

UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

LUCAS ALMEIDA DA SILVA

SISTEMA DE CLASSIFICAÇÃO DE FACES PARA SISTEMAS
EMBARCADOS COM DEEP LEARNING

Manaus

2019

LUCAS ALMEIDA DA SILVA

**SISTEMA DE CLASSIFICAÇÃO DE FACES PARA SISTEMAS
EMBARCADOS COM DEEP LEARNING**

Trabalho de Conclusão de Curso apresentado
à banca avaliadora do Curso de Engenharia
de Computação, da Escola Superior de
Tecnologia, da Universidade do Estado do
Amazonas, como pré-requisito para obtenção
do título de Bacharel em Engenharia de
Computação.

Orientador: Prof. Dr. Jucimar Maia da Silva Júnior

Coorientadora: Profa. MSc. Cristina Souza de Araújo

Manaus – Dezembro – 2019

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

S586zs SILVA, LUCAS ALMEIDA DA
Sistema de Classificação de Faces para Sistemas
Embarcados com Deep Learning / LUCAS ALMEIDA
DA SILVA. Manaus : [s.n], 2019.
53 f.: il., color.; 30 cm.

TCC - Graduação em Engenharia de Computação -
Universidade do Estado do Amazonas, Manaus, 2019.

Inclui bibliografia

Orientador: Jucimar Maia da Silva Júnior

Coorientador: Cristina Souza de Araújo

1. Sistema embarcado. 2. Classificação de Faces. 3.
Aprendizado profundo. I. Jucimar Maia da Silva Júnior
(Orient.). II. Cristina Souza de Araújo (Coorient.). III.
Universidade do Estado do Amazonas. IV. Sistema de
Classificação de Faces para Sistemas Embarcados com
Deep Learning

Elaborado por Jeane Macelino Galves - CRB-11/463

LUCAS ALMEIDA DA SILVA

**SISTEMA DE CLASSIFICAÇÃO DE FACES PARA SISTEMAS
EMBARCADOS COM DEEP LEARNING**

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenharia de Computação.

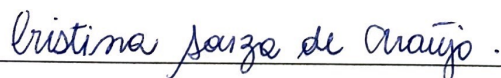
BANCA EXAMINADORA

Aprovado em: 06/12/2019



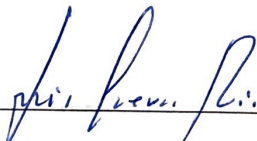
Prof. Dr. Jucimar Maia da Silva Júnior

UNIVERSIDADE DO ESTADO DO AMAZONAS



Profa. MSc. Cristina Souza de Araújo.

UNIVERSIDADE DO ESTADO DO AMAZONAS



Prof. Dr. Luis Cuevas Rodríguez.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Resumo

É apresentado um sistema que utiliza *Deep Learning* para reconhecimento e classificação de faces. Por meio de redes neurais profundas altamente eficientes, leves e rápidas será desenvolvido um sistema para aplicações mobile e embarcadas. Diante da necessidade de menor custo computacional será realizado um balanceamento entre latência e precisão.

Palavras Chave: Sistema embarcado, Classificação de Faces, Aprendizado profundo

Abstract

A system that uses deep learning for recognition and classification of faces is presented. Through highly efficient, light and fast deep neural networks will be developed a system for mobile and embedded applications. Faced with the need for lower computational cost, a trade off between latency and precision will be constantly performed.

Key-words: Embedded System, Face recognition, Deep Learning

Agradecimentos

Agradeço a Deus pela vida que Ele me concedeu. Só cheguei aqui por conta Dele e é por Ele todas estas coisas.

Agradeço aos meus pais por todo o esforço investido na minha educação. Muito obrigado por terem me ensinado desde criança o caminho que deveria seguir.

Agradeço aos meus irmãos pelo grande apoio nas batalhas do dia a dia.

Agradeço à minha namorada que sempre esteve ao meu lado e me ajuda em tudo que preciso.

Sou grato pelas várias oportunidades dadas pelos professores orientadores neste trabalho. Professor Jucimar quem me abriu as portas para fazer parte do Ludus e durante muitos anos me ensinou muito além do ensino técnico.

Professora Cristina, quem iniciou minha carreira profissional e diariamente me ensina, estimula e promove melhorias pessoais, técnicas e científicas em minha formação.

Professor Cuevas, quem abriu meus olhos quanto ao pensamento científico.

Por último, quero agradecer também à Universidade do Estado do Amazonas e todo o seu corpo docente.

Sumário

Lista de Tabelas	ix
Lista de Figuras	xi
1 Introdução	1
1.1 Descrição do problema	1
1.2 Objetivos	1
1.2.1 Objetivo geral	1
1.2.2 Objetivos específicos	2
1.3 Justificativa	2
1.4 Metodologia	3
1.5 Estruturação da Monografia	3
2 Revisão de Literatura	5
2.1 Inteligência Artificial	5
2.1.1 História da inteligência artificial	5
2.1.2 Conceitos de Inteligência Artificial	6
2.1.3 Fundamentos da Inteligência Artificial	6
2.1.4 Aplicações	8
2.2 <i>Machine Learning</i>	10
2.2.1 História sobre o Aprendizado de máquina	10
2.2.2 Conceitos de <i>Machine Learning</i>	10

2.2.3	Justificativa para uso de <i>Machine Learning</i>	11
2.2.4	Tipos	11
2.2.5	<i>Machine Learning</i> e o reconhecimento de Imagens	12
2.3	Visão Computacional	13
2.4	<i>Deep Learning</i>	16
2.4.1	Redes Neurais	16
2.4.2	Conceitos de <i>Deep Learning</i>	16
2.4.3	Como funciona	17
2.5	<i>Deep Learning</i> X <i>Machine Learning</i>	22
2.6	<i>Deep Learning</i> X Sistemas Embarcados	23
2.6.1	<i>Transfer Learning</i>	24
2.7	Detecção Facial	25
2.7.1	<i>Haar Cascade</i>	26
2.7.2	<i>Deep Learning Model</i>	27
2.8	Tecnologias	27
2.8.1	PostgreSQL	28
2.8.2	Flask	28
2.8.3	uWSGI	29
2.8.4	NGINX	29
2.8.5	Raspiberry PI	30
2.8.6	OpenCV	31
2.8.7	Dlib	31

3 Desenvolvimento 32

3.1	Descrição Geral do Sistema	32
3.2	Especificação do Sistema	33
3.2.1	Levantamento de Requisitos	33
3.2.2	Diagrama de Atividades	35
3.2.3	Diagrama de Componentes	36

3.3	Detectores Faciais	38
3.4	Modelos para Reconhecimento Facial	39
3.4.1	Módulo para Cadastro	40
3.4.2	Módulo para Reconhecimento Facial	41
3.4.3	Módulo treino de RNN	42
4	Resultados	45
4.1	Detecção Facial	45
4.1.1	Seleção de hiperparâmetros	45
4.1.2	<i>Haar Cascade</i>	45
4.1.3	DNN	47
4.1.4	Algoritmo de detecção facial proposto	48
4.2	Cadastro de novas Classes	49
4.3	Reconhecimento Facial	49
4.4	Desempenho	50
5	Considerações Finais	51

Lista de Tabelas

2.1	ML x DL	23
2.2	Bibliotecas de detecção facial	31
3.1	Requisitos Não Funcionais	34
4.1	Resultados Ganho de informação por hiperparâmetro	46
4.2	Resultados Ganho de informação por hiperparâmetro	47

Lista de Figuras

2.1	Processo de aprendizado de máquina	13
2.4	Uma rede neural profunda para classificação de dígitos	17
2.5	Representações profundas aprendidas do modelo de classificação de dígitos. . . .	18
2.6	Representação do aprendizado pelo ajuste dos pesos	19
2.7	O papel da função de perda no aprendizado	20
2.8	O papel do otimizador no aprendizado	21
2.9	Hierarquia de aprendizado	22
2.10	Treino Tradicional X <i>Transfer Learning</i>	24
2.11	Características Haar	26
2.12	Fluxo <i>Haar Cascade</i>	27
3.1	Diagrama de atividades	35
3.2	Diagrama de componentes	36
3.3	Componente de detecção facial	37
3.4	Componente de IA	37
3.5	Diagrama de componentes completo	38
3.6	Fluxo de cadastro	40
3.7	Fluxo de reconhecimento	41
4.1	Resultados Haar Cascade	47
4.2	Resultados DNN	48
4.3	Interface para captura no estado de captura de olhos	49
4.4	Interface mostrando a captura feita por um dispositivo Raspberry PI	50

Capítulo 1

Introdução

1.1 Descrição do problema

Com o novo mundo globalizado e o advento de tecnologias que funcionam em tempo real, a luta contra o fenômeno da latência se intensificou. Na ciência da computação *Real-Time Computing* - *RTC* descrevem-se sistemas de software com tempos de respostas restritos ao "tempo real". Esses programas devem garantir uma resposta dentro de restrições de tempo pré estabelecidas comumente chamadas de *deadlines*.(BEN-ARI, 1990).

Então embora já existam diversos modelos para detecção e classificação de imagens dos quais, em geral, são pesados e com alto custo computacional. É necessário um sistema escalável, leve e flexível que permita alta velocidade em gerar modelos com grande precisão.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo geral deste trabalho é o desenvolvimento de um sistema de classificação de faces, com baixo custo computacional, para sistemas embarcados que se caracterize como *Real-Time Computing* - *RTC*.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Efetuar a fundamentação teórica dos conceitos relativos ao Aprendizado de Máquina, Aprendizado Profundo e Redes Neurais Convolucionais;
2. Realizar análise em trabalhos relacionados a fim de obter métricas e empregar técnica já existentes.
3. Consolidar uma base de dados para treino.
4. Desenvolver módulo para treinamento de redes neurais profundas, permitindo o ajuste de parâmetros de treino.
5. Desenvolver módulo para classificação de faces com o modelo obtido no treino.
6. Desenvolver a solução proposta baseada em sistema Web;
7. Avaliar a acurácia e desempenho do módulo de inferência.

1.3 Justificativa

Sistemas de monitoramento, jogos online e até na indústria 4.0, há imposição de tempo de resposta extremamente baixo. Muitos dos recursos de Inteligência Artificial e Visão Computacional não são explorados nestes setores, devido a estas imposições de tempo e custo.

Este trabalho ocorreu numa época onde *Internet of Things-IOT* (à interconexão digital de objetos cotidianos com a internet) está em alta. Embarcar modelos e criar servidores de inferência com baixo custo computacional, será primordial para um futuro em que sistemas pequenos e inteligentes terão mais espaço.

1.4 Metodologia

Inicialmente, o objetivo foi determinar as técnicas de Inteligência Artificial a serem aplicadas no preditor a ser desenvolvido, bem como definir a arquitetura do sistema. Ambas as etapas baseiam-se essencialmente no levantamento bibliográfico realizado, permitindo assim, definir escolhas em função dos melhores resultados já alcançados.

Com base nos requisitos e regras de negócio do sistema foi realizado o levantamento das principais ferramentas e bibliotecas de software a serem utilizadas, levando-se em consideração:

- Capacidade de interoperabilidade
- Controle dos recursos de Hardware
- Escalabilidade, Performance e Elasticidade em sistemas distribuídos.

A partir do resultado da simulação feita com base de dados em algoritmo próprio para coleta de *Dataset*, foi observado que, embora um conjunto de dados maior agregue mais informação e ganho de acurácia, há um maior esforço e tempo para coleta das imagens dos usuários, dessa forma descartou-se utilizar *datasets* que impliquem em um tempo dispendioso para o cliente. Além disso, o uso de bases de dados maiores tornaria o teste impraticável, dado o tempo de treinamento e o poder computacional necessários para sua realização.

1.5 Estruturação da Monografia

Este documento está estruturado em quatro capítulos. O capítulo 1, Introdução, apresentou a visão geral do trabalho e seus objetivos. O capítulo 2 abordará os conceitos utilizados como base ao longo do trabalho, em que são listados conceitos que dizem respeito aos temas computacionais que concernem ao trabalho, como conceitos de *IA*, *Machine Learning* e *Deep Learning*, possíveis aplicações de inteligência artificial, tipos de aprendizado de máquina, visão computacional e tecnologias web. O capítulo 3 descreve o desenvolvimento do trabalho, o qual é resultante da metodologia proposta. Estão aí contidos como foram feitos o projeto e treinamento do modelo

de aprendizado, a obtenção dos dados, a validação do preditor, arquitetura do sistema e o relacionamento entre componentes. O capítulo 4 apresentam-se conclusões, como resultados obtidos.

Capítulo 2

Revisão de Literatura

2.1 Inteligência Artificial

Compreender o simples ato de "pensar" tem sido objeto de debate durante milhares de anos. O ser humano, um mero punhado de matéria, pode perceber, compreender, prever e manipular um mundo muito maior do que ele próprio. O campo da inteligência artificial, ou IA, vai ainda mais além: ele tenta não apenas compreender, mas também construir entidades inteligentes. (RUSSELL PETER NORVIG, 2013)

Foi com ajuda das Técnicas de Inteligência Artificial, em particular Aprendizado de Máquina, que computadores obtiveram grande sucesso na resolução de problemas considerados difíceis ou até mesmo impossíveis para seres humanos. (FACELLI et al., 2015)

2.1.1 História da inteligência artificial

A Inteligência Artificial (IA) surgiu por volta dos anos 50, quando os pioneiros na nova área da Ciência da Computação se questionavam se os computadores poderiam ser feitos para pensar - Uma questão cujas ramificações continuam sendo exploradas nos dias atuais (FRANCOIS, 2017). Entretanto, essa área era visualizada apenas como teórica, cujas aplicações eram desempenhadas em pequenos problemas curiosos, desafiadores e com pouco valor prático. A partir da década de 1970, em paralelo ao avanço dos computadores, houve a maior disseminação do

uso de técnicas de computação baseadas em IA para a solução de problemas reais. Na maioria das vezes, os problemas eram solucionados computacionalmente por meio de regras lógicas, que eram codificadas através do conhecimento de um especialista de uma área específica. (FACELLI et al., 2015)

2.1.2 Conceitos de Inteligência Artificial

A IA é um dos campos mais recentes em ciências e engenharia. O trabalho começou logo após a Segunda Guerra Mundial, e o próprio nome foi cunhado em 1956. Há um debate histórico na própria definição de "inteligência artificial". Uma abordagem empírica centrada nos seres humanos e outra racionalista envolvendo uma combinação de matemática e engenharia. É possível ver na literatura este embate. (RICH; KNIGHT, 1991) afirma que IA é "O estudo de como os computadores podem fazer tarefas que hoje são melhor desempenhadas pelas pessoas." enquanto (CHARNIAK; MCDERMOTT, 1985) diz que inteligência artificial é "O estudo das faculdades mentais pelo uso de modelos computacionais".

2.1.3 Fundamentos da Inteligência Artificial

Embora IA possa ser aplicada a qualquer área do saber, os campos de estudo a seguir, contribuíram grandemente com ideias, pontos de vista e técnicas para a área:

Filosofia Aristóteles (384-322 a.C.) foi o primeiro a formular um conjunto preciso de leis que governam a parte racional da mente. Desenvolveu um sistema informal de silogismos para raciocínio que permitiam, em princípio, gerar conclusões mecanicamente, dadas as premissas iniciais. Muitos filósofos ao longo da história, como Ramon Lull (1315), Hobbes (1588-1679) e Leonardo da Vinci (1452-1519) contribuíram para desenvolver fundamentos da inteligência artificial e responder questões como as seguintes: "Como obter regras formais podem ser usadas para obter conclusões válidas?", "Como o intelecto se desenvolve a partir de um cérebro físico", "De onde vem o conhecimento?".

Matemática Embora seja dos filósofos o crédito sob o desenvolvimento de muitas ideias importante sobre IA, é responsabilidade dos matemáticos o salto para uma ciência formal. Essa formalização foi trabalhada em 3 áreas fundamentais: lógica, computação e probabilidade. (RUSSELL PETER NORVIG, 2013)

Algoritmos surgiram no campo da matemática para realizar induções matemáticas. Kurt Gödel (1906-1978) mostrou que existe um procedimento efetivo para provar qualquer afirmação verdadeira na lógica de primeira ordem de Frege e Russell. Seu teorema da incompletude mostrou que, em qualquer teoria formal tão forte como a aritmética de Peano (a teoria elementar dos números naturais), existem afirmações verdadeiras que são indecidíveis no sentido de que não existem provas na teoria. Isto é, **a demonstração de que existem algumas funções sobre os inteiros que não podem ser representados por um algoritmo.**

A teoria da NP-completude, apresentada primeiro por Steven Cook (1971) e Richard Karp (1972), fornece um método para reconhecer um problema intratável. Mas a IA ajudou a categorizar algumas instâncias de problemas NP-completos entre problemas difíceis, e outros mais fáceis. (CHEESEMANN P.; TAYLOR, 1991)

Neurociência O ser humano é o principal exemplo de uma entidade inteligente. Em meados do século XVIII, o cérebro foi amplamente reconhecido como a sede da consciência e responsável pelo pensamento, e surgiu a neurociência como o estudo do sistema nervoso, em particular, o cérebro. Foi por meio dos estudos dessa área que descobriu-se que o cérebro consistia em células nervosas ou **neurônio**. Estes conceitos são amplamente utilizados na área em particular ***Machine Learning***.

Psicologia Foi por meio da psicologia a síntese de conceitos repartidos em diversas áreas do conhecimento. A visão do cérebro como um dispositivo de processamento de informações, uma característica importante da psicologia cognitiva, tem suas origens nos trabalhos de William James (1842-1910).

O nascimento da **ciência cognitiva** aconteceu em um seminário, em setembro de 1956 no MIT. No seminário, George Miller apresentou *The Magic Number Seven*, Noam Chomsky

apresentou *Three Models of Language* e Allen Newell e Herbert Simon apresentaram *The Logic Theory Machine*. Esses três documentos influentes mostraram como modelos de computadores podiam ser usados para tratar a psicologia da memória, a linguagem e o pensamento lógico, respectivamente.

Entre os psicólogos é comum a visão de que “uma teoria cognitiva deve ser como um programa de computador” (ANDERSON, 1980), isto é, ela deve descrever um mecanismo detalhado de processamento de informações por meio do qual alguma função cognitiva poderia ser implementada.

Engenharia de computadores ”Para a inteligência artificial ter sucesso, precisamos de inteligência e de um artefato. O computador tem sido o artefato preferido.”(RUSSELL PETER NORVIG, 2013). Enquanto IA teve sua fundamentação trabalhada desde antes de Cristo, o primeiro computador *operacional* só foi contruída em 1940, pela equipe de Alan Turing. Em 1941 surgiu Z-3, o primeiro computador *programável*, criado por Konrad Zuse na Alemanha.

Deste então, cada geração de *hardware* computador trouxe um aumento em velocidade e capacidade, e redução no preço. Com problemas de energia, os fabricantes começaram a multiplicar o número de núcleos de CPU e não mais a aumentar a velocidade de *clock*. Espera-se que os computadores trabalhem, assim como o cérebro humano com um paralelismo maciço. Isto, na verdade, já é utilizado amplamente na área de ***Deep Learning*** com hardwares cada vez mais preparados para **computação paralela**.

2.1.4 Aplicações

É uma tarefa difícil encontrar uma explicação concisa sobre as aplicações de inteligência artificial, porque existem muitas atividades em vários subcampos que a IA pode ser aplicada. A seguir, nota-se algumas dessas aplicações que já estão amplamente em utilização:

- Veículos robóticos: Já existem veículos autônomos que através da Inteligência artificial e Visão computacional são capazes de se locomover na cidade, com desempenho em constante evolução.

- Reconhecimento de voz: Muitos sistemas de atendimento por telefone já utilizam IA para reconhecimento de voz e gestão de diálogo.
- Planejamento autônomo e escalonamento: A uma centena de milhões de quilômetros da Terra, o programa Remote Agent da Nasa se tornou o primeiro programa de planejamento autônomo de bordo a controlar o escalonamento de operações de uma nave espacial.
- Jogos: O DEEP BLUE da IBM se tornou o primeiro programa de computador a derrotar o campeão mundial Garry Kasparov em uma partida de xadrez em 1997. Isso elevou o valor das ações da IBM a um aumento de 18 bilhões de dólares.
- Combate a spam: Algoritmos de aprendizagem já conseguem classificar de maneira eficiente, mensagens como spam, poupando destinatários de receberem lixo eletrônico que poderia corresponder até 80% de todas as mensagens. Devido aos spammers estarem constantemente atualizando suas táticas, é difícil que uma abordagem estática programada se mantenha, e algoritmos de aprendizagem funcionam melhor. (GOODMAN; HECKERMAN, 2004)
- Planejamento logístico: Durante a crise do Golfo Pérsico em 1991, as forças armadas dos Estados Unidos distribuíram uma ferramenta denominada *Dynamic Analysis and Replanning Tool*, ou DART. (CROSS; WALKER, 1994)
- Robótica: A *iRobot Corporation* já vendeu mais de dois milhões de aspiradores robóticos Roomba para uso doméstico. Eles possuem uma inteligência artificial para trabalhar sozinhos em ambientes domésticos.
- Tradução automática: A partir de um modelo estatístico, construído a partir de exemplos de traduções de um determinado idioma, são construídos programas de computador capazes de realizar traduções de forma precisa, mesmo sem alguém da equipe de desenvolvimento ter conhecimento do idioma.

2.2 *Machine Learning*

Nas últimas décadas, embora a Inteligência Artificial tenha provado ser adequada para solução de problemas bem descritos e problemas lógicos, surgiu a necessidade de ferramentas computacionais mais robustas que reduzissem a intervenção humana e a dependência de especialistas (FACELLI et al., 2015). *Machine Learning*, ou Aprendizado de Máquina, é análoga a forma como o ser humano aprende.

2.2.1 História sobre o Aprendizado de máquina

Em 1959, um cientista da computação da IBM chamado Arthur Samuel, escreveu um programa de computador para jogar xadrez. A cada posição do tabuleiro foi dada uma pontuação, com base em posicionamento, que elevariam a chance de vitória. Em 1970, Arthur desenvolveu um programa de computador capaz de melhorar sua performance, por meio da experiência chegando ao nível de um jogador amador de xadrez. Aqui o aprendizado de máquina nasceu e, como falado anteriormente, em o DEEP BLUE da IBM derrotou o campeão mundial Garry Kasparov em uma partida de xadrez em 1997. A primeira vitória da máquina sob o intelecto humano.

2.2.2 Conceitos de *Machine Learning*

Machine Learning, é uma ramificação da Inteligência Artificial que trata do estudo sistemático de algoritmos e sistemas, que são capazes de melhorar seu desempenho com a experiência. Um algoritmo neste domínio é capaz de aprender a partir de dados, através da captura de padrões e efetuando inferências com intuito de solucionar um respectivo problema.

A Inteligência Artificial emprega o paradigma de programação clássico, onde as entradas do programa são as regras e os dados que serão processados pelas mesmas, para a obtenção de respostas como saída. Entretanto, em *Machine Learning* os dados e as respostas associadas aos dados são obtidos na entrada e as regras como saída (FRANCOIS, 2017).

2.2.3 Justificativa para uso de *Machine Learning*

”Se dados tivessem massa, a Terra seria um buraco negro”(MARS LAND, 2011) Com a tecnologia cada vez mais presente na vida do ser humano, a quantidade de dados produzida tem aumentado diariamente. Tarefas como analisar os milhares de vetores que representam o DNA ou detectar fraudes no cartão de crédito dentre os milhares clientes de um banco, tornam-se atividades que o cérebro humano é incapaz de armazenar ou processar.

É faculdade de *Machine Learning* aprender dos dados e encontrar relações inteligentes, podendo realizar **inferências** e **classificações** que só uma inteligência artificial poderia. Aprendizado de máquina é, portanto, uma subárea de IA.

2.2.4 Tipos

O ato de *aprender* implica em se tornar melhor em uma tarefa por meio da prática. Assim como seres humanos podem aprender de diversas formas, seja pela tentativa e erro, ou exaustivamente refinar uma técnica, existem diversas abordagens de aprendizado de máquina.

- **Aprendizado Supervisionado:** Um conjunto de treino com as repostas corretas (*targets*) é apresentado ao algoritmo que tentará generalizar e aprender a partir dos exemplos mostrados.
- **Aprendizado não Supervisionado:** As respostas corretas não são mostradas, em contrapartida o algoritmo tentará encontrar similaridades entre os objetos de entrada e tentará categorizá-los.
- **Aprendizado por reforço:** Está entre o aprendizado supervisionado e o não supervisionado. O algoritmo será avisado quando sua resposta estiver errada, mas não será dito como poderá melhorar, deverá explorar diferentes possibilidades até obter a resposta correta.
- **Aprendizado evolutivo:** A evolução biológica pode ser vista como um processo de aprendizado: os organismos se adaptam para melhorar suas chances de sobrevivência e ter seus descendentes adaptados ao meio ambiente. Similar à teoria de evolução de Darwin, algoritmos evolutivos serão modelados usando a ideia de sobrevivência dos mais qualificados.

Criando ambientes hostis para os menos qualificados, por meio de critérios de *fitness*, realizar mutação genética e realizar o *crossover* entre os indivíduos da população, farão parte das técnicas destes tipos de algoritmos.

- Aprendizado por transferência: Mais conhecido como *Transfer learning* é uma técnica de *Machine Learning*(ML) onde o conhecimento obtido durante um treino, poderá ser utilizado para resolver outros problemas similares.(SARKAR; BALI; GHOSH, 2018) Esta técnica permite a economia de recursos computacionais, ao reciclar modelos já treinados. Será uma das principais técnicas utilizadas neste trabalho.

2.2.5 *Machine Learning* e o reconhecimento de Imagens

O aprendizado de máquina para reconhecimento de imagens pode ser análogo a forma que humanos aprendem. Por exemplo, um pai ensinando seu filho a classificar um animal entre cachorro e gato. O pai mostrará um animal ao filho e pedirá que o classifique entre cão ou gato, mostrando a resposta certa quando a criança errar. Com a prática, haverá uma melhoria de desempenho e uma habilidade de **generalização** irá permitir que a criança possa classificar até mesmo exemplos nunca antes vistos.

Da mesma forma, um programa de *Machine Learning* poderá aprender de imagens, analisando características como formas, cores, texturas e proporções. A avaliação será feita a partir do seu poder de generalização e capacidade de classificar imagens nunca antes vistas, por meio de um processo de *recall* dos exemplos previamente vistos. (BRINK et al., 2017) A figura 2.1, mostra o processo de aprendizado de máquina, com o modelo sendo apresentado imagens já categorizadas, depois sendo posto a prova com imagens não antes vistas.

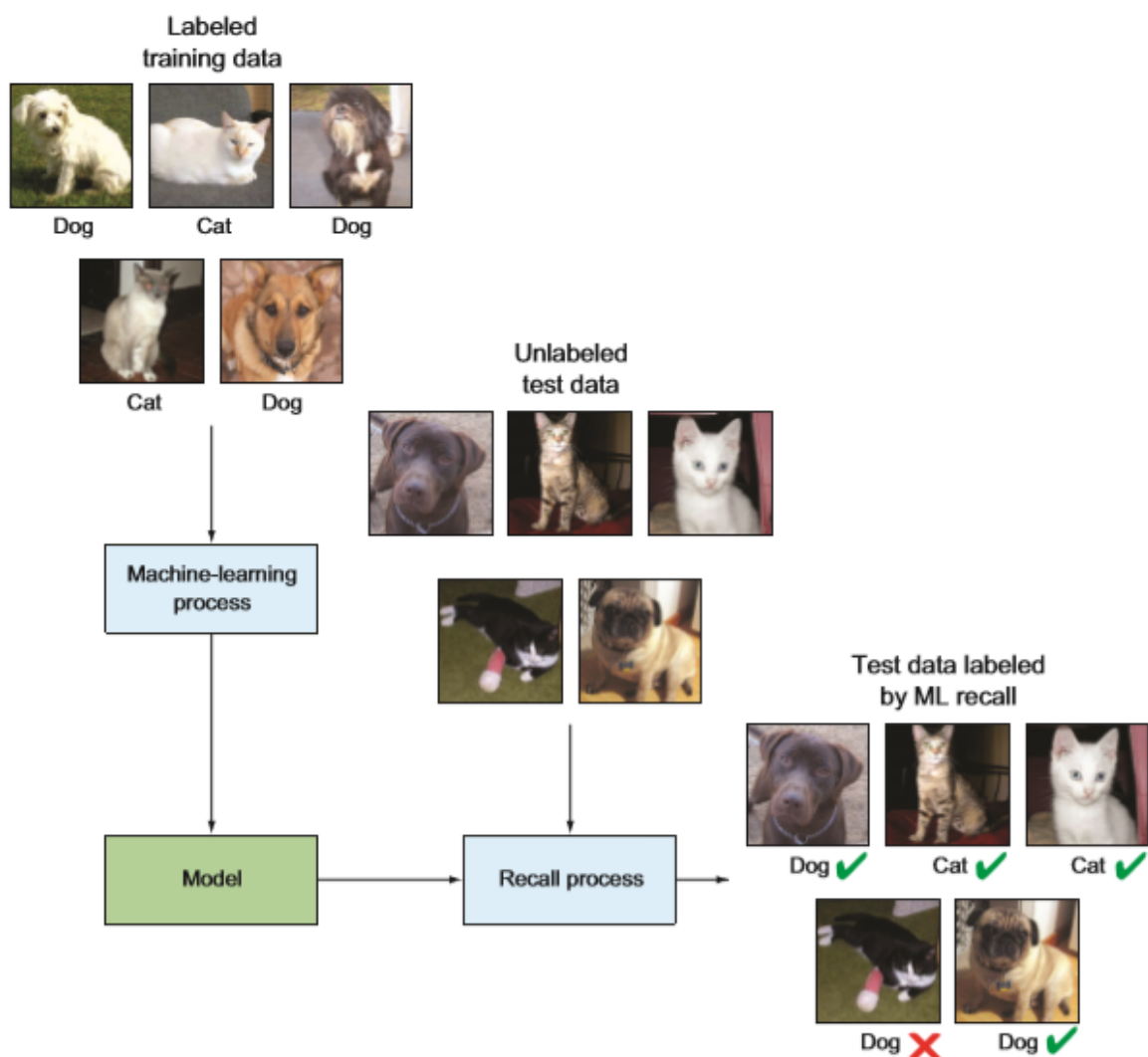


Figura 2.1: Processo de aprendizado de máquina
Fonte: (BRINK et al., 2017)

2.3 Visão Computacional

A Visão computacional é uma área de pesquisa e estudo da computação, que possui o objetivo de realizar a análise de forma automática, de imagens e vídeos, com o intuito de obter uma melhor compreensão do mundo. Surgiu por volta dos anos 60 e 70, com a ideia de ser um problema de fácil resolução. Entretanto, aparentou-se trivial, pois os seres humanos possuem o próprio sistema visual bem definido. (DAWSON-HOWE, 2014). Não se imaginava que a interpretação de imagens poderia ser feita por um computador, nem havia meios para realizar a captação de dados necessários com o poder de hardware da época.

Embora só recentemente, o estudo da área ser conduzido, é rápido seu desenvolvimento.

Pesquisadores em visão computacional tem desenvolvido, em paralelo, técnicas para reproduzir formas tridimensionais partir de imagens 2D. Os algoritmos *"Structure From Motion"* podem reconstruir um espaço 3D por meio de imagens planas (Figura 2.2a).

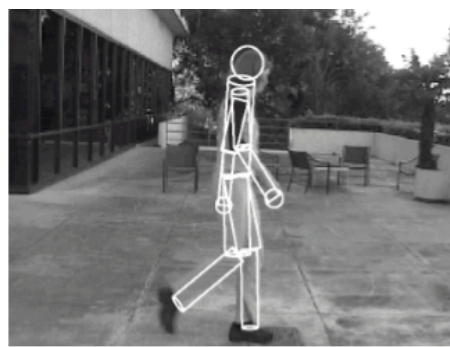
A Figura 2.2b mostra que também já é possível mapear o movimento de uma pessoa sob um plano de fundo complexo. Algoritmos *"Person tracking"* podem detectar o andar de uma pessoa dentre outros elementos do plano.

Por fim, mesmo sob moderado nível de confiança, é objeto de estudo da visão computacional e objetivo deste trabalho encontrar e classificar uma pessoa por meio de sua face. Algoritmos de Face detection podem localizar e reconhecer indivíduos em uma imagem. (Figura 2.3a) (SZELISK, 2011)



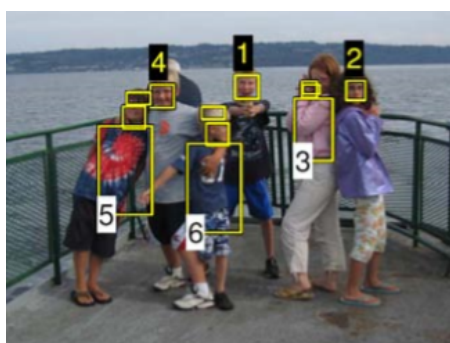
(a) Contrução de espaço 3D

Fonte: (SNAVELY N.; SZELISKI, 2006)



(b) Detecção do andar humano

Fonte: (SIDENBLADH H.; FLEET, 2000)



(a) Reconhecimento de indivíduos

Fonte: (SNAVELY N.; SZELISKI, 2006)

Quando o objetivo é **reconhecimento facial**, é preciso primeiro encontrar a localização e tamanho de qualquer face na imagem em análise. Em princípio, seria possível aplicar algoritmos de reconhecimento facial em qualquer pixel e escala, mas na prática, esse processo seria muito

lento. (MOGHADDAM; PENTLAND, 1997) Com a necessidade de economia de processamento computacional em sistemas embarcados, isto seria totalmente inviável.

2.4 *Deep Learning*

As aplicações de aprendizado de máquina eram limitadas pela falta de dados e de tecnologias capazes de processá-los de forma veloz e eficiente. Hoje, a grande quantidade de dados disponíveis e o avanço da computação, permitiram o desenvolvimento de algoritmos muito mais complexos e rápidos, que trouxeram um novo impulso para *Machine Learning*. É o caso de *Deep learning*.

2.4.1 Redes Neurais

Aprendizagem profunda normalmente envolve dezenas ou centenas de camadas representativas, a quantidade destas representa a profundidade do modelo, o que dá origem ao *deep* de *deep learning*. Todas essas camadas são aprendidas automaticamente quando expostas aos dados de treino. Essa forma de estruturação por camadas representativas é quase sempre aprendida por meio de modelos chamados **redes neurais**, constituídas de camadas sobrepostas.(FRANCOIS, 2017) O termo "rede neural" se refere à neurociência e é usual o termo "**neurônio**" ao componente matemático presente em toda estrutura de uma rede neural que calcula a soma ponderada de vários inputs, aplica uma função e passa o resultado adiante.

2.4.2 Conceitos de *Deep Learning*

Aprendizagem Profunda, ou *Deep Learning*, é a parte do aprendizado de máquina que, por meio de algoritmos de alto nível, imita a rede neural do cérebro humano. Para chegar nível de intelecto mais avançado, o princípio de redes neurais artificiais foi desenvolvido para suportar camadas discretas, conexões e direções de propagação de dados. Assim, os dados são submetidos a várias camadas de processamento não lineares que simulam a forma de pensar dos neurônios.

Deep learning é constituído de algoritmos complexos construídos, a partir de um empilhamento de diversas camadas de "neurônios", alimentados por quantidades imensas de dados, que são capazes de realizar tarefas como o reconhecimento de imagens e fala, processar a linguagem natural e aprender a realizar tarefas extremamente avançadas sem interferência humana. A

principal aplicação dos algoritmos de *Deep Learning* é a tarefa de classificação de entidades, em especial, **reconhecimento de imagens**.

Consequentemente, o interesse em aprendizagem profunda tem disparado, com cobertura constante na mídia popular. A pesquisa de aprendizagem profunda agora aparece rotineiramente em revistas, jornais e até no dia a dia do cidadão comum ao utilizar os serviços da *Google*, *Facebook* ou *Netflix*.

Para sintetizar "Deep Learning é um *framework* matemático para aprender representações de dados"(FRANCOIS, 2017)

2.4.3 Como funciona

A figura 2.4 mostra uma rede neural profunda para exemplificar como essas representações são aprendidas por algoritmos de aprendizado profundo. A imagem passará por diversas camadas, até o nível mais profundo da rede, a fim de reconhecer qual o dígito ali presente.

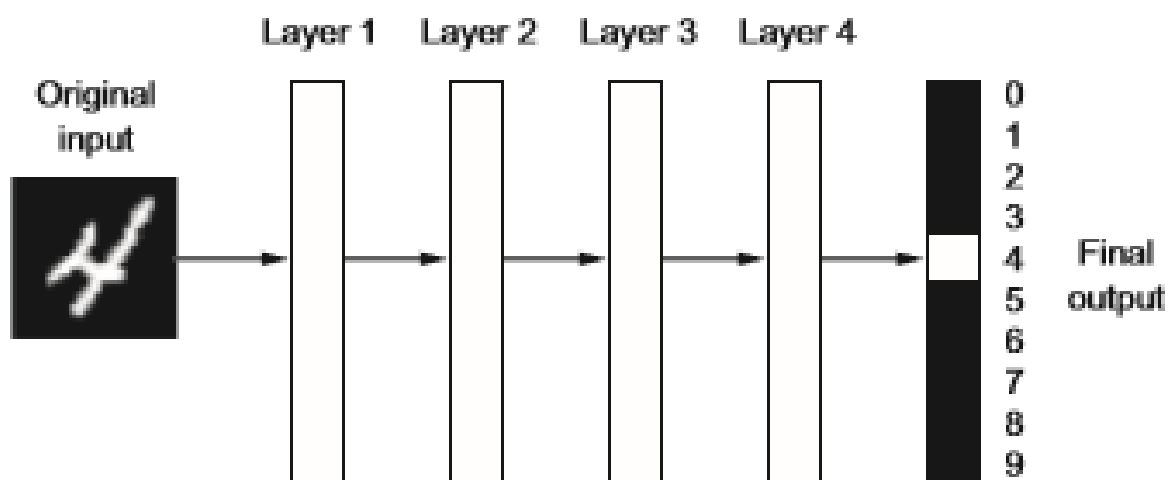


Figura 2.4: Uma rede neural profunda para classificação de dígitos
Fonte: (FRANCOIS, 2017)

Como pode ser visto na figura 2.5, ao mesmo tempo que a rede neural torna a imagem do dígito cada vez mais diferente da original também a torna mais informativa para obter o resultado final.

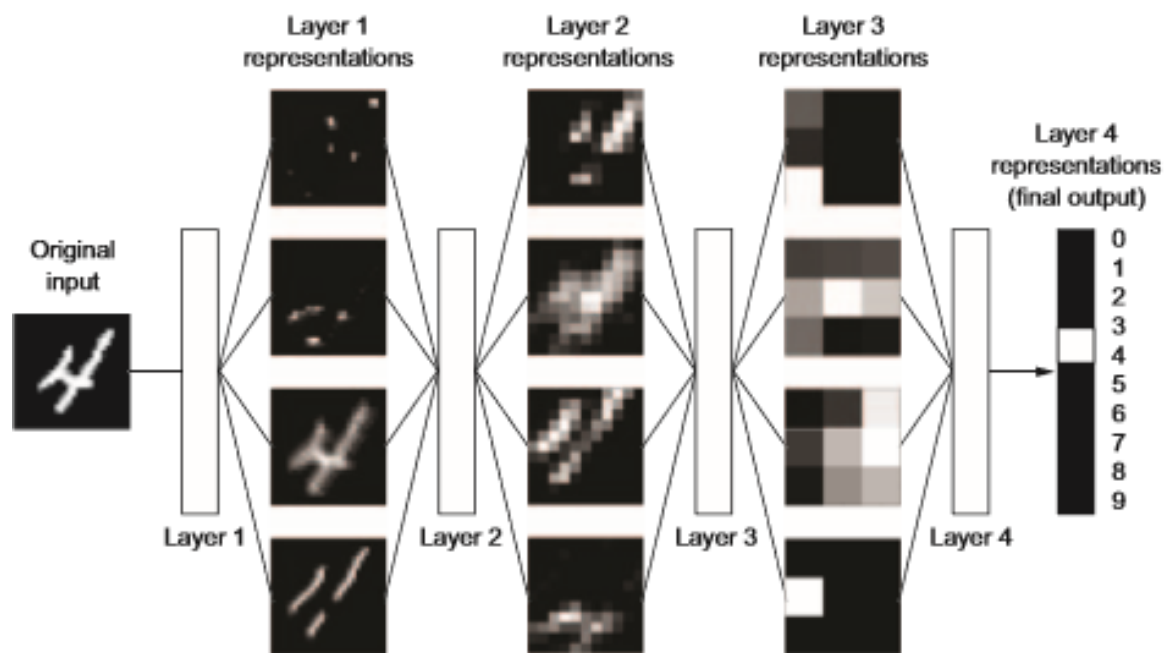


Figura 2.5: Representações profundas aprendidas do modelo de classificação de dígitos.
Fonte: (FRANCOIS, 2017)

Uma rede neural profunda, pode ser associada a um processo de "destilação de dados", onde a informação (neste trabalho a imagem) irá passar por sucessivos filtros que no final trarão uma representação "pura" que se finaliza com a classificação final da imagem.

É o objetivo final desta técnica mapear a entrada (*inputs*) a categorias (*labels*) por meio de um aprendizado consequente da observação de **dados iniciais** que determinam camadas que compõe uma rede neural profunda. A especificação do que cada camada faz com os dados de entrada é armazenado como o *peso* da camada. Em termos técnicos pode-se dizer que as transformações nos dados de entrada feitas por cada camada são parametrizadas por seus pesos como mostrados na figura 2.6. Nesse contexto o *aprendizado* se dá em encontrar o conjunto de valores dos pesos de todas as camadas da rede de forma que conseguirá mapear corretamente um conjunto de valores de entrada as suas respectivas categoriais.

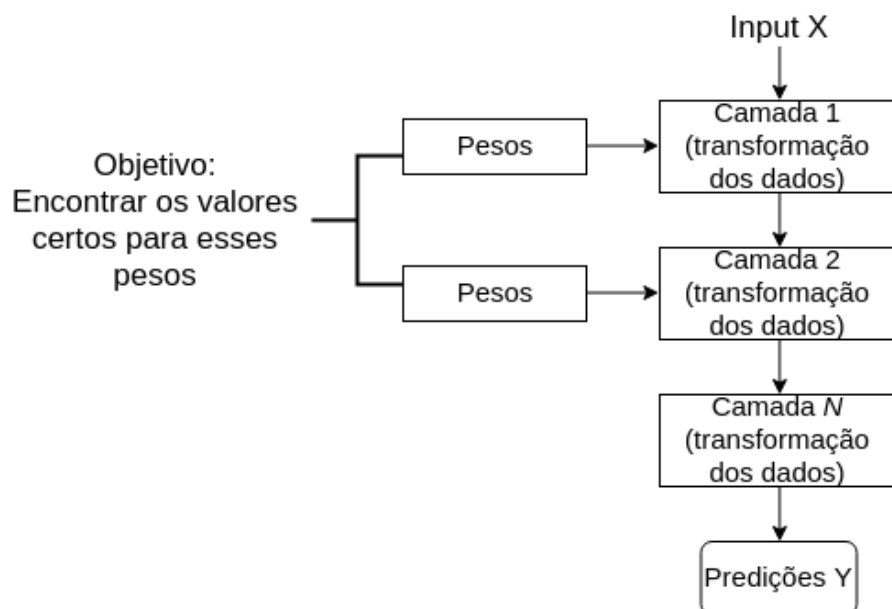


Figura 2.6: Representação do aprendizado pelo ajuste dos pesos
Fonte: Própria

Em cada iteração do processo de aprendizado é medido o quão longe a saída inferida está da saída esperada por meio da **função de perda** da rede. A função de perda calcula a distância das predições feitas em relação as saídas reais, determinando assim, o "quão bem" a rede treinada está, isto é o ponto de perda ou *score* da rede. A figura 2.7 mostra esse processo.

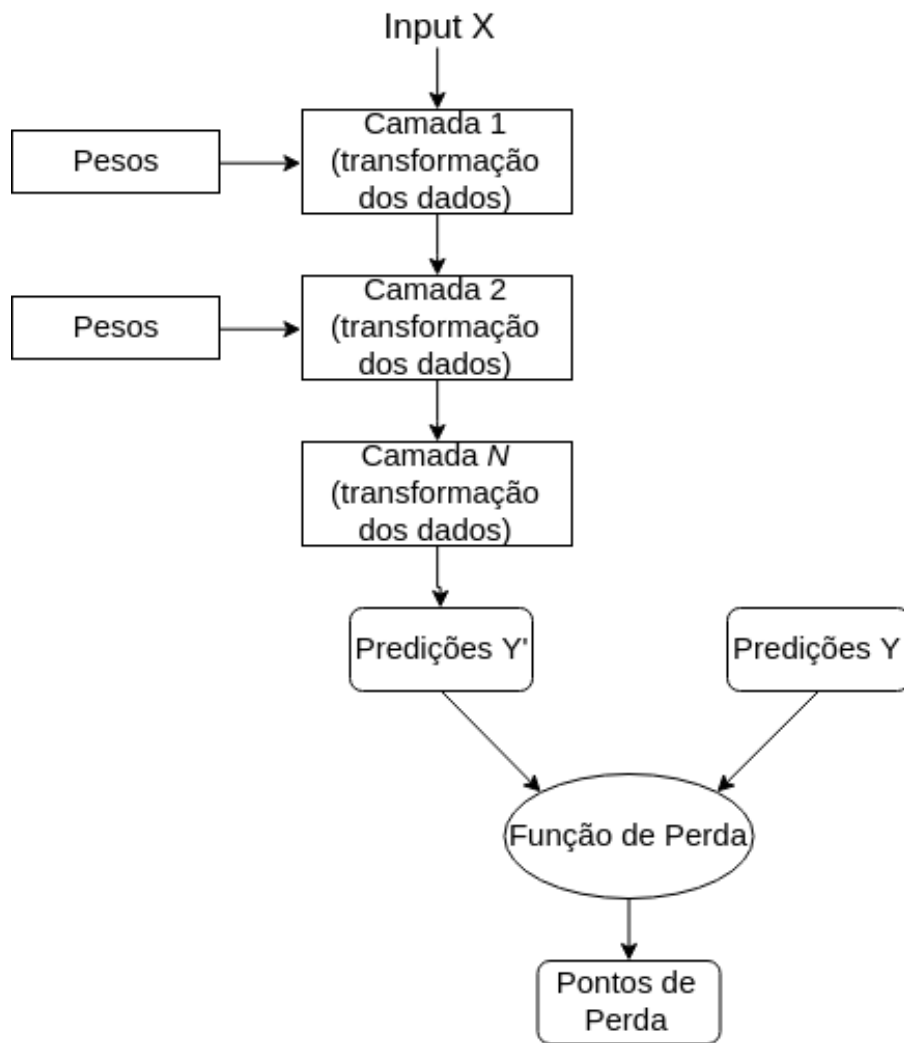


Figura 2.7: O papel da função de perda no aprendizado
Fonte: Própria

O ponto fundamental de *deep learning* está em utilizar os pontos de perda como sinal de *feedback*, para o ajuste dos valores dos pesos das camadas de forma a minimizar os pontos de perda. Este ajuste é feito por um **otimizador** que implementa um algoritmo chamado de *Backpropagation*: o algoritmo central de *deep learning*, que cuidará de atualizar os pesos de cada camada.

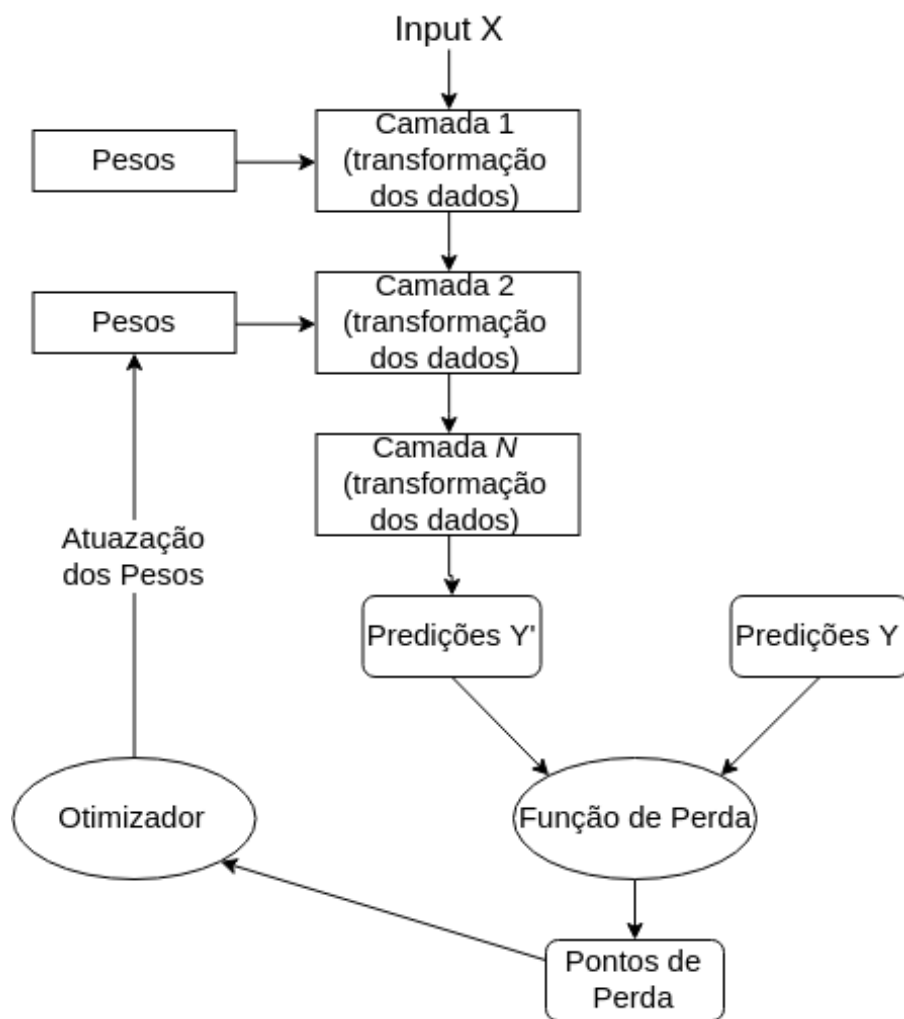


Figura 2.8: O papel do otimizador no aprendizado

Fonte: Própria

O processo todo é feito em um **loop de treino** onde em cada iteração são ajustados os pesos de forma a minimizar os pontos de perda. O resultado final é um grafo de camadas com os pesos ajustados para resolução de um problema para um conjunto de entrada de tipo específico. Esse **grafo treinado** que será utilizado para realizar o reconhecimento facial deste trabalho.

2.5 *Deep Learning X Machine Learning*

Conforme mostrado na figura 2.9 observa-se as áreas que compõe inteligência artificial, onde *Deep Learning* é uma subárea de *Machine Learning* que por sua vez são subáreas de Inteligência Artificial.

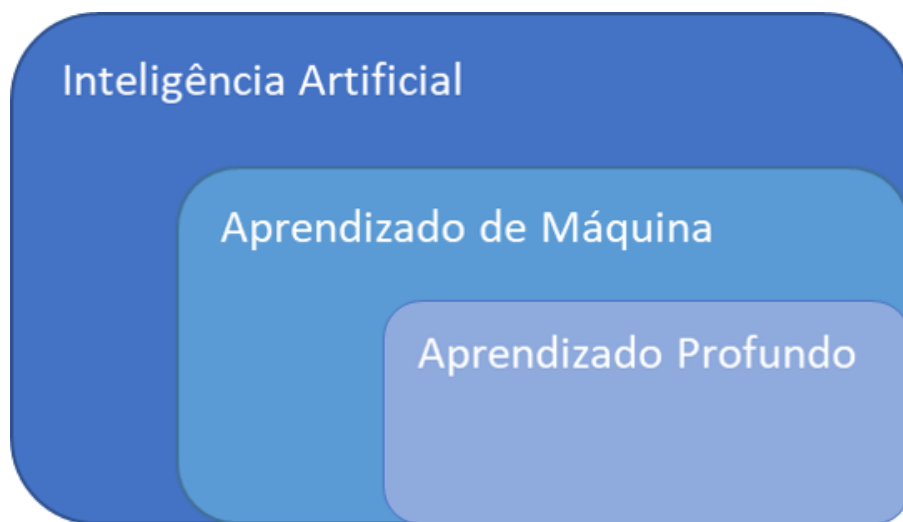


Figura 2.9: Hierarquia de aprendizado
Fonte: Própria (2019)

É comum a confusão dos termos e a delimitação de cada área de estudo. Sem as atuais máquinas poderosas *deep learning* não existiria. É preciso, ao menos atualmente, da máquina, para ocorrer o processo de deep learning. O aprendizado profundo é, portanto, um subcampo do aprendizado de máquina com este último permitindo uma maior generalização de definições e técnicas. A tabela 2.1 deixa evidente o relacionamento e foco das áreas.

Tabela 2.1: ML x DL

	Machine Learning	Deep Learning
O que é	Ciência de fazer com que computadores realizem ações, sem precisarem ser programados para tal	Uma técnica mais sofisticada de <i>machine learning</i> , construída a partir do princípio de redes neurais.
Como funciona	São fornecido dados iniciais aos algoritmos que irão aprender, por conta própria, para fazer previsões e inferências a partir de modelos.	São capazes de suportar e trabalhar com big data e funcionam como uma mente própria.
Quando surgiu	Começou nos anos 70, como a primeira forma de colocar em práticas os conceitos de inteligência artificial.	Desenvolveu-se a partir de 2010 com o surgimento de computadores poderosos e o aumento dos dados acessíveis, tornando possível o processamento de grandes quantidades de dados.

2.6 Deep Learning X Sistemas Embarcados

É fato que *deep learning* só foi possível com o avanço tecnológico, notadamente, a partir dos anos 2000. Treinar redes profundas altamente precisas com poderes de inferência superiores ao cérebro humano, se tornou algo trivial com os atuais computadores. A Engenharia da Computação, ao mesclar engenharia elétrica e engenharia de software propiciou o aparecimento de sistemas embarcados, que são, em sua maioria, portáteis, pequenos e podem realizar tarefas específicas, similares a um computador. No entanto, modelos normalmente utilizados de maneira eficiente por um computador, podem não ser facilmente processado por um sistema embarcado, muito menos ter aplicabilidade em sistemas em tempo real como de monitoramento e de transmissão instantânea.

Google foi pioneiro ao apresentar uma classe de modelos eficientes chamados *MobileNets* própria para dispositivos móveis e sistemas de visão computacional embarcados. Essa classe de modelo realiza convoluções por camadas de profundidade permitindo criar uma rede neural profunda leve onde traz um menor custo computacional ao realizar o processo de classificação de imagens. É dessa forma realizada um eficiente balanceamento entre latência e precisão.

O método de *transfer learning* pode ser utilizado para criar redes neurais profundas leves e altamente precisas.

2.6.1 *Transfer Learning*

A principal técnica utilizada neste trabalho foi a Transferência de aprendizado - *Transfer Learning* que é a reutilização de um modelo pré-treinado em um novo problema. Trata-se de utilizar uma rede neural treinada em outro outro conjunto de dados, geralmente maior, para resolver um novo problema. Diversos são os motivos para utilizar deste recurso, mas em suma, o motivo é a redução do custo computacional em relação aos métodos de treino tradicionais:

- Já que é difícil obter um conjunto de dados grande o suficiente, raramente CNNs são treinadas do zero;
- CNN muito profundas são muito caras para serem treinadas e o processo de inferência com esses modelos também demoram mais.

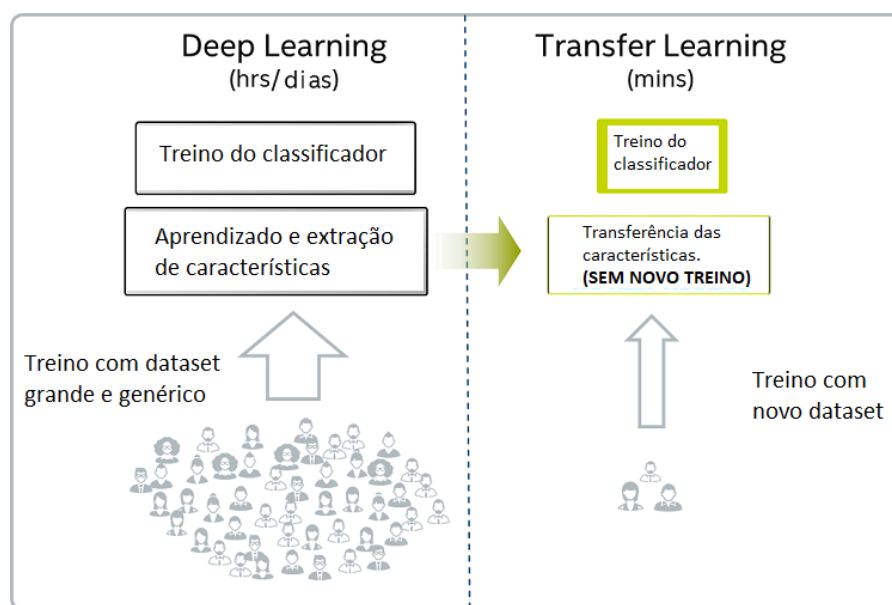


Figura 2.10: Treino Tradicional X *Transfer Learning*
Própria (2019)

A figura 2.10 mostra como funciona o método de *Transfer Learning* em relação aos modelos tradicionais. *Deep learning* tradicional poderá levar horas ou até mesmo dias para gerar um

modelo enquanto a técnica de *Transfer Learning* permitirá que, às vezes em questão de minutos, seja obtido um classificador eficiente.

2.7 Detecção Facial

Como falado em (RANJAN et al., 2018), existem três módulos necessários para definir um sistema de reconhecimento facial. Primeiro, um **detector facial**, usado para localizar as faces em imagens e vídeos. Segundo, por meio de um mapeamento facial, para normalização da face. Terceiro, um módulo de reconhecimento facial para classificação. Este artigo aborda o módulo de detecção facial.

Foram analisados quatro populares algoritmos para detecção facial, maximizadas suas performances e como artefato final um algoritmo que os combina em um detector mais robusto, com menor taxa de erros.

A detecção de face forma parte do processo de reconhecimento facial usado como parte essencial no reconhecimento de pessoas. Os métodos para detecção facial focam em um conjunto de recursos computacionais ao realizar a análise de uma imagem com o objetivo de determinar se há algum rosto ou não.

Estes métodos podem ser complicados devido à variabilidade de características presentes em rostos humanos e pela quantidade de variáveis que podem influir no reconhecimento de um rosto. Dentre os atributos que podem-se citar a pose, expressão, posição e orientação do rosto, outros como a cor da pele, a presença de óculos ou pelos faciais, e um terceiro grupo se relaciona com as características da imagem: diferenças no ganho da câmera, condições de iluminação da cena e resolução da imagem.

Um momento transcendente na detecção de rosto aconteceu com os trabalhos de Viola-Jones, eles apresentaram o sistema *Real-Time Face Detector* (VIOLA; JONES, 2004) capaz de detetar os rostos em tempo real com alta precisão. A utilização de técnicas tais como imagem integral e cascata de atenção tornaram o algoritmo de Viola-Jone altamente eficiente. Em (VIOLA; JONES, 2004) é apresentado um estudo detalhado dos dos algoritmos empregados.

Os métodos de detecção de face podem ser utilizados em varias circunstancias: em imagens

representadas com escala de cinza, em imagens coloridas e em vídeos; são identificados quatro tipos de métodos: **baseado em conhecimento, baseado em características, usando correspondência entre padrões e baseado na aparência** (YANG; KRIEGMAN; AHUJA, 2002)).

Com a observação que os detectores faciais do *Dlib* possuem um menor capacidade de detecção em faces menores em uma imagem, foram utilizados em conjunto os detectores do OpenCV os quais são Haar Cascade e DNN.

2.7.1 *Haar Cascade*

Já se sabe que a detecção facial em humanos é feita quando há correspondência de características com atributos previamente definidos como grandes contribuidores de informação que se comportam como "máscaras" os quais os atributos *Haar* da face correspondem. Presentes nos dados de treino, ocorre uma aproximação das características sendo possível definir o número de vizinhos mínimos para ser considerado de fato uma face. Estes atributos podem ser visualizadas no esquema da figura 2.11:

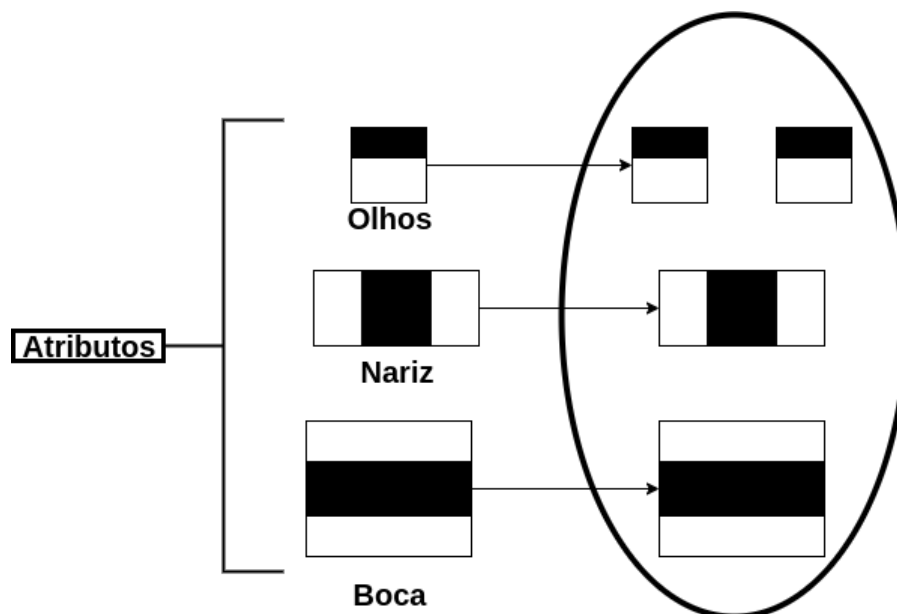


Figura 2.11: Características Haar
Fonte: Própria (2019)

A figura 2.12 mostra os passos para detecção de faces em imagens e como se dá o relacio-

namento entre os atributos *Haar*.

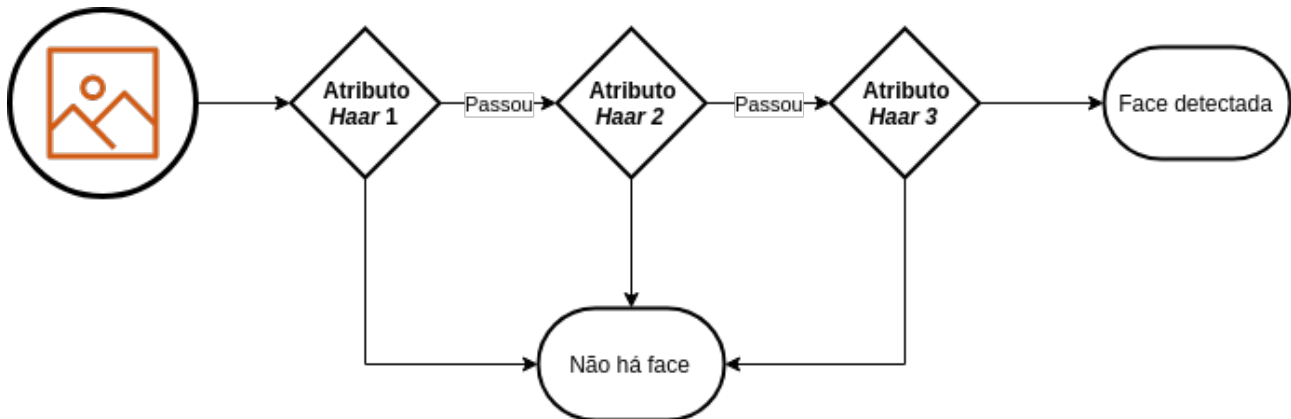


Figura 2.12: Fluxo *Haar Cascade*
Fonte: Própria (2019)

Isso significa que em *Haar Cascade* assim que o algoritmo não identifica uma das características necessárias de uma face, já é retornado um falso positivo no quadrante em análise, dessa forma, reduzindo seu custo computacional.

2.7.2 *Deep Learning Model*

São infinitas as possibilidades de modelos treinados utilizando-se de *Deep Learning*. O algoritmo feito neste trabalho permite que sejam utilizados modelos dos dois mais populares frameworks *Caffe* e *Tensorflow*. Tendo em vista a necessidade de baixo custo computacional em sistemas embarcados, o foco foi no modelo utilizando-se do framework *tensorflow* o mais novo até a presente data [OpenCV 2019]. O grafo possui 237 nós e a imagem de entrada passará por diversos filtros até uma camada de saída que determinará, se presente, as coordenadas da face na imagem analisada. A arquitetura completa da rede foi analisada com a ferramenta *tensorboard* e se encontra como anexo deste trabalho.[TensorboardModel 2019]

2.8 Tecnologias

Foram utilizadas diversas tecnologias para a elaboração deste projeto com ênfase em confiabilidade e custo computacional.

2.8.1 PostgreSQL

Um poderoso Sistema de Gerenciamento de Banco de Dados (SGBD) objeto-relacional de código fonte aberto que utiliza e estende a linguagem SQL combinada com outros recursos que armazenam e dimensionam de forma segura as cargas de trabalho de dados mais complexos. O sistema surgiu por volta do ano de 1986 como parte do projeto POSTGRES da Universidade da Califórnia em Berkeley e possui mais de 30 anos de desenvolvimento ativo na plataforma central. PostgreSQL possui forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software para fornecer soluções eficazes e inovadoras de maneira consistente.

2.8.2 Flask

Lançado em 2010 e desenvolvido por Armin Ronacher, Flask é um *Framework Web Python* de alto nível, gratuito e de código aberto que incentiva o rápido desenvolvimento de aplicações web. Possui um núcleo simples e expansível que permite que um projeto possua apenas os recursos necessários para sua execução (conforme surja a necessidade, um novo pacote pode ser adicionado para incrementar as funcionalidades da aplicação). Dentre as principais características do Flask, podem-se citar:

- Simplicidade: Por possuir apenas o necessário para o desenvolvimento de uma aplicação, um projeto escrito com Flask é mais simples se comparado aos frameworks maiores, já que a quantidade de arquivos é muito menor e sua arquitetura é muito mais simples.
- Rapidez no desenvolvimento: Com o Flask, o desenvolvedor se preocupa em apenas desenvolver o necessário para um projeto, sem a necessidade de realizar configurações que muitas vezes não são utilizadas.
- Projetos menores: Por possuir uma arquitetura muito simples (um único arquivo inicial) os projetos escritos em Flask tendem a ser menores e mais leves se comparados a frameworks maiores.

- Aplicações robustas: Apesar de ser um micro-framework, o Flask permite a criação de aplicações robustas, já que é totalmente personalizável, permitindo, caso necessário, a criação de uma arquitetura mais definida.

2.8.3 uWSGI

uWSGI é um servidor de container de aplicação feito em C puro, rápido, auto-ajustável e amigável a desenvolvedores e administradores de sistema.

O servidores de aplicação para várias linguagens de programação, protocolos, proxies e gerenciadores de processo poderão ser implementados utilizando uma api comum compartilhando das mesmas configurações.

Versatilidade, performance, baixo uso dos recursos computacionais e confiança são as principais diretrizes que regem este serviço. São componentes deste servidor:

- O núcleo: implementa configuração, gerenciadores de processo, criação de sockets, monitoramento, registros de logs, gerenciamento de memória,etc.
- Plugins de requisição: implementa as interfaces da aplicação para várias linguagens de programação e plataformas.
- *Gataways*: implementa balanceadores de carga, proxies e rotas.
- O imperador: implementa o gerenciamento e monitoramento massivo de instâncias de aplicações.
- *Loop Engine*: implementa controle de eventos e concorrência, threads, rotinas,etc.

2.8.4 NGINX

O Nginx, que se pronuncia “engine-ex”, consiste num servidor HTTP e *reverse-proxy* de alto desempenho, gratuito e de código aberto. o NGINX é amplamente conhecido pelo seu alto desempenho, estabilidade, riqueza em conjunto de recursos e configuração simples associados a um baixo consumo de recursos. Ao contrário de servidores tradicionais, o NGINX não depende

de *threads* para lidar com as solicitações, ao invés disso, ele utiliza uma arquitetura orientada a eventos que é muito mais escalável, ou seja, assíncronos. Essa modelo de arquitetura utiliza pequenas quantidades previsíveis de memória sob carga.

Algumas empresas de grande reputação que usam o Nginx incluem Autodesk, Atlassian, Intuit, T-Mobile, GitLab, DuckDuckGo, Microsoft, IBM, Google, Adobe, Salesforce, VMWare, Xerox, LinkedIn, Cisco, Facebook, Target, Citrix Systems, Twitter, Apple, Intel e muitos mais (fonte).

O Nginx foi criado originalmente por Igor Sysoev, com seu primeiro lançamento público em outubro de 2004. Igor inicialmente concebeu esse software como uma resposta ao problema C10k, um problema relacionado com a dificuldade de desempenho em lidar com 10,000 conexões simultâneas.

Como as suas raízes estão na otimização de desempenho em escala, o Nginx normalmente supera outros servidores populares web em testes de benchmark, particularmente em situações com conteúdo estático e/ou elevadas solicitações simultâneas.

Devido a sua maior elasticidade e menor custo computacional foi escolhido para este trabalho no lugar do popular *Apache*, este por sua vez, mesmo tendo suporte a *.htaccess* (ou "arquivos de configuração distribuída"), oferecendo um meio de fazer mudanças nas configurações por diretório fazem com que *Apache* consuma muito mais recursos do servidor quando vai entregar qualquer requisição.

2.8.5 Raspberry PI

Apesar do tamanho diminuto e das feições pouco convencionais, o Raspberry Pi é um computador como outro qualquer. Isso quer dizer que ele pode servir para navegação na internet, reprodução de conteúdo multimídia, criação de conteúdo em forma de texto, planilhas e imagens e, é claro, inteligência artificial. Desenvolvido no Reino Unido pela Fundação Raspberry Pi, os modelos custam entre *US\$ 25* e *US\$ 35*. Tendo em vista a limitação dos recursos computacionais foram inseridos parâmetros de controle que poderão definir em que ponto utilizar qualquer algoritmo deste trabalho, a resolução de entrada da imagem, canais de cores, etc. Dessa forma

tendo um algoritmo escalável e personalizável conforme o propósito da aplicação.

2.8.6 OpenCV

OpenCV - *Open Source Computer Vision* é uma biblioteca aberta de visão computacional que vai permitir captar e manipular as imagens com python. Possui também bibliotecas prontas para detecção facial: Neste trabalho foram explorados duas funcionalidades de OpenCV para detecção, a primeira classificador ***Haar cascade*** que é um método de detecção de objetos proposto por Paul Viola e Michael Jones no artigo "*Rapid Object Detection using a Boosted Cascade of Simple Features*" em 2001. A partir da versão OpenCV 3.3 foi embutido um novo detector facial que utiliza *deep learning* podendo utilizar modelos de diversos *frameworks* como *Caffe*, *TensorFlow* e *Torch/PyTorch*.

2.8.7 Dlib

Dlib é uma biblioteca de software multi-plataforma de propósito geral escrita na linguagem de programação C++. Seu design é fortemente influenciado por ideias de design por contrato e engenharia de software baseada em componentes. Assim, é, antes de tudo, um conjunto de componentes de software independentes. Não é dedicada à área de visão computacional mas é amplamente utilizada nesse setor por ser *open-source* e possuir kits para trabalhar com redes, *threads*, interfaces gráficas, estruturas de dados, álgebra linear, ***machine learning***, ***processamento de imagem***, *data mining*, XML e processamento de texto, dentre outros.

OpenCV e *Dlib* dispõem de funções que utilizam os algoritmos de detecção conforme a tabela 2.2.

OpenCV	Dlib
Haar Cascade Face Detector	HoG Face Detector
Deep Learning based Face Detector(DNN), usando Single-Shot-Multibox detector e a arquitetura ResNet-10.	Deep Learning based Face Detector, usando Maximum-Margin Object Detector (MMOD)

Tabela 2.2: Bibliotecas de detecção facial

Capítulo 3

Desenvolvimento

Neste capítulo serão abordados os aspectos estruturais do sistema proposto, como modelagem, características e seu funcionamento. Para uma melhor visualização e compreensão entre os componentes do sistema, diagramas de UML e modelo entidade relacionamento foram elaborados e detalhados durante o capítulo. Além da realização do processo de engenharia de requisitos para se entender com mais clareza as funcionalidades essenciais do projeto.

3.1 Descrição Geral do Sistema

Foi definido como um Sistema Reconhecimento facial o qual será empregado tecnologias atuais com a proposta de prover uma API em forma que responda a solicitações de outros sistemas. Por meio deste será possível realizar treinamento de arquiteturas de redes neurais profundas e requisitar inferência de novas faces para reconhecimento. Foi também desenvolvido um módulo cliente, que poderá ser executado em qualquer sistema operacional baseado em *Debian* que é o caso do *Raspbian*, o sistema operacional do *Raspberry PI*.

A principal funcionalidade do software é o reconhecimento facial, a qual será dividido em 3 etapas: detecção facial, processamento da imagem e reconhecimento facial. Foi feito um módulo para treinamento de redes neurais profundas juntamente com uma ferramenta para coleta do *Dataset*.

3.2 Especificação do Sistema

Nesta etapa, será realizada a modelagem de sistema que consiste no processo de elaboração de modelos abstratos do sistema a ser desenvolvido. Cada modelo pretende apresentar uma visão ou perspectiva, diferente do sistema. Como forma de representação do sistema em notação gráfica será utilizado a UML (Linguagem de Modelagem Unificada, do inglês Unified Modeling Language) , tendo como papel auxiliar a visualizar o desenho e a comunicação entre objetos do sistema, bem como as principais funcionalidades.

3.2.1 Levantamento de Requisitos

O processo de análise de requisitos também faz parte dessa primeira fase. Um requisito funcional são declarações de funções ou características que devem ser implementadas no sistema. Ao passo que os requisitos não funcionais são aqueles que definem uma restrição, que pode estar relacionado às propriedades emergentes do software que está sendo desenvolvido ou ao processo de desenvolvimento, ou de um comportamento esperado que se aplica ao sistema.

Requisito Funcional

Identificação	Descrição
[RF01]	Permitir que usuários possam visualizar em tempo real o status do reconhecimento.
[RF02]	Dispôr de detectores inteligentes para captura de Dados.
[RF03]	Permitir que uma pessoa efetue o cadastro dos de pessoas conhecidas.
[RF04]	Permitir que uma pessoa se cadastre no dispositivo embarcado
[RF05]	Permitir a requisição de informações de reconhecimento facial
[RF06]	Permitir a requisição de novo treino para geração de novo modelo com DL
[RF07]	Permitir que usuários após efetuarem uma solicitação acompanhem o status da mesma.
[RF08]	Dispor de ferramenta para coleta de dataset

Requisito Não Funcional

Tabela 3.1: Requisitos Não Funcionais

Identificação	Descrição
[RNF01]	O sistema deverá estar em padrão de código PEP8.
[RNF02]	O sistema deverá estar estruturado em componentes de software.
[RNF03]	O sistema deve garantir a integridade dos dados trafegados entre o cliente e o servidor, tomando medidas para proteger contra ameaças de Man in the Middle (MITM).
[RNF04]	O sistema deverá ter um gerenciamento dos recursos computacionais.
[RNF05]	O sistema web deve zelar por um baixo custo computacional.
[RNF06]	Os dados não serão modificados sem a intervenção do usuário. Os dados mostrados sempre estarão corretos de acordo com os dados inseridos previamente no sistema.
[RNF07]	Tempo de inferência compatível com <i>Real-Time Computing</i> .
[RNF08]	O sistema deverá notificar os erros ao usuário em linguagem natural.
[RNF09]	Deverá dispor de parâmetros facilmente acessíveis e configuráveis
[RNF10]	O sistema deve ter a capacidade de recuperar o último estado no caso de ocorrência de erros.
[RNF11]	Todos os formulários de entrada necessitam ser validadas frente a um conjunto de entradas aceitáveis, antes do software aceita-los para processamento

Regras de Negócio

Identificação	Descrição
[RN01]	O sistema deverá ser desenvolvido em <i>Python3</i> .
[RN02]	O servidor para treino deverá ser executável na plataforma web.
[RN03]	Todos os componentes deverão rodar em qualquer ambiente <i>Debian</i> .
[RN04]	Toda biblioteca utilizada deverá ser de licença aberta.
[RN05]	Todo código deverá estar em inglês.

3.2.2 Diagrama de Atividades

O diagrama da figura 3.1 mostra o diagrama de atividades simplificado do desenvolvimento. Basicamente três ações constituem a execução do sistema.

1. Etapa para cadastro de novas classes. Que criará o *Dataset* para treino de novos modelos.
2. Módulo para treino de modelos de IA. Esse módulo estará presente no servidor dedicada para treino de modelos com *hardware* apropriado para essa tarefa.
3. Módulo para reconhecimento de pessoas com tratamentos de imagem presente.

A estrutura simplificada dos módulos mostra o que foi executado neste trabalho, embora a parte de treino seja fundamental para o funcionamento correto do reconhecimento, este é ferramenta a parte deste trabalho e não compõe objeto de estudo dessa monografia. Dessa forma serão analisados apenas os componentes que estão presentes no dispositivo foco deste trabalho, o *Raspberry PI*.

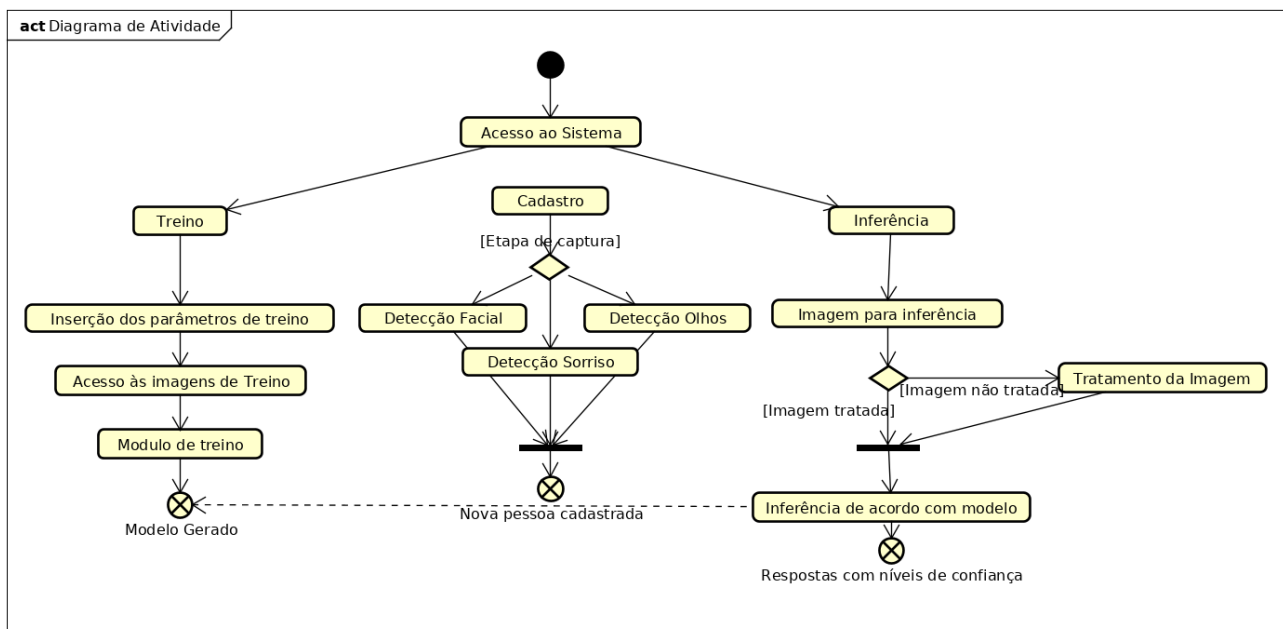


Figura 3.1: Diagrama de atividades

3.2.3 Diagrama de Componentes

Os componentes do sistema podem ser visualizados na figura 3.2.

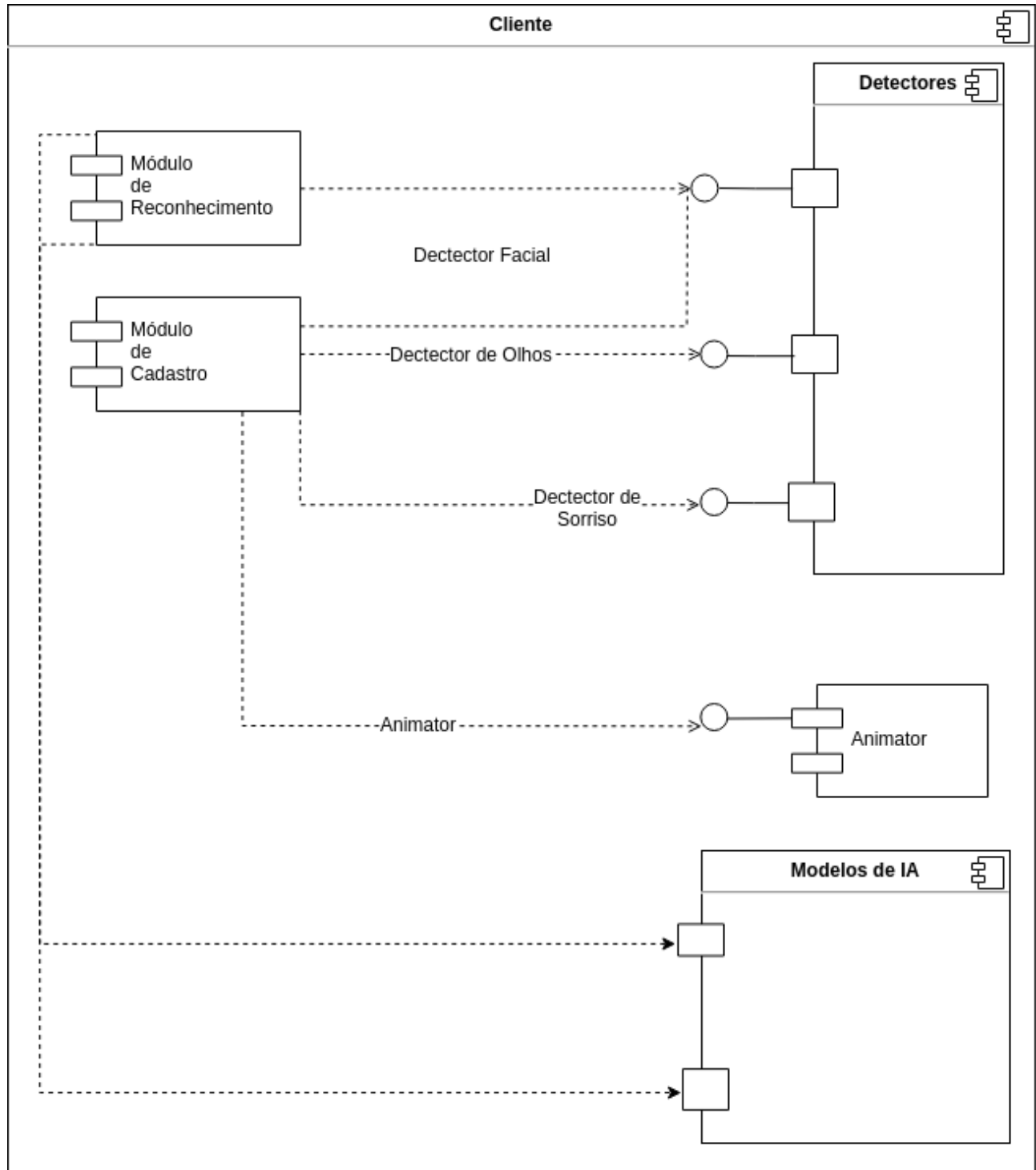


Figura 3.2: Diagrama de componentes

A especificação dos componentes dos *detetores* e *Modelos de IA* podem ser vistos, respecti-

vamente, nas figuras 3.3 e 3.4.

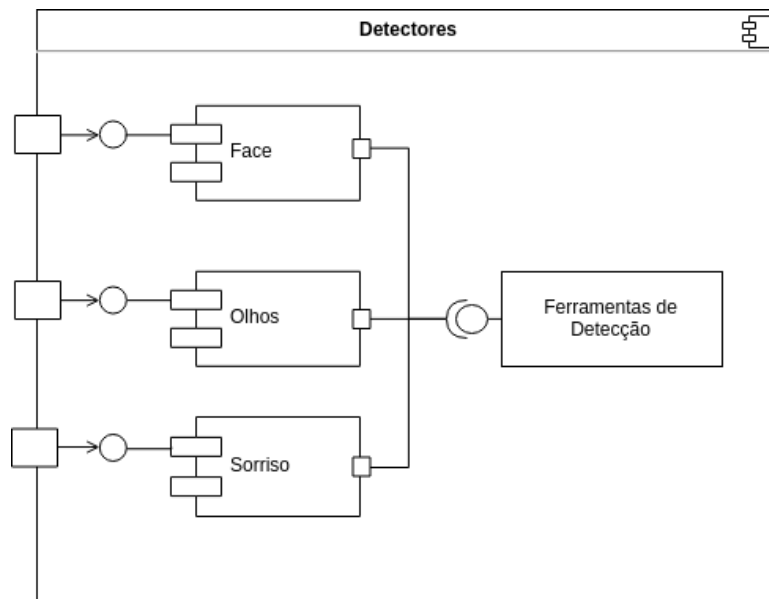


Figura 3.3: Componente de detecção facial

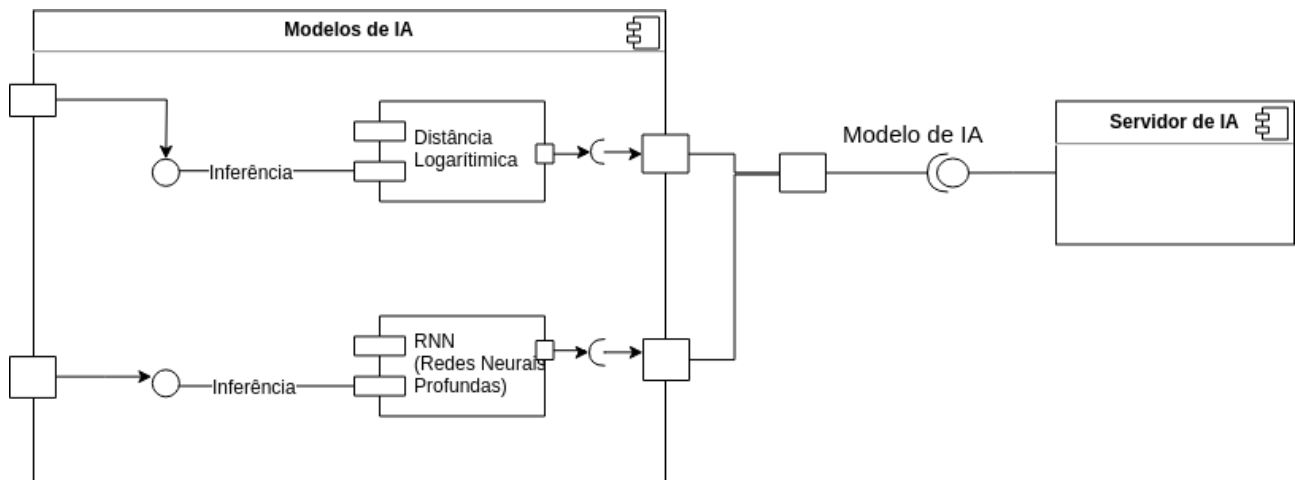


Figura 3.4: Componente de IA

Todos os componentes desenvolvidos podem ser visualizados no diagrama da figura 3.5

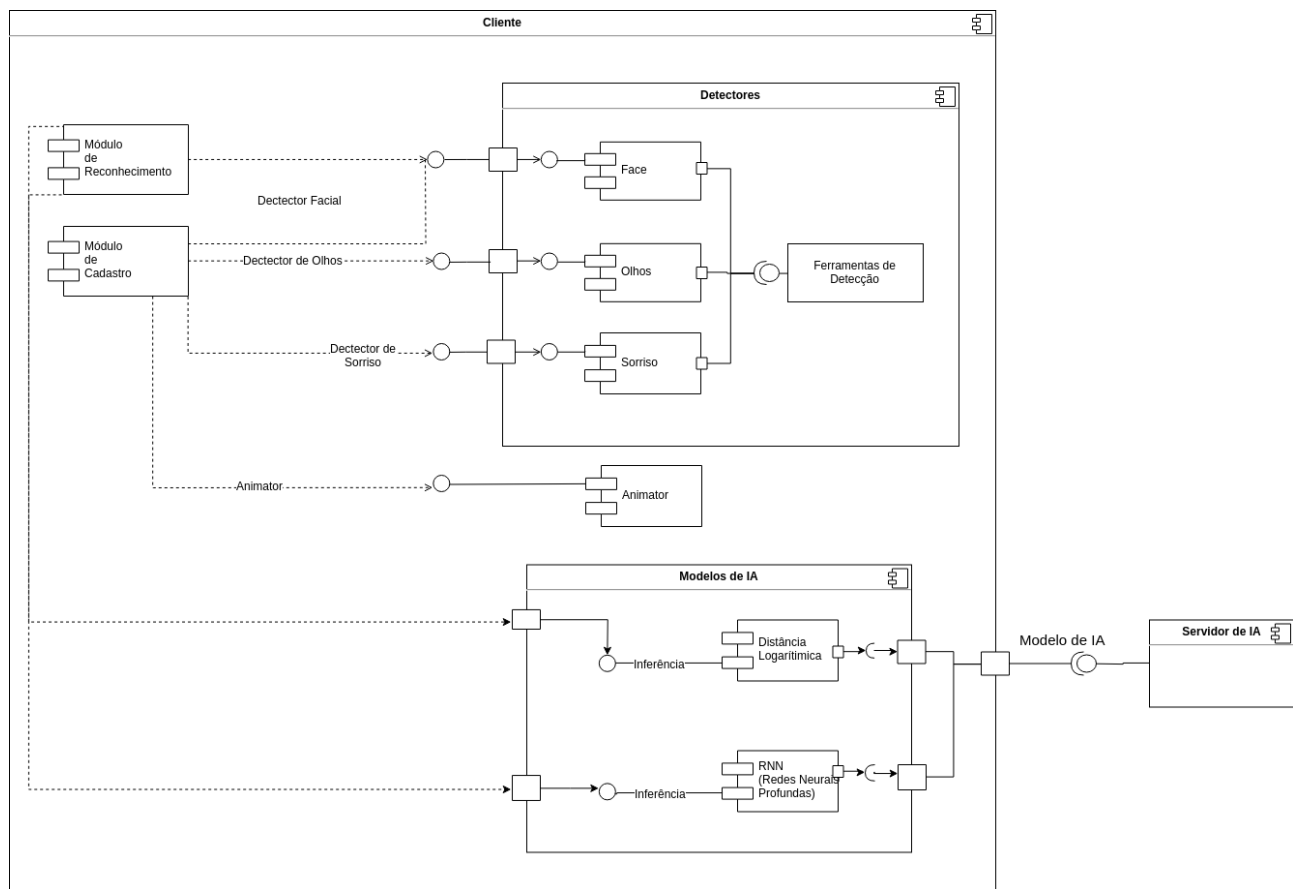


Figura 3.5: Diagrama de componentes completo

3.3 Detectores Faciais

Como mostrado na figura 3.3 este projeto inclui detectores inteligentes para, durante a etapa de cadastro ocorrer, sequencialmente:

1. Detecção de face
2. Detecção de olhos
3. Detecção de sorriso

Embora durante a etapa de reconhecimento seja necessário apenas o reconhecimento facial, observou-se que muitas vezes os algoritmos atuais são incapazes de inferir, em diferentes situações de luminosidade, distância e oclusão, apropriadamente a presença facial, tendo diferentes performances conforme o algoritmo. Dessa forma incorporados em um só algoritmo as técnicas

de detecção facial do *OpenCV*. Onde as imagens passariam pelos algoritmos de *DNN* e *Haar Cascade* seriam processadas e removidas possíveis duplicadas de detecção.

O algoritmo deste trabalho cuminou em uma classe *Detector* que deverá ser inicializada com o ajuste dos seguintes parâmetros:

- ***dnn_conf_threshold* e *haar_min_neighbours***: Embora tenha sido definido os hiper-parâmetros ideais para maximização dos resultados, o algoritmo permite que esses sejam definidos pelo usuário uma vez que poderão existir regras de negócio mais ou menos rígidas quanto a performance do algoritmo.
- ***in_height* e *in_width***: O programa permite que seja definido a resolução da imagem a ser pré-processada pelo detector. Observa-se que o ajuste desse parâmetro tem grande peso no custo computacional do algoritmo, podendo aumentar, ou diminuir a imagem de entrada.
- ***dnn_model_choice***: Está disponível para escolha dois modelos de *Deep Learning* com diferentes *frameworks*; *Caffe* e *Tensorflow*
- ***detector_mode_choice***: Dependendo dos recursos disponíveis ou necessidade ajuste do poder de detecção, o programa permite escolher dentre 3 modos de funcionamento; o primeiro rodando apenas o detector *Haar Cascade*, o segundo rodando apenas *DNN* e o último e mais poderoso, o algoritmo feito por este trabalho com os dois rodando concomitantemente com os processamentos necessários para menor custo computacional.

3.4 Modelos para Reconhecimento Facial

Embora o foco deste trabalho seja a experimentação de modelo de redes neurais profundas para dispositivos embarcados, o algoritmo desenvolvido permite utilizar diversos modelos de *Deep Learning* sendo possível personalizar, conforme a aplicação, qual deverá ser empregado. Dois módulos principais são utilizado nessa fase. O primeiro o reconhecimento em si, e o outro que será responsável em treinar os modelos inteligentes.

3.4.1 Módulo para Cadastro

O fluxo para cadastro ocorre conforme a figura 3.6. Nessa etapa também foi desenvolvida uma interface lúdica para facilitar a usabilidade para cadastro do usuário final conforme a figura 4.3.

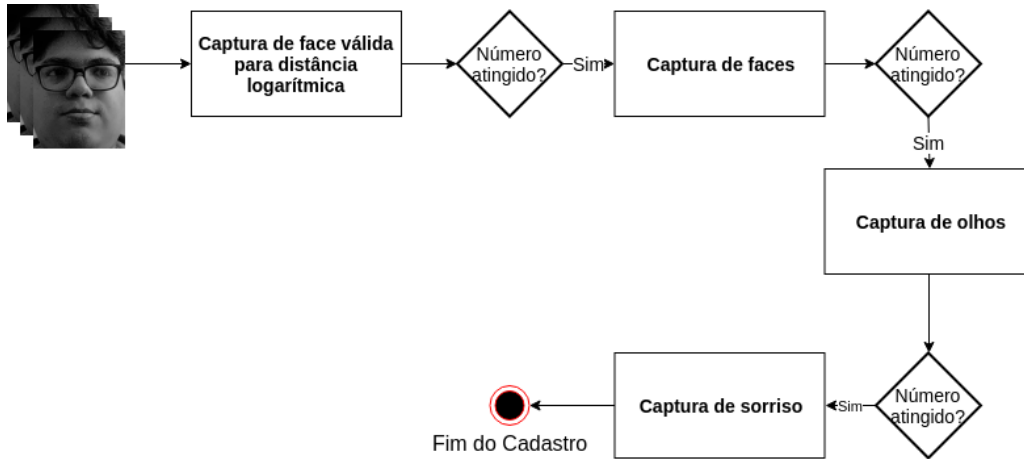


Figura 3.6: Fluxo de cadastro

A quantidade de fotos capturadas são determinadas pelos seguintes parâmetros:

- ***max_pics_validation***: Número de faces capturadas validadas para distância logarítmica.
- ***max_pics_face***: Número de faces capturadas usando modelo padrão.
- ***max_pics_eye***: Número de faces capturadas com os dois olhos do usuários aparecendo.
- ***max_pics_smile***: Número de faces capturadas com usuário sorrindo.

3.4.2 Módulo para Reconhecimento Facial

O fluxo de reconhecimento ocorre conforme a figura 3.7

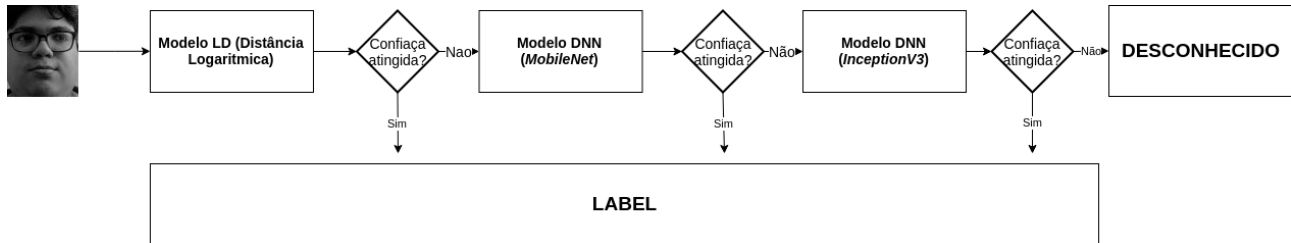


Figura 3.7: Fluxo de reconhecimento

Muitos são os parâmetros que controlam o comportamento do classificador facial, aqui serão apenas mostrados os que são determinantes quanto a acurácia das predições:

- ***display_frame_detection***: Possibilidade para mostrar o frame das detecções ocorrendo em tempo real na tela do dispositivo embarcado. Nota-se que o valor verdadeiro para este atributo tem grande impacto na velocidade de classificação.
- ***use_grayscale***: Se as imagens a serem inferidas deverão ser transformadas em tons de cinza.
- ***camera_resolution***: Qual a resolução a ser utilizada no dispositivo de captura. Aqui foi implementada a utilização automática da *PICamera*, se disponível.
- ***use_local_recognition***: Definição se o dispositivo embarcado deverá realizar a inferência ou um servidor dedicado.
- ***distance_tolerance***: Distância mínima para o modelo de distância logarítmica.
- ***min_confidence***: Valor mínimo em porcentagem do nível de confiança para o modelo de *Deep Learning*.
- ***server_url***: *URL* do servidor de IA que será utilizada para realizar diversas tarefas como inferência, download do modelo ou validação facial.

3.4.3 Módulo treino de RNN

Embora o módulo responsável no treino de redes neurais profundas não ser o foco deste trabalho, devido ao teor deste projeto ser a valorização do custo computacional, implementou-se a possibilidade para inserção de parâmetros e hiperâmetros do modelo que podem se dividir em: parâmetros para definição de diretórios, *data augmentation*, parâmetros controladores de treino e parâmetros de definição do modelo. Uma descrição de cada parâmetro se apresenta a seguir.

Parâmetros para definição de diretórios

Têm como objetivo definir os diretórios dos objetos a serem trabalhados:

- ***image_dir***: Diretório de entrada contendo as imagens, cada uma dentro de sua pasta de sua respectiva *label*.
- ***output_graph***: Diretório de saída onde será armazenado o grafo após o treino.
- ***intermediate_output_graphs_dir***: Diretório de saída onde serão armazenados os grafos intermediários do treino. Podendo ser recuperados em caso de falha ou necessidade de comparação com grafos futuros durante o treino.
- ***output_labels***: Diretório de saída onde serão armazenadas as *labels* das classes após o treino.
- ***summaries_dir***: Diretório para armazenamento de dados que poderão ser analisado com *Tensorboard*, ferramenta para depuração de modelos gerados com *Tensorflow*.
- ***bottleneck_dir***: Diretório para armazenar os dados de cache que permitirão uma maior agilidade em futuros treinos.

Hiperparâmetros para *Data augmentation*

Dependendo das regras de negócio do sistema a ser utilizado o reconhecimento facial, como por exemplo a necessidade de reconhecimento de dia ou de noite, reconhecimento por apenas

uma fração da face ou menos o uso de câmeras com diferentes resolução, o sistema permite que parâmetros que utilizarão pre processamento por *Data Augmentation* sejam utilizados:

- ***flip_left_right***: Inversão da imagem no eixo X, similar à uma rotação de 180°.
- ***random_crop***: Corte aleatório em forma de retangular da imagem, ou seja, uma sub-região da imagem.
- ***random_scale***: Uma porcentagem que representará, aleatoriamente um aumento ou diminuição das dimensões de cada imagem de entrada.
- ***random_brightness***: Uma porcentagem que representará, aleatoriamente um aumento ou diminuição do brilho de cada imagem de entrada.

Hiperparâmetros de Treino

Os hiperparâmetros de treino a serem definidos são:

- ***how_many_training_steps***: Em quantas etapas será realizado o treino do modelo.
- ***learning_rate***: Hiperparâmetro que definirá a taxa de aprendizado que controla o quanto os pesos serão ajustados em função do gradiente da função de perda.
- ***testing_percentage***: A porcentagem dos dados de entrada que deverão ser utilizado para teste.
- ***validation_percentage***: A porcentagem dos dados de entrada que deverão ser utilizado para validação.
- ***eval_step_interval***: A quantidade de passos até que uma nova avaliação do modelo seja realizada.
- ***train_batch_size***: A quantidade de imagens de entrada, por passo de treino, que o modelo deverá ser alimentado para o treino.
- ***test_batch_size***: A quantidade de imagens de entrada, por passo de teste, que o modelo deverá ser alimentado para realizar a sua avaliação final.

- ***validation_batch_size***: quantidade de imagens da coletânea de imagens, por passo de validação, que o modelo deverá ser alimentado para realizar a sua validação durante as etapas de treino.

Parâmetros do modelo

Finamente os parâmetros que controlarão o próprio modelo:

- ***intermediate_store_frequency***: A cada quantos passos um grafo intermediário deverá ser salvo.
- ***print_misclassified_test_images***: *Boolean* que definirá se, durante a etapa de testes, as imagens do dataset de treino deverão ser mostradas. Útil para depuração do modelo e encontro de possíveis imagens de entrada ruins.
- ***tfhub_module***: O modelo a ser utilizada a técnica de *Transfer Learning*. Aqui foi implementada a possibilidade de definição da url do modelo diretamente do domínio do *Tensorflow Hub*.

Capítulo 4

Resultados

4.1 Detecção Facial

O experimento foi realizado por meio da maximização dos dois algoritmos com maior poder de detecção a distância. *OpenCV - Haar Cascade* e *OpenCV - DNN*. Foi montado um dataset com 2778 imagens **sem qualquer face**, onde uma face detectada contaria como erro. Dessa forma foi possível avaliar os falsos positivos dos detectores nativos presente na biblioteca do *OpenCV*.

4.1.1 Seleção de hiperparâmetros

Ambos os modelos trabalhados possuem características que devem ser definidas conforme a aplicação. Tratando-se de reconhecimento facial alguns parâmetros podem "ajustar" como o modelo vai se comportar conforme outras variáveis de entrada. Em busca da limitação da taxa de erro o dataset utilizado a seguir busca encontrar erros por valor de hiperparâmetro experimentado.

4.1.2 *Haar Cascade*

No primeiro algoritmo foi experimentado o valor do hiperparâmetro (*Minimum Neighbours*) número mínimo de vizinho mais próximos. E avaliado o ganho de informação a partir do último valor do hiperparâmetro experimentado. O resultado se apresenta conforme a tabela 4.1

Tabela 4.1: Resultados Ganho de informação por hiperparâmetro

Porcentagem Acertos	Ganho (%)	Min. Vizinhos
23.15	23,1461483081353	0
57.99	34,8452123830094	1
73.79	15,8027357811375	2
81.97	8,17134629229659	3
87.01	5,03959683225351	4
90.32	3,31173506119509	5
92.48	2,15982721382281	6
93.88	1,4038876889849	7
95.03	1,1519078473723	8
96.08	1,043916486681	9
96.51	0,431965442764593	10
97.05	0,539956803455709	11
97.48	0,431965442764593	12
97.77	0,287976961843	13
98.02	0,251979841612709	14
98.49	0,467962562994998	15
98.70	0,215982721382304	16
98.99	0,287976961843	17
99.10	0,107991360691187	18
99.28	0,179985601151913	19

O grafo da figura 4.1 mostra claramente que a partir de 8 o número mínimo de vizinhos o grafo se estabiliza e o ganho converge a zero.

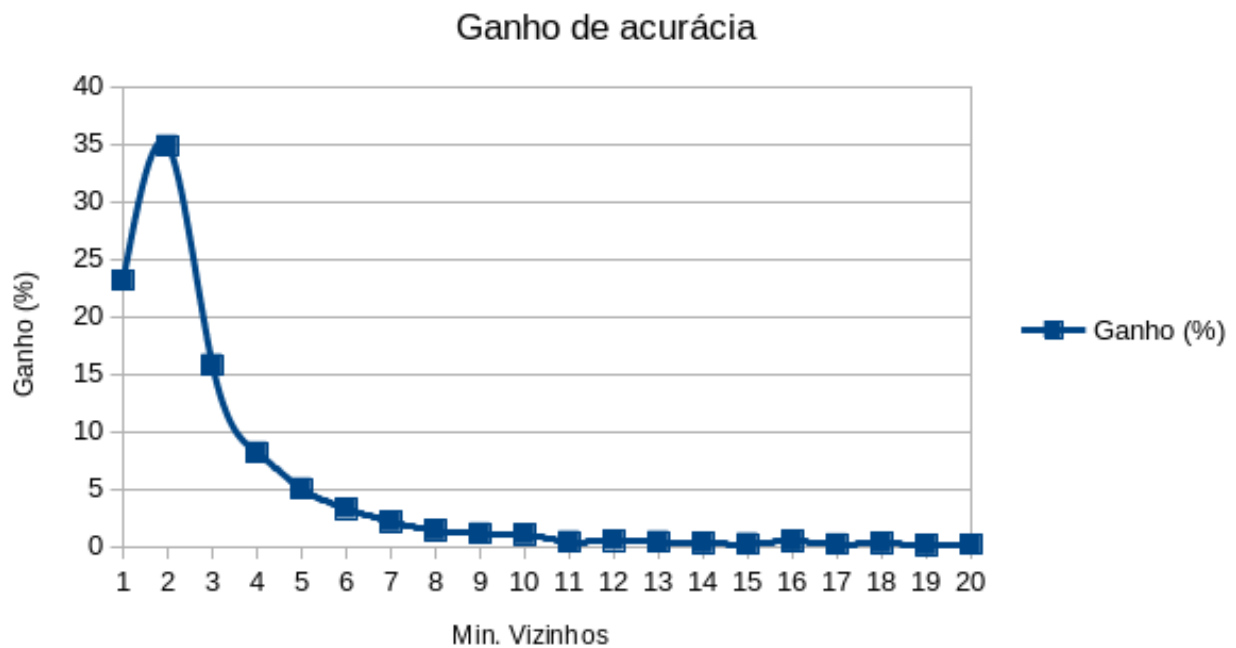


Figura 4.1: Resultados Haar Cascade
Fonte: Própria (2019)

4.1.3 DNN

No segundo algoritmo, que utiliza *Deep Learning* para detecção facial, foi experimentado o valor do hiperparâmetro nível mínimo de confiança, que define, para cada detecção realizada, o quão "confiante" o modelo está e se deve descartar ou não conforme o valor mínimo definido. Neste contexto foi obtido o seguinte resultado.

Tabela 4.2: Resultados Ganho de informação por hiperparâmetro

Porcentagem Acertos	Ganho (%)	Confiança (%)
0.00	0	10
64.47	64,4708423326134	20
80.78	16,3066954643628	30
87.29	6,5154787616991	40
90.78	3,49172066234699	50
93.27	2,4838012958963	60
95.68	2,4118070554356	70
97.23	1,5478761699064	80
98.34	1,1159107271418	90

Neste segundo caso, o grafo da figura 4.2 mostra claramente que a partir do nível de confiança

de 70% o grafo se estabiliza e o ganho converge a zero.

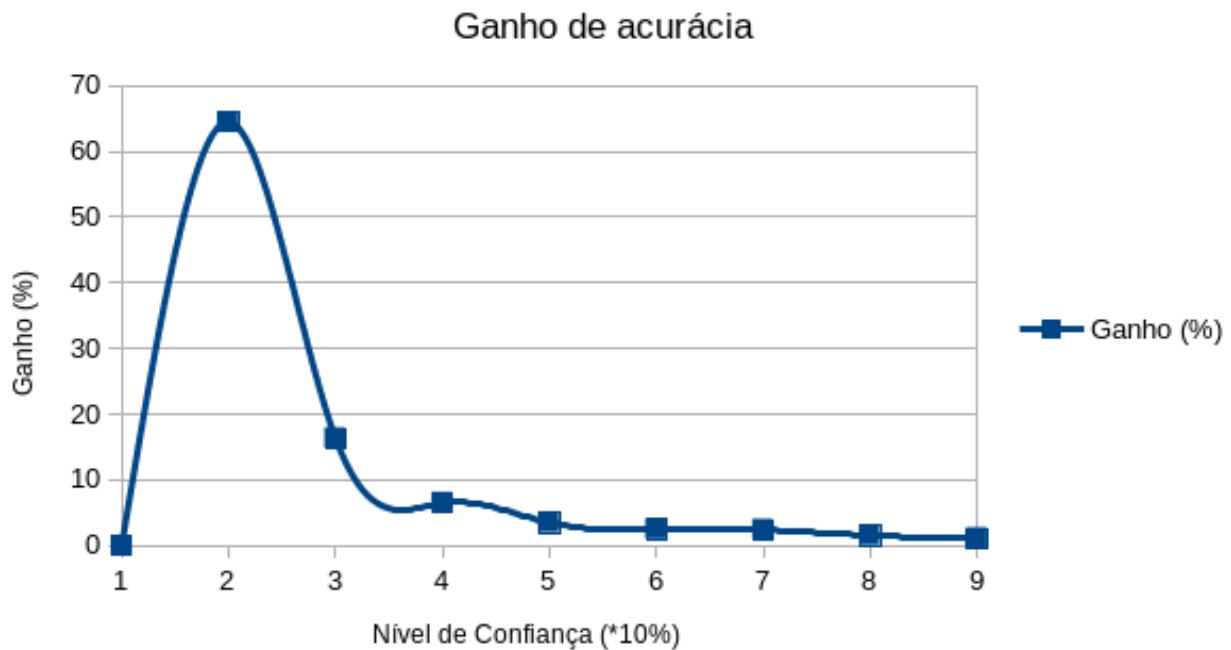


Figura 4.2: Resultados DNN
Fonte: Própria (2019)

4.1.4 Algoritmo de detecção facial proposto

Ao analisar os dados, é possível observar em ambos os gráficos que há, em determinado ponto, uma convergência do ganho de precisão. Tendo a natureza de algoritmos *machine learning*, quanto maior o nível de confiança para obter um resultado, menor é a capacidade de inferência do algoritmo, neste contexto, o poder de detecção facial. Procura-se **obter constante que maximize o número de faces detectadas com o menor falso positivo possível**.

Dessa forma foram escolhidos os valores de:

- Número de vizinhos: 8
- Nível de confiança: 70%

Para média de 95% de confiança que não ocorrerá erros nas detecções.

4.2 Cadastro de novas Classes

A imagem da figura 4.3 mostra a interface de captura lúdica do *dataset* com a observação que mesmo a disposição dessa janela é atributo configurável no sistema.

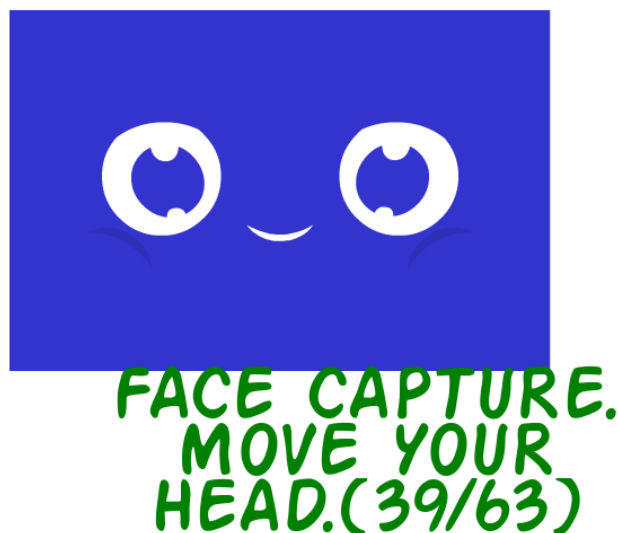


Figura 4.3: Inteface para captura no estado de captura de olhos

Foram utilizadas apenas 80 faces de cada usuário. Onde:

1. 1 face válida para o modelo de distância logarítmica
2. 65 faces com captura facial.
3. 10 faces com captura mostrando ambos os olhos.
4. 4 faces com captura mostrando captura de sorriso.

Esses valores não foram obtidos por meio da análise de precisão dos modelos, pois dessa forma, quanto mais imagens, melhor o comportamento do modelo. Foram restringidos valores para usabilidade do usuário final, onde, valores maiores, demandariam tempo demasiado do usuário.

4.3 Reconhecimento Facial

A imagem da figura 4.4 mostra a interface de reconhecimento operando no dispositivo embarcado, novamente o aparecimento dessa janela é atributo configurável no sistema.

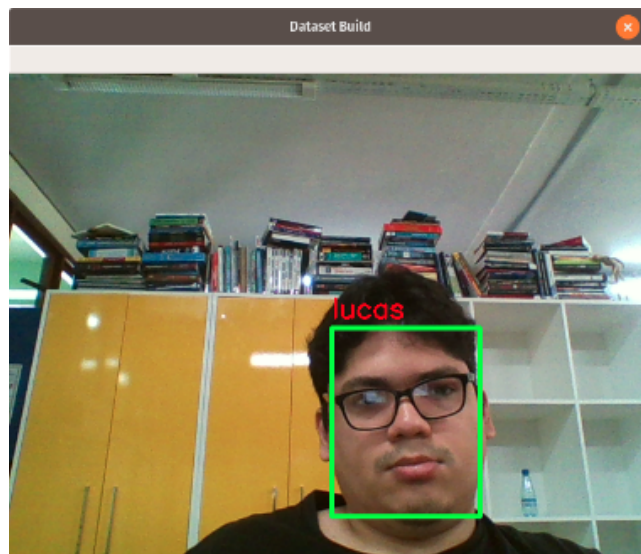


Figura 4.4: Interface mostrando a captura feita por um dispositivo Raspberry PI

4.4 Desempenho

Como mostrado na figura 3.7 só serão gastos recursos computacionais com novas inferências a medida que um modelo não consegue identificar a face. Essa abordagem foi necessária para reduzir um dos maiores desafios da área de reconhecimento facial: **reduzir os números de falsos positivos**. Essa abordagem foi idealizada ao ser analisado o algoritmo de *Haar Cascade*, como mostra a figura 2.12 recursos serão utilizado *on-demand* e só serão feitos novas gastos de computacionais quando for preciso. Dessa forma é difícil realizar uma análise de *frames* por segundo, uma vez que a análise de cada *frame* dependerá em qual modelo ocorreu o reconhecimento facial.

O algoritmo desse trabalho também permite dois modos de operações, onde os recursos computacionais poderão ser utilizado no dispositivo embarcado ou em um servidor externo.

Capítulo 5

Considerações Finais

Este trabalho consistiu no reconhecimento facial de pessoas por meio de técnicas de Inteligência artificial. O problema foi tratado como uma tarefa de aprendizado supervisionado de classificação, o qual utiliza um conjunto de dados para gerar modelos inteligentes. Foram utilizados modelos para distância logarítmica, Redes Neurais profundas como *MobileNet* e *InceptionV3*.

Embora seja natural ao se trabalhar com modelos inteligentes que quanto maior a carga inicial para gerar o modelo, maior a acurácia, o cadastro de novas classes foi feito centrado no usuário, onde de uma maneira simples e rápida, fará sua inserção no *Dataset*. A captura facial, feita por um dispositivo embarcado ou qualquer dispositivo baseado em *Debian*, passará pelo modelo de menor computacional até o que demanda mais recursos do dispositivo. O treino foi feito em servidor dedica, com Hardware capaz para se trabalhar com RNN, utilizando a biblioteca *TensorFlow*.

Em trabalhos futuros, pretende-se analisar mais modelos leves de *Deep Learning*, foi inserido no algoritmo a possibilidade de definição de modelo da *Web* ao se definir as suas *URLs*. No final deste projeto também foi homologada a versão 2.0 para da biblioteca *Tensorflow*, pode ser rentável analisar e, dependendo dos benefícios, realizar a migração de biblioteca.

Referências Bibliográficas

ANDERSON, J. R. *Cognitive Psychology and Its Implications*. [S.l.]: W. H. Freeman, 1980.

BEN-ARI, M. *Principles of Concurrent and Distributed Programming*. Rehovot, Israel: Prentice Hall, 1990.

BRINK, H. et al. *Real-world machine learning*. [S.l.]: Manning, 2017.

CHARNIAK, E.; MCDERMOTT, D. *Introduction to Artificial Intelligence*. [S.l.]: Addison-Wesley, 1985.

CHEESEMAN P., K. B.; TAYLOR, W. Where the really hard problems are. *IJCAI*, n. 91, p. 331–337, 1991.

CROSS, S. E.; WALKER, E. Dart: Applying knowledge based planning and scheduling to crisis action planning. *Morgan Kaufmann*, p. 711–729, 1994.

DAWSON-HOWE, K. *A practical introduction to computer vision with opencv*. [S.l.]: John Wiley & Sons, 2014.

FACELLI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. Rio de Janeiro, Brasil: LTC - Livros Técnicos e Científicos Editora Ltda, 2015.

FRANCOIS, C. *Deep learning with Python*. [S.l.]: Manning Publications Company, 2017.

GOODMAN, J.; HECKERMAN, D. Fighting spam with statistics. *Significance, the Magazine of the Royal Statistical Society*, n. 1, p. 69–72, 2004.

MARSLAND, S. *Machine learning: an algorithmic perspective*. [S.l.]: Chapman and Hall/CRC, 2011.

MOGHADDAM, B.; PENTLAND, A. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, n. 19, p. 696—710, 1997.

RANJAN, R. et al. Deep learning for understanding faces: Machines may be just as good, or better, than humans. *IEEE Signal Processing Magazine*, IEEE, v. 35, n. 1, p. 66–83, 2018.

RICH, E.; KNIGHT, K. *Artificial Intelligence*. 2. ed. [S.l.]: McGraw-Hill, 1991.

RUSSELL PETER NORVIG, T. R. C. S. S. *Inteligência Artificial*. 3. ed. Rio de Janeiro, Brasil: Elsevier, 2013.

- SARKAR, D.; BALI, R.; GHOSH, T. *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. [S.l.]: Packt Publishing Ltd, 2018.
- SIDENBLADH H., B. M. J.; FLEET, D. J. *Stochastic tracking of 3D human gures using 2D image motion*. Dublin, Ireland: Sixth European Conference on Computer Vision (ECCV 2000), 2000. 72-89 p.
- SNAVELY N., S. S. M.; SZELISKI, R. *Photo tourism: Exploring photo collections in 3D*. *ACM Transactions on Graphics*. [S.l.]: SIGGRAPH, 2006.
- SZELISK, R. *Computer Vision: Algorithms and Applications*. Nova York, Estados Unidos: Springer-Verlag GmbH, 2011.
- VIOLA, P.; JONES, M. J. Robust real-time face detection. *International journal of computer vision*, Springer, v. 57, n. 2, p. 137–154, 2004.
- YANG, M.-H.; KRIEGMAN, D. J.; AHUJA, N. Detecting faces in images: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE Computer Society, v. 24, n. 1, p. 34–58, 2002.