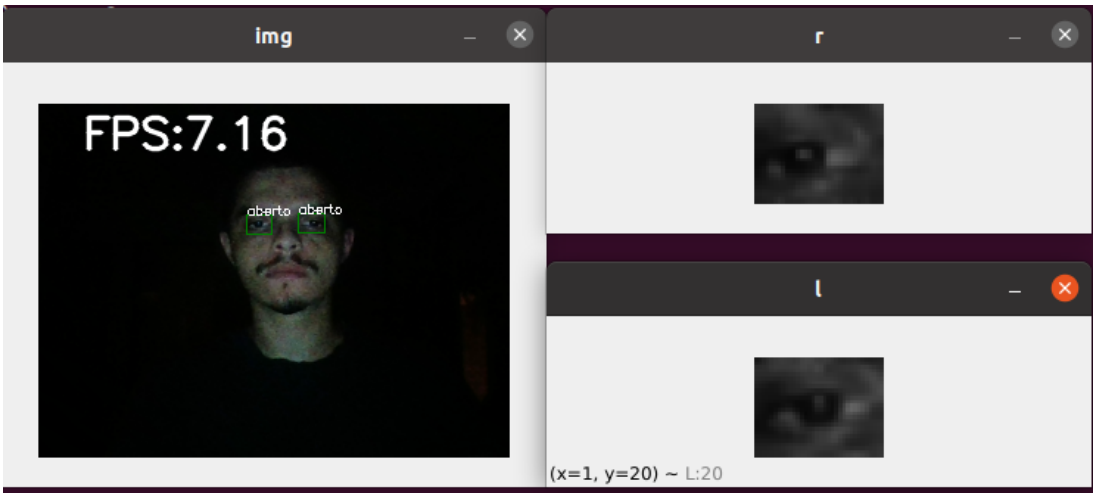


Figura 5.16: Teste embarcado com baixa iluminação com olhos abertos

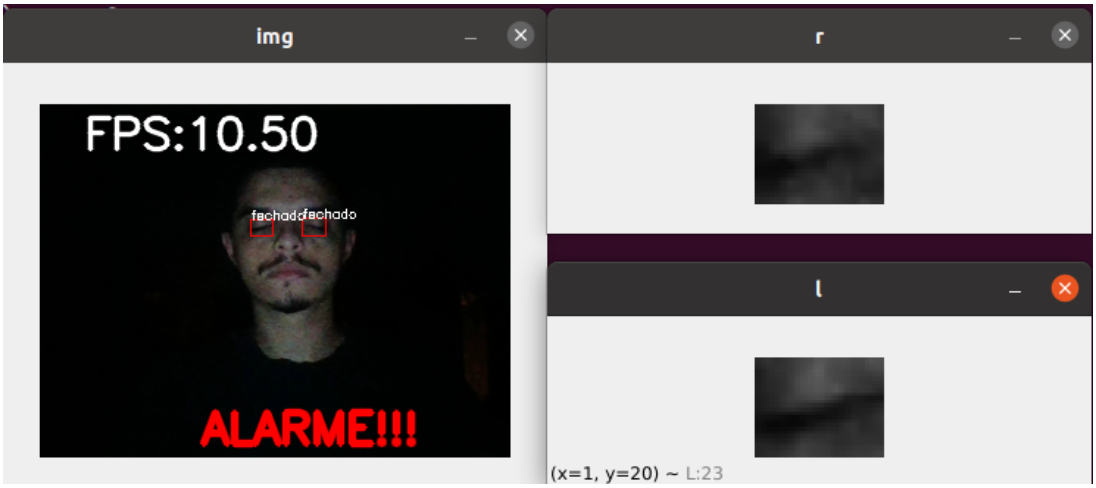


Figura 5.17: Teste embarcado com baixa iluminação com olhos fechados

Posto o contraponto com os testes não-embarcados, e os objetivos do trabalho, o bom desempenho apresentado pelo trabalho em condições de ambiente com baixa iluminação foram satisfatórios. A velocidade e a veracidade de resposta obtidas nos teste embarcados assegura a generalização de aplicação de projeto em diversas situações o que atesta a viabilidade operacional do sistema.

Capítulo 6

Conclusões Finais

Neste capítulo são apresentadas as conclusões e perspectivas futuras para o trabalho de conclusão de curso.

6.1 Conclusão

O algoritmo desenvolvido através da elaboração dos métodos de aprendizagem de máquina atenderam as expectativas, visto que os objetivos específicos como detecção facial, e detecção e reconhecimento ocular foram testados e obtiveram resultados satisfatórios. Quanto às limitações impostas pelas condições de operação do sistema, e sendo considerado o contexto do projeto, a ocultação de face se mostra o principal obstáculo no desenvolvimento de uma ferramenta ainda mais robusta quanto à variabilidade de situações presentes na sua operação.

A garantia de viabilidade operacional do sistema de detecção de fadiga parte da premissa de que o condutor não tem o rosto oculto, sendo aconselhável evitar o uso de acessórios como bonés e óculos escuros. Quanto às outras condições de operação explicadas ao longo do texto, nenhuma compromete a autenticidade e rapidez de resposta do projeto.

Sobre a capacidade de processamento, o dispositivo escolhido foi providencial para a melhora do desempenho do software desenvolvido, posta a baixa performance apresentada nos testes não-embarcados em ambientes com baixa luminosidade.

6.2 Tabela de Custos

A tabela abaixo apresenta o custo de cada um dos componentes adquiridos empregados no trabalho:

Componente	Valor
Raspberry pi 3 B	R$ 550
Câmera	R$ 100
Cartão de Memória 16GB	R$ 19,84

6.3 Trabalhos Futuros

Para a continuidade de trabalho as seguintes alternativas são elencadas:

- Emprego da biblioteca TensorLite, designada para a aplicação em soluções embarcadas;
- Emprego da biblioteca SimpleCV, uma variante da OpenCV para visão computacional, compactada para menor armazenamento;
- Estudo de outras técnicas com menor demanda computacional;
- Desenvolvimento de softwares complementares que monitorem as condições de condução veicular e de operação, de modo a modelar outros comportamentos erráticos do motorista, como manejo equivocado dos pedais, padrão de aperto do volante, troca de faixa na pista, movimentos não usual da cabeça, entre outros;
- Fabricação de circuito para substituição da matriz de contatos; e
- Construção de estrutura mecânica para protótipo.

Apêndice A

Algoritmos

A.1 Algoritmo de determinação de hiperparâmetro de regularização para detecção facial

```
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(LinearSVC(), {'C': [1.0, 2.0, 4.0, 8.0]})
grid.fit(X_train, y_train)
grid.best_score_
```

A.2 Algoritmo de modelo de detecção ocular

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import SGD

model = Sequential()
model.add(Conv2D(32, (2, 2), padding = 'same', activation='relu',
input_shape=(h1, w1, 1)))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (2, 2), padding = 'same', activation='relu',
input_shape=(h1, w1, 1)))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```
model.add(Conv2D(128, (2, 2), padding = 'same', activation='relu',
input_shape=(h1, w1, 1)))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(s, activation='relu'))

model.compile(loss='mean_squared_error', optimizer='adam',
metrics=['accuracy'])

model.fit(Xtrain, Ytrain, batch_size=512, epochs=50,
validation_data = (Xtest, Ytest), verbose = 1)
```

A.3 Algoritmo de *data augmentation*

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2
)

val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow(
    x=x_train, y=y_train,
    batch_size=32,
    shuffle=True
)

val_generator = val_datagen.flow(
```

```
    x=x_val, y=y_val,
    batch_size=32,
    shuffle=False
)
```

A.4 Algoritmo de modelo de classificação ocular

```
inputs = Input(shape=(26, 34, 1))

net = Conv2D(32, kernel_size=3, strides=1, padding='same',
activation='relu')(inputs)
net = MaxPooling2D(pool_size=2)(net)

net = Conv2D(64, kernel_size=3, strides=1, padding='same',
activation='relu')(net)
net = MaxPooling2D(pool_size=2)(net)

net = Conv2D(128, kernel_size=3, strides=1, padding='same',
activation='relu')(net)
net = MaxPooling2D(pool_size=2)(net)

net = Flatten()(net)

net = Dense(512)(net)
net = Activation('relu')(net)
net = Dense(1)(net)
outputs = Activation('sigmoid')(net)

model = Model(inputs=inputs, outputs=outputs)

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

A.5 Algoritmo de sinalização de fadiga

```
import RPi.GPIO as GPIO

led_verm = 33
```

```
led_verde = 35

GPIO.setmode(GPIO.BOARD)
GPIO.setup(led_verm, GPIO.OUT)
GPIO.setup(led_verde, GPIO.OUT)

while ret:
    ret, img = cap.read()
        if (state_l == 'fechado' and state_r == 'fechado'):
            GPIO.output(led_verm, 1)
            GPIO.output(led_verde, 0)
        else:
            GPIO.output(led_verm, 0)
            GPIO.output(led_verde, 1)
cap.release()
cv2.destroyAllWindows()
```

Referências

ABRAMET. *Problemas na saúde de motoristas causaram mais de 280 mil acidentes nas rodovias desde 2014, aponta Abramet*. Acessado: 21/06/2021, `https://www.abramet.com.br/noticias/problemas-na-saude-de-motoristas-causaram-mais-de-280-mil-acidentes-nas-rodovias`

AMARANG. *Bae Suzy Facial Landmarks*. Acessado: 15/08/2021, `https://www.kaggle.com/amarang/bae-suzy-facial-landmarks`.

ANDRADE, I.; ALBUQUERQUE, M. Portes de; ALBUQUERQUE, M. Portes de. *Processamento Digital de Imagens*. [S.l.]: Universidade Estadual do Norte Fluminense (UENF), 2021.

BABJAK, R. *ML blink detector*. Acessado: 16/08/2021, `https://gitlab.spacecode.sk/diplomka/semestralny-projekt/ml-blink-detector/-/blob/ae5acd8e6c6568562b491a6b22ad4523b9e82c42/dataset/dataset.csv`.

BALASUBRAMANIAN, V.; BHARDWAJ, R. Grip and electrophysiological sensor-based estimation of muscle fatigue while holding steering wheel in different positions. *IEEE Sensors Journal*, [S.l.], v.19, n.5, p.1951–1960, 2018.

BINIAS, B.; MYSZOR, D.; PALUS, H.; CYRAN, K. A. Prediction of pilot's reaction time based on EEG signals. *Frontiers in neuroinformatics*, [S.l.], v.14, p.6, 2020.

BOTTER MARTINS, S. *Introdução ao Processamento Digital de Imagens*. [S.l.]: Unicamp, 2020.

BOTTOU, L. *Stochastic Gradient Descent Tricks*. [S.l.]: Microsoft Research, Redmond, WA, 2021.

BROWNLEE, J. *A Gentle Introduction to k-fold Cross-Validation*.

BUDHIRAJA, A. *Dropout in (Deep) Machine learning*. Acessado: 14/08/2021, https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-mach

BUSHAEV, V. *Understanding RMSprop faster neural network learning*. Acessado: 07/08/2021, https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a.

C. GONZALEZ, R.; E. WOODS, R. *Processamento Digital de Imagens*. [S.l.]: Pearson Education, Inc, 2010.

CARPENTRY, D. *Creating Histograms*. Acessado: 05/08/2021, https://datacarpentry.org/image-processing/05-creating-histograms/.

CHOU, C.-L.; HUANG, Y.-H.; HO, S.-C. Blink Detection Using Facial Landmark Blink Detector and Multi-Layer Perceptron. In: NCS 2019 . *Anais...* [S.l.: s.n.], 2019. p.542–545.

CHOWDHURY, K. *Understanding loss functions* : hinge loss. Acessado: 09/08/2021, https://medium.com/analytics-vidhya/understanding-loss-functions-hinge-loss-a0ff112b40a1.

CORNELL. *Convolution*. Acessado: 08/08/2021, https://www.cs.cornell.edu/courses/cs1114/2013sp/sections/S06_convolution.pdf.

CREMEB. *Em dez anos, acidentes de trânsito consomem quase R$ 3 bilhões do SUS*. Acessado: 03/08/2021, https://www.cremeb.org.br/index.php/noticias/em-dez-anos-acidentes-de-transito-consomem-quase-r-3-bilhoes-do-sus/.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'05), 2005. *Anais...* [S.l.: s.n.], 2005. v.1, p.886–893 vol. 1.

DEVI, M. S.; BAJAJ, P. R. Driver fatigue detection based on eye tracking. In: FIRST INTERNATIONAL CONFERENCE ON EMERGING TRENDS IN ENGINEERING AND TECHNOLOGY, 2008. *Anais...* [S.l.: s.n.], 2008. p.649–652.

DUA, M.; SINGLA, R.; RAJ, S.; JANGRA, A. *et al*. Deep CNN models-based ensemble approach to driver drowsiness detection. *Neural Computing and Applications*, [S.l.], v.33, n.8, p.3155–3168, 2021.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, [S.l.], v.12, n.7, 2011.

ERIKSSON, M.; PAPANIKOTOPOULOS, N. Eye-tracking for detection of driver fatigue. In: CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS. *Proceedings...* [S.l.: s.n.], 1997. p.314–319.

FIGUEIREDO, V. *Seus primeiros passos como Data Scientist*: introdução ao pandas! Acessado: 14/08/2021, `https://medium.com/data-hackers/uma-introdução-simples-ao-pandas-1e15eea37fa1`.

FOUNDATION, R. *Raspberry pi 4 Computer*. Acessado: 05/12/2021, `https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf`.

GANDHI, A. *Data Augmentation | How to use Deep Learning when you have Limited DataPart 2*. Acessado: 11/08/2021, `https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/`.

GAY, W. *Advanced Raspberry Pi*: raspbian linux and gpio integration. [S.l.]: Apress, 2018.

GODOY, D. *Uma explicação visual para função de custo binary cross-entropy ou log loss*. Acessado: 11/08/2021, `https://medium.com/ensina-ai/uma-explicação-visual-para-função-de-custo-binary-cross-entropy-ou-log-loss-eaee`

GUPTA, A.; EFROS, A. A.; HEBERT, M. Blocks World Revisited: image understanding using qualitative geometry and mechanics. In: COMPUTER VISION – ECCV 2010, Berlin, Heidelberg. *Anais...* Springer Berlin Heidelberg, 2010. p.482–496.

HAMID, N.; RAZALI, R. A.; IBRAHIM, Z. Comparing bags of features, conventional convolutional neural network and AlexNet for fruit recognition. *Indonesian Journal of Electrical Engineering and Computer Science*, [S.l.], v.14, n.1, p.333–339, 2019.

HEMALATHA, G.; SUMATHI, C. A study of techniques for facial detection and expression classification. *International Journal of Computer Science and Engineering Survey*, [S.l.], v.5, n.2, p.27, 2014.

IMPORT.IO. *What is Data Normalization and Why Is It Important?* Acessado: 05/08/2021, `https://www.import.io/post/what-is-data-normalization-and-why-is-it-important/`.

JAIN A. K.; SHARMA, R. S. A. *A Review of Face Recongnition System Using a Raspberry Pi in the Field of IoT*. Acessado: 03/12/2021, `https://easychair.org/publications/open/pmvP`.

JORDAN, A. A.; PEGATOQUET, A.; CASTAGNETTI, A.; RAYBAUT, J.; LE COZ, P. Deep Learning for Eye Blink Detection Implemented at the Edge. *IEEE Embedded Systems Letters*, [S.l.], p.1–1, 2020.

KIM, K. W.; HONG, H. G.; NAM, G. P.; PARK, K. R. A study of deep CNN-based classification of open and closed eyes using a visible light camera sensor. *Sensors*, [S.l.], v.17, n.7, p.1534, 2017.

L. BRUNTON, S.; KUTZ, N. *Data Driven Science Engineering*: machine learning, dynamical systems, and control. [S.l.]: University of Washington, 2017.

LAMBLET VAZ, A. *Gradientes Descendentes na prática melhor jeito de entender*. Acessado: 07/08/2021, `https://medium.com/data-hackers/gradientes-descendentes-na-prática-melhor-jeito-de-entender-740ef4ff6c43`.

LEARNING, D. I. D. *Padding and Stride*. Acessado: 11/08/2021, `https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html`.

LFW. *Labeled Faces in the Wild Home*. Acessado: 15/08/2021, `http://vis-www.cs.umass.edu/lfw/`.

LUO, S. *Loss Function(Part III)*: support vector machine. Acessado: 15/08/2021, `https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d`.

MARQUE FILHO O.; NETO, H. V. *Processamento Digital de Imagens*. [S.l.]: Brasport, 1999.

MARQUES FILHO, O.; VIEIRA NETO, H. *Processamento Digital de Imagens*. [S.l.]: Brasport, 1999.

MARR, D. *Vision*: a computational investigation into the human representation and processing of visual information. [S.l.]: The MIT Press, 1970.

MASTERY, M. L. *Why Optimization Is Important in Machine Learning*. Acessado: 07/08/2021, `https://machinelearningmastery.com/why-optimization-is-important-in-machine-learning/`.

MIT. *Nós não estamos preparados para o fim da lei de Moore*. Acessado: 21/06/2021, `https://mittechreview.com.br/nos-nao-estamos-preparados-para-o-fim-da-lei-de-moore/`.

MITTAL, K. *A Gentle Introduction Into The Histogram Of Oriented Gradients*. Acessado: 08/08/2021, https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa.

MIZUNO, K.; TERACHI, Y.; TAKAGI, K.; IZUMI, S.; KAWAGUCHI, H.; YOSHIMOTO, M. Architectural study of HOG feature extraction processor for real-time object detection. In: IEEE WORKSHOP ON SIGNAL PROCESSING SYSTEMS, 2012. *Anais...* [S.l.: s.n.], 2012. p.197–202.

NETWORK, A. E. *Max Pooling in Convolutional Neural Network and Its Features*. Acessado: 11/08/2021, https://analyticsindiamag.com/max-pooling-in-convolutional-neural-network-and-its-features/.

NISSAN. *WHAT IS NISSAN INTELLIGENT SAFETY SHIELD?* Acessado: 11/07/2021, https://www.delandnissan.com/nissan-safety-shield-technologies-near-deland-fl.html.

PAPERT, S. A. *The Summer Vision Project*. Acessado: 21/06/2021, https://dspace.mit.edu/handle/1721.1/6125.

PAULY, L.; SANKAR, D. Non intrusive eye blink detection from low resolution images using HOG-SVM classifier. *International Journal of Image, Graphics and Signal Processing*, [S.l.], v.8, n.10, p.11, 2016.

PERKONS. *Mais de 50 anos da Lei de Moore*. Acessado: 21/06/2021, http://www.perkons.com/pt/noticia/1782/sono-e-fadiga-sao-a-terceira-causa-de-acidentes-de-transito-no-brasil.

POKHREL, S. *6 Obstacles to Robust Object Detection*: how robust is your detector. Acessado: 21/06/2021, https://towardsdatascience.com/6-obstacles-to-robust-object-detection-6802140302ef.

POYNTON, C. *Frequently Asked Questions about Color*. Acessado: 05/08/2021, http://poynton.ca/PDFs/ColorFAQ.pdf.

PRAKASH, J. *Non Maximum Suppression*: theory and implementation in pytorch. Acessado: 09/08/2021, https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/.

QIAN, N. On the momentum term in gradient descent learning algorithms. *Neural networks*, [S.l.], v.12, n.1, p.145–151, 1999.

ROCHA, E. *Piscamos cerca de 20 vezes por minuto*. Acessado: 19/08/2021, https://jornal.usp.br/atualidades/piscamos-cerca-de-20-vezes-por-minuto/.

RUDER, S. An overview of gradient descent optimization algorithms. *CoRR*, [S.l.], v.abs/1609.04747, 2016.

SANTIAGO JR., L. *Entendendo a biblioteca NumPy*. Acessado: 14/08/2021, https://medium.com/ensina-ai/entendendo-a-biblioteca-numpy-4858fde63355.

SAVAL, P. *Chapter 2* : svm (support vector machine) theory. Acessado: 15/08/2021, https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.

SBMT. *Acidentes de trânsito*: mais de 1,35 milhão de pessoas perdem a vida, aponta oms. Acessado: 03/08/2021, https://www.sbmt.org.br/portal/traffic-accidents-over-1-35-million-people-lose-their-lives-says-who/.

SCIKIT. *Histogram of Oriented Gradients*. Acessado: 14/08/2021, https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html.

SCIKIT. *C-Support Vector Classification*. Acessado: 14/08/2021, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

SCIKIT. *The Labeled Faces in the Wild face recognition dataset*. Acessado: 15/08/2021, https://scikit-learn.org/0.16/datasets/labeled_faces.html.

SCIKIT-IMAGE. *RGB to grayscale*. Acessado: 11/07/2021, https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_gray.html.

SOLOMON, B. *Python Plotting With Matplotlib (Guide)*. Acessado: 14/08/2021, https://realpython.com/python-matplotlib-guide/.

STANFORD. *Neural Networks Part 1*: setting up the architecture. Acessado: 11/08/2021, https://cs231n.github.io/neural-networks-1/.

VOLKSWAGEN. *Driver Alert System*: senses when you start to feel tired and need a break. Acessado: 11/07/2021, https://www.volkswagen.co.uk/technology/car-safety/driver-alert-system.

YANG, H.; LIU, L.; MIN, W.; YANG, X.; XIONG, X. Driver yawning detection based on subtle facial action recognition. *IEEE Transactions on Multimedia*, [S.l.], v.23, p.572–583, 2020.

ZHU, X.; GOLDBERG, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, [S.l.], v.3, n.1, p.1–130, 2009.