INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS

CAMPUS MANAUS DISTRITO INDUSTRIAL

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

ARTHUR CABRAL RAMOS

SISTEMA DE DETECÇÃO DE CAPACETE BASEADO EM INTELIGÊNCIA ARTIFICIAL

MANAUS - AM

2023

ARTHUR CABRAL RAMOS

SISTEMA DE DETECÇÃO DE CAPACETE BASEADO EM INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do titulo de Barachel em Engenharia de Controle e Automação, do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial – IFAM/CMDI

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS - AM

2023

Dados Internacionais de Catalogação na Publicação (CIP)

R175s

Ramos, Arthur Cabral.
Sistema de detecção de capacete baseado em inteligência artificial / Arthur Cabral Ramos. — Manaus, 2023.
51f.: il. color.

Monografia (Graduação) — Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, Curso de Engenharia de Controle e Automação, 2023.
Orientador: Prof.º Alyson de Jesus dos Santos, Dr.

1. Aprendizado de Máquina. 2. Inteligência artificial. 3. Tensorflow. I. Santos, Alyson de Jesus dos. II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.89

Elaborada por Oziane Romualdo de Souza (CRB11/ nº 734)

ANEXO 7

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 24 dias do mês de Março, de 2023, às 11h, o(a) discente Arthur Cabral Ramos apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Dr. Alyson de Jesus dos Santos (docente-orientador), Prof(a). Msc. Jaidson Brandão da Costa (Membro 1) e Prof(a). Msc. Gabriel Pinheiro Compto (Membro 2). A sessão publica de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título Sistema de Detecção de Capacete Baseado em Inteligência Artificial. Na sequência, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final 9,5 (Nove, Cinco) do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 20/04/2023, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às 12h15 min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alyson de Jesus dos Santos

Prof.(a) Avaliador 1: Jaidson Brandão da Costa

Prof.(a) Avaliador 2: Gabriel Pinheiro Compto

Discente: Arthur Cabral Ramos

AGRADECIMENTOS

Dedico esse espaço principalmente aos meus pais, que durante esses anos me apoiaram e deram todo o suporte necessário para que o caminho fosse o mais tranquilo possível.

Também reserva para agradeça aos meus irmãos, namorada e amigos que acompanharam nessa trajetória e me deram suporte de várias maneiras diferente.

Agradeço ao meu orientador, Prof. Dr. Alyson de Jesus dos Santos, sendo bastante acessível, e me motiva e ajuda com esse trabalho de grande importância.

“A dor é temporária, a vitória é eterna”.
Aatrox

RESUMO

Este trabalho tem por objetivo o desenvolvimento de um sistema baseado em aprendizado de máquina que controle o acesso a áreas nas quais o uso de capacete como equipamento de proteção individual é obrigatório. O projeto foi desenvolvido é três etapas principais: seleção de dados, treinamento e teste. No projeto utilizou-se hardware de baixo custo e software gratuitos. O sistema desenvolvido utilizou Python como linguagem de programação em todas as etapas e como principal ferramenta o TensorFlow. O funcionamento ocorre através de um código escrito em Python que avalia se o uso do capacete está sendo realizado, caso a resposta seja positiva, o mesmo aciona um dispositivo que libera o acesso, no projeto o dispositivo é representado por um LED.

Palavras-chave: Aprendizado de Máquina. Inteligência Artificial. TensorFlow.

ABSTRACT

This work aims to develop a system based on machine learning that controls access to areas where the use of helmets as personal protective equipment is mandatory. The project was developed is three main stages: data selection, training and testing. The project used low-cost hardware and free software. The developed system used Python as a programming language in all stages and TensorFlow as the main tool. The operation occurs through a code written in Python that evaluates whether the use of the helmet is being carried out, if the answer is positive, it triggers a device that grants access, in the project the device is represented by an LED.

Keywords: Artificial Intelligence. Machine Learning. TensorFlow.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Os acidentes de trabalho atingem diretamente a capacidade laboral do trabalhador e geram impactos negativos nos âmbitos sociais, econômicos e ambientais. De acordo com o Ministério Público do Trabalho (MPT), entre 2012 e 2021 foram registradas 22.954 mortes no mercado formal. Segundo dados do INSS, apenas em 2021 foram gastos 17,7 bilhões com auxílio-doença e 70,6 bilhões com aposentadoria por invalidez.

Nesse contexto, os equipamentos de proteção individual (EPI) garantem a proteção do trabalhador e segundo a NR-6, são selecionados de acordo com os perigos e risco ocupacionais identificados, atividade exercida, eficácia necessária para o controle da exposição ao risco, exigências estabelecidas em normas regulamentadoras, conforto e adequação do empregado ao equipamento fornecido, compatibilidade com o uso simultâneo de vários EPIs. Referindo-se a capacete, de acordo com o ANEXO I da NR-6, o equipamento protege contra impacto de objetos sobre o crânio, choques elétricos e agentes contra a face.

O projeto propõe o desenvolvimento de um sistema baseado em inteligência artificial que controlará o acesso a ambientes que é necessário utilizar o capacete como EPI.

1.1 JUSTIFICATIVA

Em um período de trabalho em uma fábrica de equipamentos eletrônicos, notou-se a falta do uso de capacete nas áreas de expedição e recebimentos, nas quais o uso do equipamento é obrigatório. A não utilização do capacete está atrelada a falta de conscientização do trabalhador e a falta de monitoramento e controle da área, visto que a realização das atividades de rotina ocupa grande parte da carga horária disponível. Por esse motivo, a uso da tecnologia para a realização do monitoramento e controle de acesso é extremamente vantajoso, pois assim, a segurança do trabalhador é garantida.

Segundo Fernandes e Chiavegatto Filho (2018), a utilização de big data e machine learning são mais eficientes que os métodos tradicionais como registrados em papel em Saúde e Segurança do Trabalho (SST). De acordo com o Governo do Brasil (2022), em 2019 foram registrados 11.917 acidentes de trabalho onde a parte atingida do corpo envolve a cabeça.

Nesse contexto, avaliou-se a necessidade de garantir que o equipamento seja utilizado. O projeto descreve o desenvolvimento de um sistema baseado em inteligência artificial que controla o acesso em áreas nas quais o uso de capacete é obrigatório. Através de aprendizado de máquina, o sistema avalia se está ou não sendo feito o uso do EPI, e então realiza um acionamento para permitir o acesso à área.

1.2 OBJETIVO

Diante desse cenário, a realização desse Trabalho de Conclusão de Curso (TCC), tem como objetivo o desenvolvimento de um sistema baseado em inteligência artificial que irá garantir que o trabalhador acesse a área utilizando o capacete e dessa formar a diminuir a ocorrência de acidente de trabalho por conta da falta de capacete. Uma das preocupações do projeto é garantir o funcionamento em tempo real, baixo custo e devido a sua característica de sistema inteligente, ser otimizável.

.

1.3 OBJETIVO ESPECÍFICO

Para alcançarmos um resultado satisfatório, os objetivos específicos são:

a) Desenvolver um banco de dados.

b) Organizar o ambiente de desenvolvimento para o treinamento.

c) Treinar o sistema a partir do banco de dados.

d) Testar o sistema através de imagens.

e) Montar o circuito que simula o dispositivo de controle de acesso e conectar ao Arduino.

f) Integrar os sistemas de software e hardware que simulam o controle de acesso em tempo real.

1.4 METODOLOOGIA

O projeto se enquadra como um trabalho de pesquisa e desenvolvimento, pois visa o desenvolvido de um software que tem como objetivo realizar a detecção através de imagem seguindo os seguintes passos:

a) Levantamento Bibliográfico
b) Coleta de dados
c) Treinamento do sistema
d) Testes do sistema

Neste projeto foi escolhido a detecção através de visão computacional aplicando IA, pois o mesmo tem as vantagens de funcionamento em tempo real, baixo tempo de resposta, baixo custo a possibilidade de ser otimizável. A desvantagem está na demora do treinamento.

2 REFERENCIAL TEÓRICO

Este capítulo tem por objetivo descrever a base conceitual que facilitará o entendimento do projeto.

2.1 NORMA REGULAMENTADORA (NR)

A respeito das Normas regulamentadoras pode-se afirmar que:

> As Normas Regulamentadoras (NR) são disposições complementares ao Capítulo V (Da Segurança e da Medicina do Trabalho) do Título II da Consolidação das Leis do Trabalho (CLT), com redação dada pela Lei nº 6.514, de 22 de dezembro de 1977. Consistem em obrigações, direitos e deveres a serem cumpridos por empregadores e trabalhadores com o objetivo de garantir trabalho seguro e sadio, prevenindo a ocorrência de doenças e acidentes de trabalho (Ministério do Trabalho e Previdência, 2022).

As primeiras NR foram publicadas em 1978, e as demais foram criadas a longo do tempo. Sua revisão ocorre através dos sistemas tripartite paritário, por meio de grupos e comissões compostos por representantes do governo, empregadores e trabalhadores (Ministério do Trabalho e Previdência, 2022).

2.1.1 NR 6 EQUIPAMENTO DE PROTEÇÃO INDIVIDUAL - EPI

A NR 6 é a norma regulamentadora que estabelece os requisitos para o uso, fornecimento, aprovação e comercialização de EPI.

Segundo a NR 6, é dever da Organização exigir o uso e fornecer gratuitamente o EPI. Para selecionar o EPI, deve-se levar em consideração os perigos e riscos ocupacionais identificados, atividade exercida, eficácia necessária para o controle da exposição ao risco, exigências estabelecidas em normas regulamentadoras, conforto e adequação do empregado ao equipamento fornecido e compatibilidade com o uso simultâneo de vários EPIs.

Quanto aos deveres do trabalhador, cabe a ele utilizar o equipamento para a finalidade destinada, uso correto e informar a organização no caso do extravio ou danificação que torne o uso impróprio para o uso.

2.1.2 CAPACETE

Segundo o ANEXO I da NR 6 que lista os equipamentos para proteção individual, o capacete entra na categoria A, como EPI para proteção de cabeça. Abaixo são listados os casos no qual deve-se realizar o uso do capacete.

a) capacete para proteção contra impactos de objetos sobre o crânio;

b) capacete para proteção contra choques elétricos; e

c) capacete para proteção do crânio e face contra agentes térmicos.

2.2 VISÃO COMPUTACIONAL

Ballard e Brown, em 1982 definiram visão computacional como a ciência que estuda e desenvolve tecnologias que possibilitam que as máquinas enxerguem e extraiam informações do meio, através de dispositivos e sensores, como o dispositivo presente na Figura 1, que captura imagens. Através dessas informações, permite-se reconhecer, manipular e processar dados sobre as informações contidas nas imagens (Barelli, 2018).

Figura 1 - Webcam

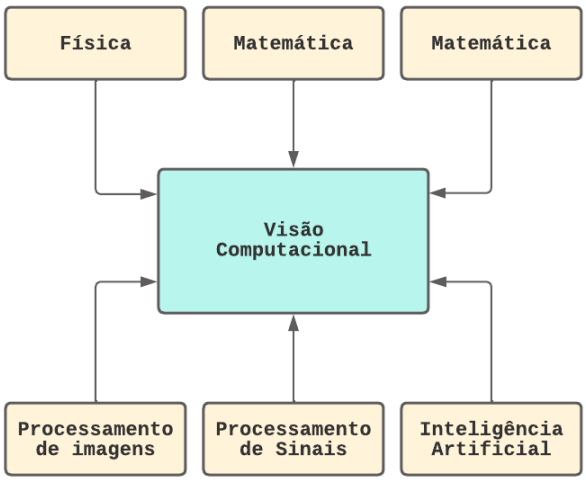
Fonte: LOGITECH (2022)

Programar computadores para que sejam capazes de enxergar o ambiente ao seu redor, pode ser uma tarefa complexa, já que dependem da matemática e de programação. Porém, os sistemas baseados em visão computacional vêm

simplificando cada vez mais essa tarefa e tornando-a mais presente no cotidiano (BARRELI, 2018). Atualmente, existem diversas bibliotecas e linguagens de programação para que sejam desenvolvidos programas voltados a visão computacional, de formar a simplificar a matemática e tornar o uso fácil.

Os principais campos ligados a visão computacional são: processamento de sinais, matemática, processamento de imagens, inteligência artificial, reconhecimento de padrões e física. A Figura 2 apresenta os campos ligados a visão computacional.

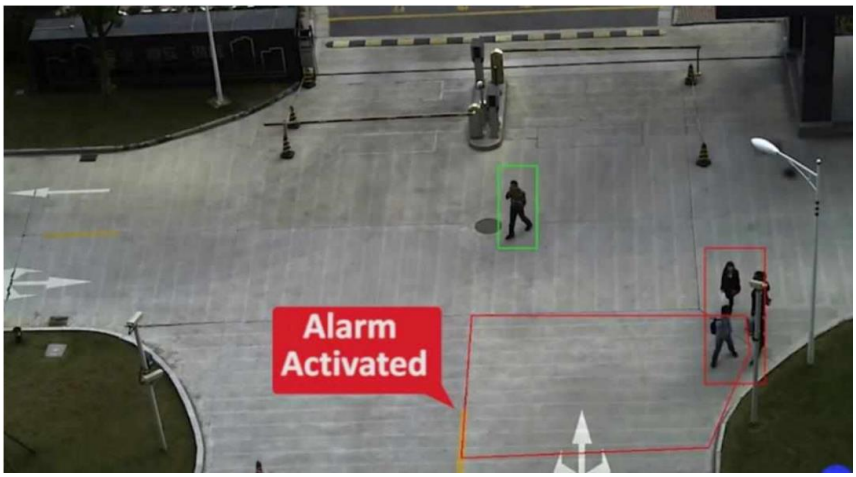
Figura 2 - Campos ligados a visão computacional



Fonte: O Autor (2023)

Como muitas tecnologias, a visão computacional é mais uma que procura imitar um comportamento humano. Outra definição de visão computacional seria definida como um complemento da visão biológica (BARRELI, 2018). Suas aplicações são variadas, sendo utilizada, por exemplo, na área de segurança com a detecção de movimentos suspeitos (GONZAGA, 2017), como mostra a Figura 3.

Figura 3 - Sistema de segurança



Fonte: GONZAGA (2017)

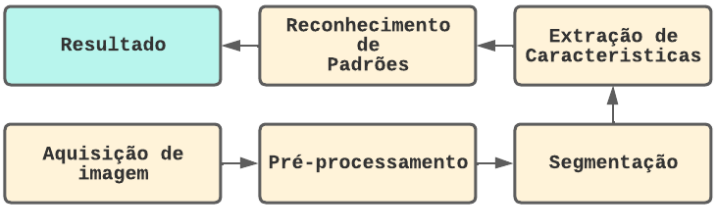
Na Figura 4 é apresentado um sistema avançado de visão computacional, no qual o veículo detecta pessoas, placas, ruas e outros objetos.

Figura 4 - Veículo com sistema de visão computacional

Fonte: BARELLI (2018)

Apesar dos sistemas de visão computacional serem utilizados em diversas áreas, normalmente o seu desenvolvimento segue o seguinte fluxo: Aquisição de imagens, pré-processamento, segmentação, extração de características, reconhecimento de padrões e resultado, como mostra a Figura 5.

Figura 5 - Fluxograma



Fonte: O autor (2022)

As etapas do fluxo de desenvolvimento da visão computacional podem ser definidas como:

1. Aquisição de imagem: O processo de aquisição de imagem para os sistemas de visão computacional é divido em dois elementos: o hardware (câmeras, computadores e sistema de iluminação) e o programa, responsável por processa as imagens e gerenciar as ações a serem realizadas (RUDEK, 2001).

2. Pré-processamento: A partir de um conjunto de dados, a fase de pré-processamento visa solucionar problemas nos dados, como dados corrompidos, atributos irrelevantes e ruídos (BATISTA, 2003). O pré-processamento é utilizado para aprimorar a imagem, melhorando a qualidade, corrigindo distorções, iluminação, contraste, nitidez, com a finalidade de evitar que o sistema identifique padrões com atributos irrelevantes.

3. Segmentação: A segmentação tem como objetivo reduzir o tamanho da imagem, removendo o fundo e mantendo apenas as informações de interesse (RUDEK, 2001). Outro objetivo é para que seja possível compreendê-la melhor. Técnicas de pré-processamento envolvem o destacamento de bordas e formas geométricas através de aplicação de ruído a imagem, facilitando a obtenção da informação pelo sistema de visão computacional (BARRELI, 2018).

4. Extração de características: Nessa etapa é feito a diferenciação do objeto de interesse, para que seja possível diferenciá-lo mesmo com ruído (RUDEK, 2001). Dessa forma, nas imagens coletadas, após a segmentação é feito a extração das informações de interesse, para que em seguida seja tomada alguma ação. Os problemas que dificultam a identificação do objeto estão relacionados à oclusão, onde o objeto fica parcialmente escondido atrás de outro, ou temos dois objetos compondo a mesma imagem. De forma semelhante, a perda de informação nas imagens ocorre

com devido a deformação, ruído e condições de iluminação anormal e ineficiência na segmentação (BEIS, 1999).

5. Reconhecimento de padrões: Imagens são processadas por ambientes computacionais, nesses sistemas é possível analisar informações das imagens e tomar alguma ação. Nesse contexto o reconhecimento de padrões consiste em separar o objeto de interesse dos outros elementos da imagem, que identifica informações semelhantes nas imagens do banco de dados.

2.3 INTELIGÊNCIA ARTIFICIAL

2.3.1 DEFINIÇÃO

A inteligência artificial (IA) é uma ciência que constrói modelos informáticos capazes de ter comportamento inteligente ou racional, de forma a facilitar e automatizar soluções de problemas em diferentes áreas de conhecimento. Também pode ser definida como a habilidade para adquirir, compreender, pensar, raciocinar e aplicar conhecimento (D'ADDARIO, 2022).

Os problemas que utilizam Inteligência Artificial são complexos e não possuem uma solução algorítmica, não se sabe como escrever programas para resolvê-los. Porém existe algo em comum em todos os problemas resolvidos com IA: a grande disponibilidade de dados, que possibilita o treinamento de algoritmos e o uso de aprendizado de máquina (LUDEMIR, 2021).

2.3.2 ORIGEM

Sua origem remete a tentativa de criar autômatos, que simulem habilidades dos seres humanos, mas a origem do conceito se deve ao matemático Alan Turing e o termo ao McCarthy, que se reuniram com pesquisadores em 1956 no Darmouth College (Estados Unidos), para discutir a probabilidade de construir máquinas que não se limitam a realizar cálculos predeterminados, mas sim, operações “inteligentes” (D'ADDARIO, 2022).

2.3.3 APLICAÇÃO

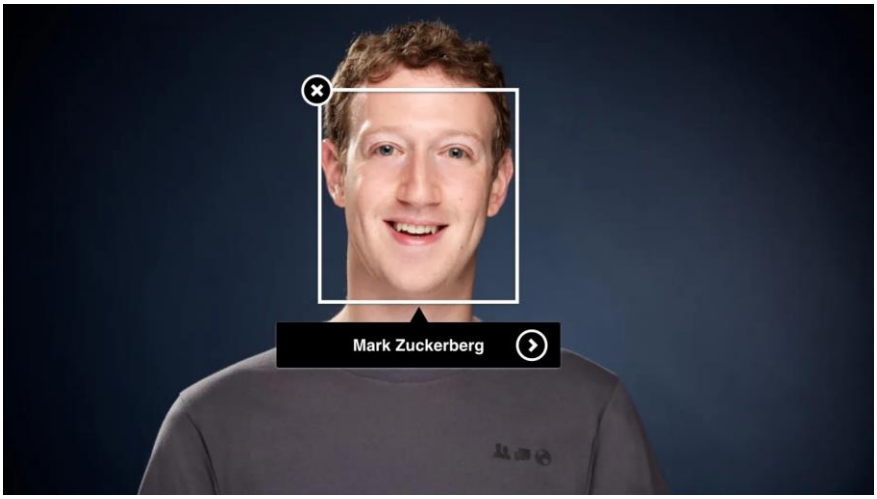
Com a inteligência artificial, as máquinas que desenvolviam trabalhos manuais passaram a realizar trabalho racionais, como dirigir carros, como os desenvolvidos pela Google e Tesla, capazes de se moverem-se sem motorista. O uso de forma totalmente autônoma está aguardando apenas a legislação adequada e teste em ambientes reais (LUDEMIR, 2021). O teste da inteligência de direção é uma etapa crítica para implementação de veículos autômatos. Atualmente, os testes realizados baseiam-se em simulações realista de ambientes de direção natural, no entanto, devido à raridade de eventos críticos para a segurança e dimensionalidade alta do ambiente, seriam necessários centenas de milhões de quilômetros para demonstrar seu desempenho (FENG et al., 2021).

Outra aplicação que se encontra em um estado mais avançado é a de tradutores automáticos, como os da Google, que devido a quantidade de textos processados, realiza traduções cada vez melhores. A manipulação de texto se enquadra no tipo de inteligência artificial conhecida como linguagem natural, que faz a ponte entre os idiomas e os computadores (TORFI et al, 2020).

Um tipo de inteligência artificial cada vez mais comum, é o de recomendação, como a Netflix (recomendação de filme e série), Amazon (recomendação de produtos de varejo e livros), Youtube (recomendação de vídeo e conteúdo). As recomendações são mais precisas de acordo com o tempo de uso do aplicativo, se tornando cada vez mais compatível com as preferências do usuário.

Uma aplicação muito conhecida e bem sucedida é a do Facebook apresentada na Figura 6, que utilizando Redes Neurais Profunda, desenvolveu o DeepFace, que identifica rostos humanos em imagens digitais independente do ângulo (LUDEMIR, 2021).

Figura 6 - Identificação de face



Fonte: SERENGIL, SEFIK ILKIN (2020)

2.3.4 EVOLUÇÃO DA IA

As primeiras técnicas de IA foram definidas na década de 50, porém sua aplicação está se tornando cada vez mais comum, devido a evolução do processamento computacional, o que é um pré-requisitos para sistemas baseados em IA (LUDEMIR, 2021). As GPU (Graphic Processing Unit), estão mais potentes e conseguem gerenciar muitos dados, o que é uma necessidade dos sistemas de IA, visto que são baseados em dados (ARADI, 2022).

2.4 APRENDIZADO DE MÁQUINA (AM)

O aprendizado de máquina surgiu como campo da inteligência artificial na década de 60 que desenvolve aprendizado através de dados. No final da década de 90, essa área se expandiu a ponto de estabelecer como um campo por si só (IZBICKI; DOS SANTOS, 2020).

No Aprendizado de máquina os algoritmos são orientados a dados, ou seja, aprendem a partir de um certo volume de dados gerando uma hipótese (Ludemir, 2021). Tendo-se um modelo com alguns parâmetros definidos, o aprendizado ocorre através de dados treinados ou experiências passadas.

Suas aplicações são multidisciplinares, sendo aplicados em finanças, detecção de fraude, manufatura, medicina, telecomunicações, robótica, reconhecimento de fala, pessoas ou objetos (ALPAYDIN, 2020).

Existem quatro principais tipos de aprendizado de máquina: Por reforço, supervisionado, não supervisionado e semi-supervisionado.

No Aprendizado por reforço, o algoritmo recebe a resposta correta e um sinal de reforço, de recompensa ou punição, o algoritmo faz uma hipótese a partir dos exemplos e determina se a detecção foi boa ou ruim (LUDEMIR, 2021). Os algoritmos possuem um agente que percorre um ambiente e aprende através de tentativa e erro, como se comportar e obter o melhor desempenho, o agente deve perceber o estado do ambiente e tomar uma ação que resulte em um novo estado (ARADI, 2020).

O aprendizado Supervisionado é o método mais utilizado. Para cada exemplo apresentado é necessário definir um rótulo informando a qual classe o exemplo pertence. O objetivo é identificar e classificar exemplos não rotulados, através de um banco de dados de exemplos, onde cada exemplo possuem um vetor de valores (atributos) e um rotulo associado (LUDEMIR, 2021).

No aprendizado não supervisionado, são fornecidos exemplos sem rótulos. E o objetivo do algoritmo é agrupar os exemplos por similaridade, analisando os exemplos fornecidos e tentando realizar o agrupado através de seus atributos. Após a formação dos agrupamentos, se faz necessário analisar o que cada agrupamento significa, no problema analisado (LUDEMIR, 2021).

O aprendizado semi-supervisionado, combina dados não rotulados e rotulados, de forma a diminuir o esforço humano no algoritmo, outra vantagem é a grande disponibilidade de dados não rotulados. As desvantagens estão na possibilidade de rotulação de baixa qualidade, o que tende a degradar o desempenho da classificação, outro ponto é a dificuldade em identificar as classificações com baixa qualidade que poluem o sistema desenvolvido (ZHU, 2005).

2.5 PYTHON

O Python foi criado em 1990 por Guido van Rossum e foi originalmente era destinado a físicos e engenheiros (BORGE; EDUARDO, 2014). Segundo da Silva e Vieira (2022), Python se tornou umas das linguagens de programação mais populares,

aparecendo em terceiro na lista TIOBE Index em setembro de 2019. Segundo Borge e Eduardo (2014), Python é uma linguagem bem aceita na indústria, utilizada por empresa como Google, Microsoft, Disney. O que facilita sua aplicação na indústria é a característica de código aberto, compatível com a General Public License [GPL], o que permite a incorporação em produtos proprietários.

Python é uma linguagem de alto nível orientada a objeto que possui sintaxe clara e facilita a legibilidade do código-fonte, o que a torna mais produtiva. A linguagem conta com diversas estruturas como lista, dicionários, módulos prontos (BORGE; EDUARDO, 2014).

Na linguagem Python, cujo logo está apresentado na Figura 7, não é necessário declarar variável, o significa dizer que é uma linguagem de tipagem dinâmica. Suas aplicações são variadas, sendo utilizada em desenvolvimento de sistemas e softwares além de permitir a integração com outras linguagens como C e Fortran (BORGE; EDUARDO, 2014).

Figura 7 - Logotipo Python

Fonte: PYTHON (2022)

Segundo Da Silva e Vieira (2022), Python permite trabalhar com vários paradigmas, é orientado a objeto e também permite que os programas sejam baseados em funções. Por permitir prototipagem rápida e não ser compilado, é ideal para áreas como aprendizado de máquina e análise de dados.

2.6 OPENCV

Tessera OpenCV (Open Source Computer Vision Library) é uma biblioteca de código aberto que possui uma comunidade de usuário de mais de 47 mil pessoas número de downloads superior a 18 milhões. Utilizada para o desenvolvimento de software de visão computacional e aprendizado de máquina, a biblioteca possui mais

de 2500 algoritmos otimizado. Os algoritmos desenvolvidos com OpenCV podem ser utilizados para reconhecer objetos, identificar rostos, rastrear movimentos de câmera, extrair modelos 3D de objetos, encontrar imagens semelhantes em bancos de dados (OPENCV, 2022).

OpenCV possui suporte a Windows, Android, Linux e Mac OS, possui interfaces para Python, C++, MATLAB e Java. É utilizado por empresas como Google, Yahoo, Microsft, Intel, IBM, Sony, Honda, Toyota (OPENCV, 2022).

2.7 TENSORFLOW

TensorFlow é uma plataforma de desenvolvimento que facilita a implementação de aprendizado de máquina. Possui vários níveis de abstração, podendo trabalhar com GPU e CPU e implementar aprendizado de máquina em grande escala (TENSORFLOW, 2022; DA SILVA; VIEIRA, 2022).

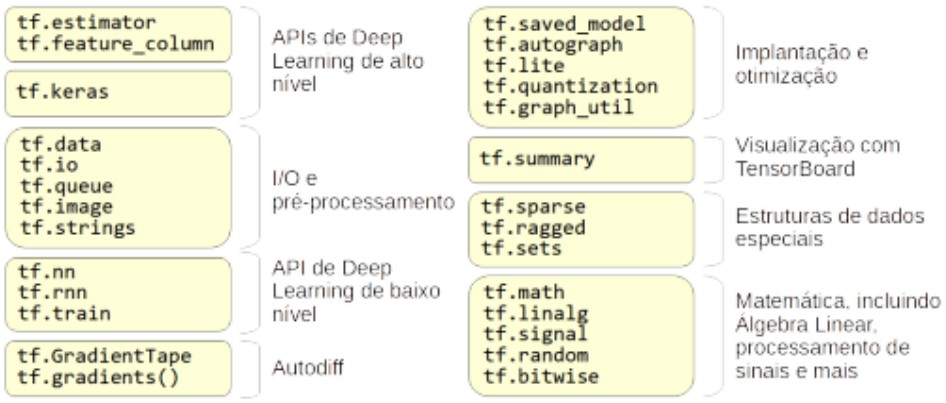
Segundo Da Silva e Vieira (2022), TF representa a execução de funções através de grafo dirigido, cada nodo representa uma operação, como soma ou multiplicação. As informações fluem através de grafo de computação na forma de vetores chamados de tensores.

TF foi desenvolvido pela Google e é utilizado por diversas empresas, como Intel, Coca-Cola, Lenovo, Paypal que utiliza para detecção de fraude e SINOVATION VENTURES que utiliza na classificação e segmentação de doenças em imagens de tomografia de retinas (TENSORFLOW, 2022).

TensorFlow foi desenvolvido pela Google e é utilizado por diversas empresas, como Intel, Coca-Cola, Lenovo, Paypal que utiliza para detecção de fraude e SINOVATION VENTURES que utiliza na classificação e segmentação de doenças em imagens de tomografia de retinas (TENSORFLOW, 2022).

Os sistemas que utilizam TF possuem alguns requisitos para sua implementação, como bibliotecas e funções apresentadas na Figura 8 como funções de processamento de E/S, otimização, operações aritméticas, deployment e tensorboard que fornece ferramentas para acompanhar as métricas como perda e precisão do sistema desenvolvido (DA SILVA; VIEIRA, 2022).

Figura 8 - API do Tensorflow para Python



Fonte: DA SILVA e VIEIRA (2022)

2.8 KERAS

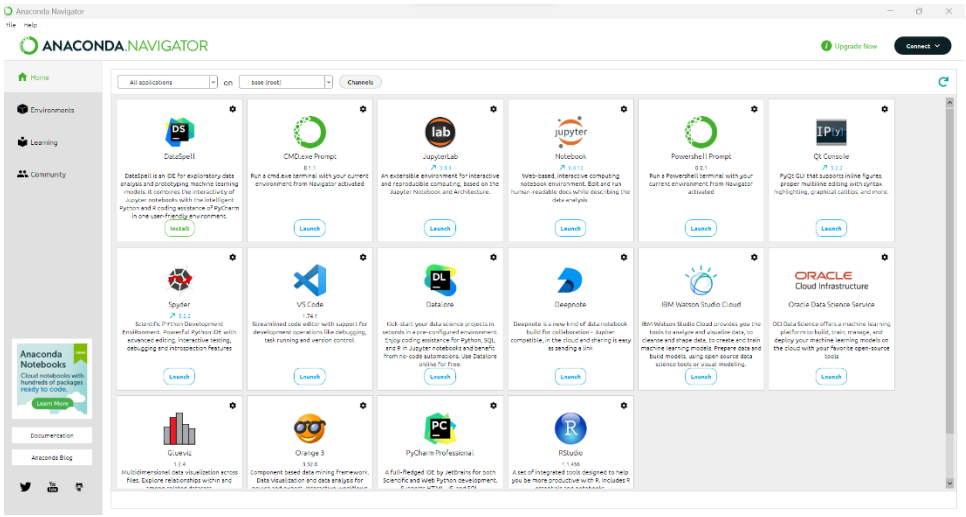
Keras é uma API desenvolvida em 2015 que facilita o uso de tensorflow, inicialmente utilizada com Theano, que é outra biblioteca para manipulação de tensores. A partir de 2018, Keras se tornou a API de alto nível oficial do tensorflow (DA SILVA; VIEIRA, 2022).

2.9 JUPYTER NOTEBOOK

Jupyter Notebook é um aplicativo Web de código aberto que suporta mais de 40 linguagens de programação que incluem Python, Julia, R e Scala. O aplicativo também pode trabalhar com uma vasta quantidade de saídas (vídeo, imagem, html) (Jupyter, 2022). O Jupyter Notebook funciona com um laboratório virtual que garante uma fácil visualização e um bom fluxo de trabalhos, códigos e dados. Sua relevância está em sua interação com várias bibliotecas digitais, como conjuntos de dados, documentações, software e publicações (RANDLES et al., 2017).

O Jupyter Notebook está disponível através do aplicativo anaconda, como mostra a Figura 9. O aplicativo está disponível em https://www.anaconda.com.

Figura 9 - Anaconda interface



Fonte: O autor (2023)

2.10 ARDUINO

O Arduino é uma plataforma de computação embarcada programada para processar entradas e saídas entre dispositivos externos conectados a ele. Esses dispositivos podem ser LEDs, displays de matriz de pontos, botões, interruptores, motores, sensores, receptores GPS, módulos Ethernet e Wifi ou qualquer dispositivo controlável ou que possa emitir dados (MCROBERTS, 2018).

A placa Arduino UNO R3, como a apresentada na Figura 10 é a mais utilizadas entre as placas oficiais do Arduino, tendo como principais recursos e características um processador ATMega328P, quatorze pinos de entrada e saída digital, seis entradas analógicas, um ressonador cerâmico de 16MHz, uma conexão USB, um conector de energia e um botão de reset (ARDUINO, 2022).

Figura 10 – Arduino UNO R3

Fonte: ARDUINO (2022)

O hardware e software são open-source, o que significa que seus esquemas e projetos podem ser utilizados de forma livre, dessa maneira, é possível projetar placas baseadas nos projetos de Arduino (MCROBERTS, 2018). Para programar o Arduino, é necessário utilizar o IDE dele, ou seja, seu ambiente de desenvolvimento integrado e sua linguagem que é baseada em C/C++ (MCROBERTS, 2018).

O Arduino permite que suas funcionalidades sejam estendidas através de Shields (escudos), que são placas de circuito que contém outros dispositivos, como exemplo, receptores de GPS, display de LCD, módulos ethernet, extensão de entradas de Micros SD (MCROBERTS, 2018; ARDUINO, 2022).

3 PROJETOS RELACIONADOS

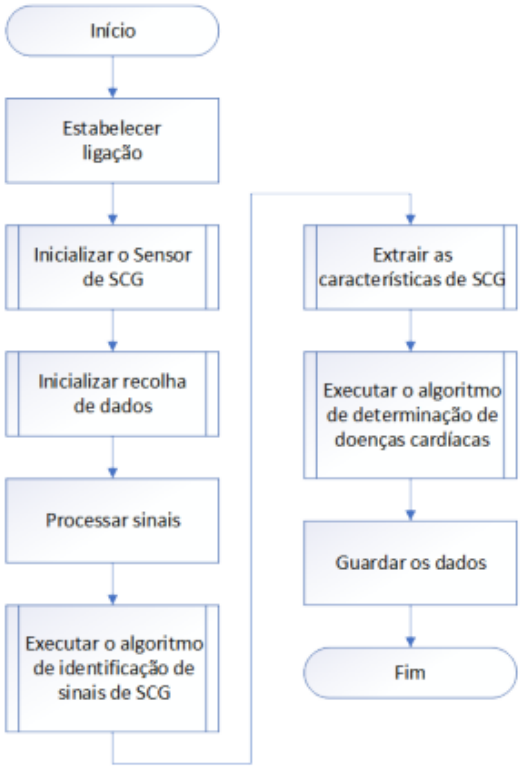
Este capítulo apresentar alguns trabalhos relacionados a tecnologias utilizadas neste projeto. Desta forma são feitas análises para identificar e comparar o foco do trabalho, linguagens de programação, ambiente integrado de desenvolvimento de programas e abordagem.

3.1 IMPLEMENTAÇÃO DE ALGORITMO DE TENSORFLOW PARA DETECTAR PATOLOGIAS CARDÍACAS (PROJETO 1)

Este projeto foi desenvolvido por pesquisadores da universidade do Minho Escola de Engenharia. O objetivo do projeto está em desenvolver um sistema capaz de analisar o sinal de um dispositivo SCG, que registra informações em tempo real do coração através de sensores e utilizar uma tecnologia baseada em inteligência artificial para analisar as informações registradas e alertar sobre possíveis problemas cardíacos (HUANG, HAO 2019).

Os dispositivos SCG possui imperfeições, por isso, no projeto foi utilizado técnicas de pré-processamento para melhorar a qualidade do sinal, para que então o sistema desenvolvido, baseado em “Machine Learning”, através da biblioteca “Open Source” de “Tensorflow” para implementar um algoritmo fosse capaz de prever a evolução de cada paciente. Dessa forma, através dos dados recolhidos do doente, possibilitou-se aos médicos uma facilidade para identificar doenças cardíacas (HUANG, HAO 2019). O fluxograma do projeto é apresentado na Figura 11.

Figura 11 – Fluxograma do sistema



Fonte: HUANG (2019)

3.2 RECONHECIMENTO FACIAL PARA DETECÇÃO DE EMOÇÕES UTILIZANDO REDES NEURAIS CONVOLUCIONAIS COM TENSORFLOW (PROJETO2)

O projeto de localização de robôs por reconhecimento ótico de caracteres de placas foi desenvolvido po O projeto desenvolvido por pesquisadores da Universidade Católica de Goiás tem como objetivo identificar expressões utilizando rede neurais convolucionais. O projeto foi desenvolvido em Python utilizando as bibliotecas OpenCV, Tensorflow e a API do Keras (FERREIRA, 2021).

Como fonte de dados, foi utilizado um banco de dados conhecido como FER2013, disponível no site Kaggle, onde os dados consistem em imagens de dimensões de 48x48 pixels e um conjunto de 28709 imagens. Dentre essas imagens de expressões faciais, tem-se setes classes de rótulos, sendo eles: triste; raiva; nojo; medo; feliz; surpreso e neutro (FERREIRA, 2021).

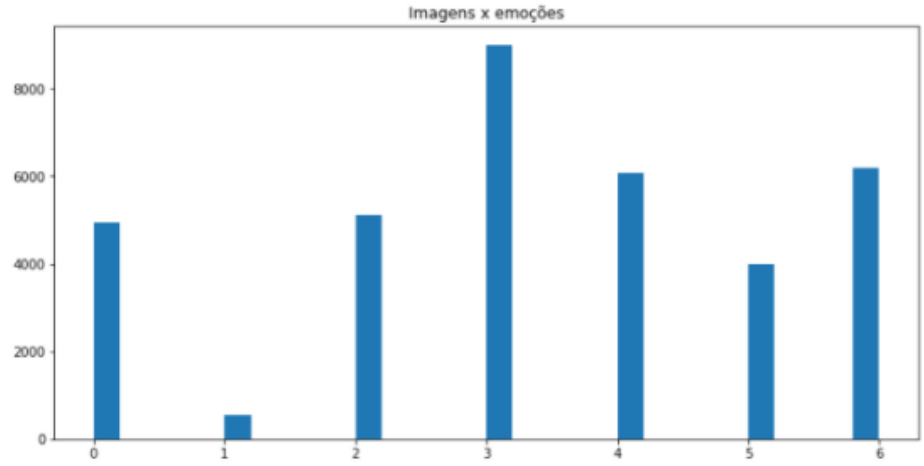
A classe que apresentou maior número de acertos foi a ‘Feliz’ e a com menor número de acertos a emoção ‘Nojo’, como mostra a Figura 12.

Figura 12 – Plot do gráfico com a quantidade de imagens para cada classe

```
plt.figure(figsize=(12,6))
plt.hist(data['emotion'], bins = 30)
plt.title('Imagens x emoções')

# Classes: ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
```

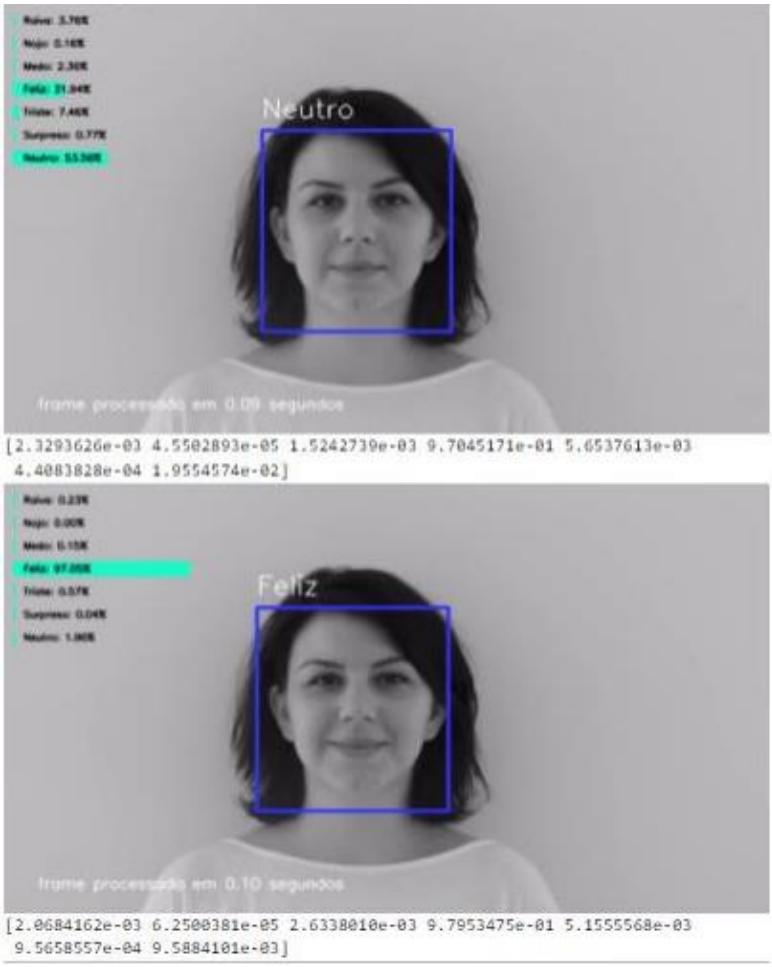
```
Text(0.5, 1.0, 'Imagens x emoções')
```



Fonte: FERREIRA (2021)

A identificação das emoções utiliza haarcascade para localizar as faces nas imagens e devido a isso em alguns momentos as faces podem não ser detectadas devido aos parâmetros para localização da face estarem incorretos ou não otimizados. No teste e detecção em vídeo dividido em frames, que funcionam como imagens para o modelo, como mostra a Figura 13.

Figura 13 – Reconhecimento de emoções em vídeo



Fonte: FERREIRA (2021)

3.3 SISTEMA DE DETECÇÃO DE OBJETO PERSONALIZADO UTILIZANDO INTELIGÊNCIA ARTIFICIAL (PROJETO 3)

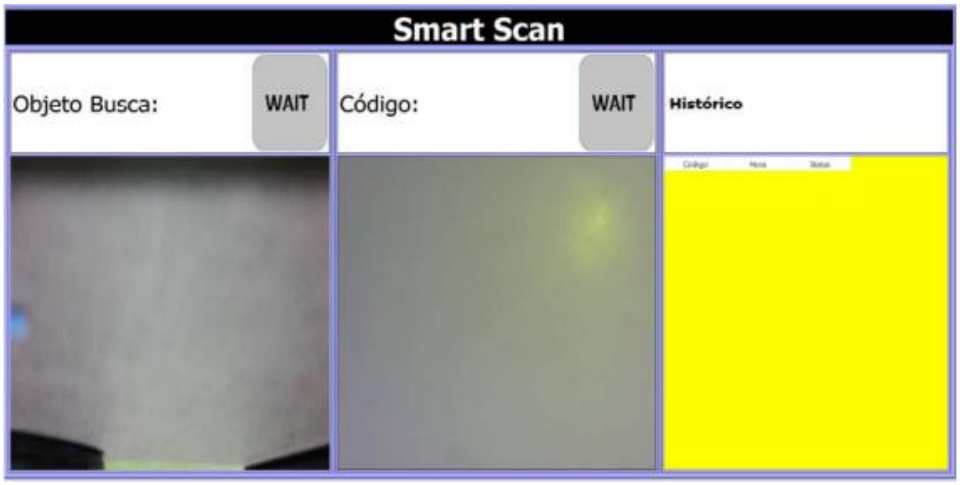
O projeto, implementado na indústria, propõe o desenvolvimento de um sistema baseado em inteligência artificial, utilizando um sistema de visão computacional em linguagem Python com o objetivo de resolver problemas de paradas de linha.

A etapa de software do sistema foi desenvolvida na plataforma Pycharm utilizando Python e TensorFlow e também foi desenvolvido uma interface utilizando PyQt, contendo recursos como imagens, botões, abas e tabelas. Os materiais utilizados foram duas Webcams, CLP, Cabo Ethernet, PC desktop, teclado, mouse e monitor.

A interface simples apresentada na Figura 14 tem como objetivo ser facilmente interpretada por qualquer funcionário, além de gerar um histórico com

rastreabilidade. O sistema desenvolvido recebeu grande destaque devido ao seu tempo de 3 segundos para inspeção e baixo custo comparando-se as soluções disponíveis no mercado que chegavam a preços de R$100.000,00 além do pagamento anual de licença. O projeto foi realizado com um orçamento abaixo de R$ 1000,00 na empresa onde foi aplicado. O resultado de detecção é apresentado na Figura 15.

Figura 14 – Interface da câmera



Fonte: MARQUES (2020)

Figura 15 – Resultado de detecção

Fonte: MARQUES (2020)

3.4 COMPARATIVO DOS RECURSOS OFERECIDOS PELOS PROJETOS (PROJETO 3)

A Tabela 1 faz um comparativo dos trabalhos de acordo com os seguintes parâmetros: Foco, Linguagem de programação, ambiente integrado de desenvolvimento de programas (IDE) e abordagem.

Tabela 1 - Características dos projetos

Parâmetros	**Projeto 1**	**Projeto 2**	**Projeto 3**
Foco	Área da saúde	Interação homem-máquina, acadêmico	Industria
Linguagem	Python/C/C++	Python	Python
IDE	Pycharm/Psoc Creator		Pycharm
Abordagem	Experimental e simulação	Simulação	Aplicação

Fonte: O autor (2022)

Os projetos listados demostram quais foram as ferramentas utilizadas em projetos utilizando TensorFlow. O Tensorflow possui APIs para C, C++, Java, Go, Rust, C# (TENSORFLOW, 2022). Observou-se que todos os projetos listados utilizaram Python, que pode ser justificada pela integração com as diversas bibliotecas disponíveis, que aumentam de forma significativa as possibilidades de aplicações.

4 MÉTODO E MATERIAIS UTILIZADOS

Esse tópico visa explicar as etapas para alcançar o objetivo do projeto, passando pelas etapas de:

1. Listagem de materiais
2. Instalação e configuração
3. Coleta de dados
4. Treinamento e teste

O desenvolvimento do projeto ocorreu no aplicativo web Jupyter Notebook, utilizando principalmente a linguagem Python para o desenvolvimento de um sistema capaz de identificar capacete para acesso às áreas restritas.

4.1 COMPONENTES

Para o desenvolvimento e teste de funcionalidade foram utilizados os materiais listados na Tabela 2.

Tabela 2 - Lista de Componentes

Item	Quantidade	Função
WebCam	1	Aquisição de imagem e detectar objetos de interesses
PC Desktop	1	Treinamento e teste dos sistemas de IA
Teclado	1	Interação com o usuário
Mouse	1	Interação com o usuário
Monitor	1	Exibir softwares
Arduino	1	Interação entre software e hardware
Resistor	1	Controle de corrente ao componente.
Protoboard	1	Base para montagem do circuito
LED	1	Simular dispositivo digital de controle de acesso

Fonte: O Autor (2022)

A WebCam disponível na Figura 16, foi utilizada na etapa anterior ao treinamento na aquisição de imagens para compor uma parte do banco de dados e na etapa de teste com vídeo.

Figura 16 – Webcam

Fonte: O autor (2023)

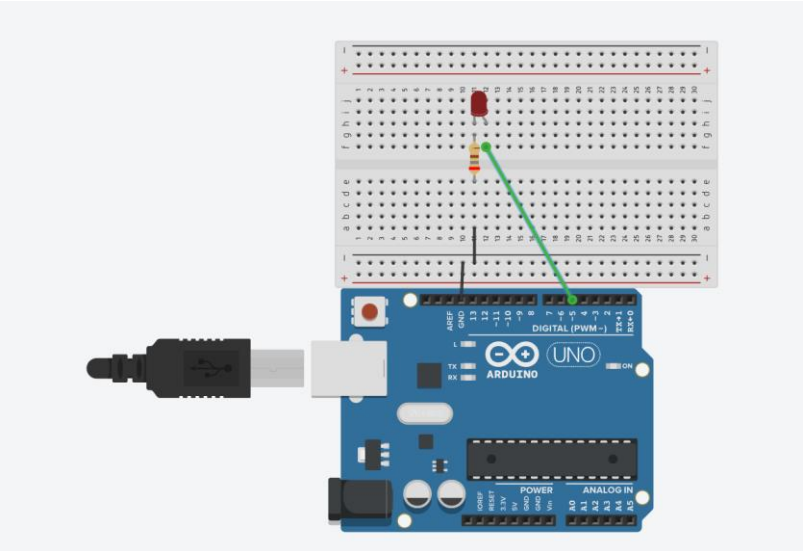
O dispositivo escolhido para os testes foi o PC Desktop como mostra a Figura 17, que possuí como especificação 16GB de memória RAM, 512GB no SSD e uma placa de vídeo RX 5500 de 8gb.

Figura 17 – PC Desktop

Fonte: O autor (2023)

A etapa de circuito conta com protoboard como mostra a Figura 18, resistor, LED e Arduino de acordo com a quantidade apresentada na Tabela 2. O circuito é responsável por simular um dispositivo digital, podendo ser esse dispositivo uma fechadura eletrônica.

Figura 18 – Circuito e hardware para simulação de acesso

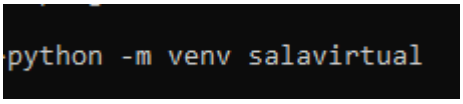


Fonte: Tinkercard (2023)

4.2 PREPARAÇÃO DO AMBIENTE

Os projetos que utilizam TensorFlow em Python, necessitam de diversas bibliotecas e em alguns casos, necessitam de versões especificas das bibliotecas, como no caso de aceleração de treinamento utilizando GPU. Uma etapa realizada no projeto foi criação de “ambiente virtual” através do módulo “venv”, de forma a isolar todas as bibliotecas baixadas para o projeto dos diretórios do sistema operacional. A utilização do ambiente virtual permite isolar a API de TensorFlow e evita que bibliotecas baixadas anteriormente afetem o treinamento e funcionamento do projeto. Para criar o ambiente virtual é necessário executar o código exibido na Figura 19.

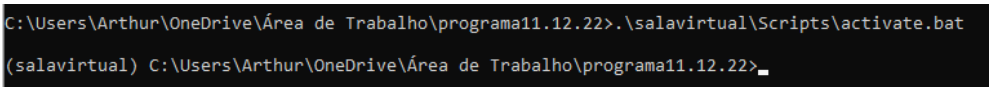
Figura 19 – Criação do ambiente virtual



Fonte: O Próprio Autor (2023)

Através do comando, criou-se um ambiente virtual chamado de “salavirtual”. A Figura 20 apresenta o comando de ativação e o nome do ambiente virtual no início da linha do Prompt de comando que sinaliza que o ambiente está ativo.

Figura 20 – Ativação do ambiente virtual



Fonte: O Próprio Autor (2023)

Após as etapas de criação e ativação, faz-se necessário adicionar o ambiente virtual ao Jupyter notebook. Para isso, foram executados os seguintes comandos: “pip install ipykernel” e “python -m ipykernel install --user --name=salavirtual”.

Em seguida, no ambiente virtual foi necessário instalar as bibliotecas em python para o desenvolvimento de projeto com TensorFlow. O primeiro passo foi instalar o Tensorflow, utilizando o comando “pip install tensorflow”. A API para TensorFlow para detecção de objetos está disponível no link: https://github.com/tensorflow/models. Também se fez necessário extrair os arquivos na pasta do projeto, junto ao ambiente virtual. O último passo para configuração do ambiente foi instalar os buffers de protocolo, desenvolvido pela google para serializar dados estruturados, em TensorFlow melhora o tráfego de dados processados. Está disponível no link: https://github.com/protocolbuffers/protobuf/releases. Após baixar os

buffers, adicionou-se o caminho do arquivo em variáveis do ambiente e então executou-se o código, como mostra a Figura 21.

Figura 21 – Instalação de *buffers* de protocolos

```
>protoc objetc_detection/protos/*.proto --python_out=.
```

Fonte: O Próprio Autor (2023)

A execução do código deve ocorrer dentro do ambiente virtual, na pasta em que os arquivos da API do Tensorflow foram disponibilizados, disponível em models/research.

4.3 BANCO DE IMAGENS

O projeto conta com dois conjuntos a serem classificados, a classe “com_capacete” e a classe “sem_capacete”. O sistema necessita de um conjunto de imagens para o treinamento e teste para cada classe. No conjunto de imagens “com_capacete”, é necessário fornecer imagens com o EPI e ao conjunto “sem_capacete”, é necessário fornecer imagens sem o EPI.

Para o conjunto de imagens com o EPI as imagens foram selecionadas dos bancos de dados vodan37, disponível no link https://www.kaggle.com/datasets/vodan37/yolo-helmethead. O banco de dados conta com imagens diversas de pessoas utilizando o equipamento de proteção como as disponíveis na Figura 22. Além disso, utilizou-se a coleta de imagens utilizando a biblioteca OpenCV e a WebCam e utilizando aparelho celular.

De acordo com Ludermir (2021), dados mais precisos geram generalizações mais precisas. O contrário também é verdade, fornecer dados inadequados geram resultado inadequados. Por esse motivo não foram utilizadas todas as imagens disponíveis no banco de dados, visto que algumas imagens não se apresentavam adequadas e essas podem atrapalhar o funcionamento da Inteligência artificial. Para o projeto, o tipo de dado interessante, são os casos em que a face e o EPI são exibidos, conforme a Figura 22.

Figura 22 – Imagem exemplar de uma pessoa com capacete

Fonte: VODAN37 (2021)

Referindo-se ao conjunto de imagens sem o Epi, repetiu-se os passos feitos no conjunto com o EPI. O banco de dados selecionado está disponível no link https://www.kaggle.com/datasets/ashwingupta3012/male-and-female-faces-dataset, o dado interessante para essa classe, são imagens em que a cabeça e a face são apresentadas, como mostra a Figura 23.

Figura 23 - Imagem exemplar de uma pessoa sem capacete

Fonte: KAGGLE (2021)

4.4 LABELIMG

O Labelimg é uma ferramenta de rotulagem de dados disponível em: https://github.com/tzutalin/labelImg. O aplicativo realiza a etapa de extração de características, permitindo que o usuário destaque o objeto de interesse nas imagens.

O aplicativo é um arquivo .py e para ser executado basta executar a linha de comando apresentada na Figura 24.

Figura 24 - Código para execução do labelimg

```
: LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')

: LABELIMG_PATH

: 'Tensorflow\\labelimg'

: !cd {LABELIMG_PATH} && python labelImg.py
```

Fonte: O autor (2023)

Na Figura 25, é exibido a tela inicial do aplicativo. Ao abri-lo, é necessário indicar o diretório pressionando “Open dir”, onde estão as imagens para realizar a extração de características e o caminho em que será salvo em “Change Save dir”. Após isso, pressionar “creat Rectbox” e selecionar o objeto de interesse e nomeá-lo de acordo com as classes e salvar.

Figura 25 - Labelimg

Fonte: O autor (2023)

O resultado dos passos executados no aplicativo será um arquivo em XML no formato pascal como mostra a Figura 26, contendo informações sobre: localização do objeto, nome do arquivo, caminho e classe.

Figura 26 – Arquivo em XML

homem_com_capacete - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
<annotation>
        <folder>Downloads</folder>
        <filename>homem_com_capacete.png</filename>
        <path>C:\Users\Arthur\Downloads\homem_com_capacete.png</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>602</width>
                <height>509</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>com_capacete</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>220</xmin>
                        <ymin>8</ymin>
                        <xmax>349</xmax>
                        <ymax>175</ymax>
                </bndbox>
        </object>
</annotation>
```

Fonte: O autor (2023)

A etapa de extração de dados foi realizada em um total de 2596 imagens, que são distribuídas em dois diretórios, o test e o train. O diretório test recebe as imagens para avaliar o funcionamento e o de train para o treinamento do AM. O projeto foi divido em dois sistemas como mostra a Tabela 3, o primeiro utilizou 683 imagens com EPI e 705 sem EPI para a pastar train e 206 com EPI e 207 sem EPI para a pasta test, ao segundo sistema adicionou-se mais imagens aos dados anteriores, resultando em 980 com EPI e 979 sem EPI para a train e 333 com EPI e 304 sem EPI para test.

Tabela 3 - Distribuição de dados

	TRAIN		TEST		TEST + TRAIN
Classes	**sistema 1**	**sistema 2**	**sistema 1**	**sistema 2**	**Total (sistema 2)**
Com EPI	683	980	206	333	1313
Sem EPI	705	979	207	304	1283

Fonte: O autor (2023)

4.5 FORMATAÇÃO DE DADOS

Para o treinamento faz-se necessário preparar alguns dados para serem processados pelo Tensorflow. Para o treinamento precisamos gerar a partir dos dados das pastas train e test um arquivo train.record e test.record e um arquivo com extensão ".pbtxt" que conterá as classes com_capacete e sem_capacete. Os arquivos nos formatos pbtxt e record são arquivos de texto Protocol Buffer, que são formatos entendidos pelo Tensorflow.

Para gerar o arquivo label_map.pbtxt em Python, utilizado-se a biblioteca os, que manipular arquivos no sistema operacional.

Após incluir a biblioteca utilizando o comando "import os", é possível indicar um caminho ou arquivo e armazenar em uma variável, também permite escrever informações dentro de arquivos. Ao executar o código da Figura 27, é criado o conteúdo para o arquivo label_map.pbtxt apresentado na Figura 28.

Figura 27 - Criação do arquivo label_map.pbtxt

```
files['LABELMAP']
```

```
'Tensorflow\\workspace\\annotations\\label_map.pbtxt'
```

```
labels = [{'name':'com_capacete', 'id':1}, {'name':'sem_capacete', 'id':2}]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

Fonte: O autor (2023)

Figura 28 - Label_map.pbtxt

```
item {
        name:'com_capacete'
        id:1
}
item {
        name:'sem_capacete'
        id:2
}
```

Fonte: O autor (2023)

Para criar os arquivos train.record e test.record, utilizou-se o código desenvolvido por Renotte, disponivel em: https://github.com/nicknochnack/GenerateTFRecord. O código converte dados de texto armazenados no arquivo .xml para .record. Ao executar o código contido no arquivo 'generate_tfrecord.py', nas pastas train e test, são criados os arquivos train.record e test.record, que estão escritos em binário como mostra a Figura 29.

Figura 29 – Formato .record

```
02f4 0000 0000 0000 ab15 849d 0afe e703
0a22 0a16 696d 6167 652f 6f62 6a65 6374
2f62 626f 782f 786d 6178 1208 1206 0a04
9a99 2f3f 0a22 0a16 696d 6167 652f 6f62
6a65 6374 2f62 626f 782f 786d 696e 1208
1206 0a04 0000 603e 0a15 0a0b 696d 6167
652f 7769 6474 6812 061a 040a 0280 050a
89e5 030a 0d69 6d61 6765 2f65 6e63 6f64
6564 12f6 e403 0af2 e403 0aee e403 ffd8
ffe0 0010 4a46 4946 0001 0101 0048 0048
0000 ffdb 0043 0006 0405 0605 0406 0605
0607 0706 080a 100a 0a09 090a 140e 0f0c
1017 1418 1817 1416 161a 1d25 1f1a 1b23
1c16 1620 2c20 2326 2729 2a29 191f 2d30
2d28 3025 2829 28ff db00 4301 0707 070a
080a 130a 0a13 281a 161a 2828 2828 2828
2828 2828 2828 2828 2828 2828 2828 2828
2828 2828 2828 2828 2828 2828 2828 2828
```

Fonte: O autor (2023)

4.6 TREINAMENTO DE INTELIGÊNCIA ARTIFICIAL

Para modelos de detecção de objeto, pode-se desenvolver um modelo de detecção do zero ou utilizar modelos pré-treinados. Os modelos pré-treinados, podem ser adaptados diversos modelos, outra vantagem são os dados a respeito de seu desempenho, ao escolher um modelo, estão disponíveis informações como a velocidade e precisão do modelo. Para o treinamento utilizou-se o modelo pré-treinado SSD MobileNet V2 FPNLite 320x320, disponível em: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md. A abordagem utilizando Single Shot MultiBox Dectector(SSD), necessita que para cada imagem exista uma caixa delimitando o objeto que será classificado durante o treinamento, como mostra a Figura 25 apresentada anteriormente (LIU, Wei et al,2016).

Ao baixar o modelo pré-treinado, deve-se extrair o arquivo pipeline.config na pasta em que será armazenado o modelo treinado, o modelo pré-treinado é genérico para adaptá-lo ao projeto é necessário fornecer algumas informações dos arquivos formatados anteriormente, como o número de classes, caminhos dos arquivos train.record e test.record, como mostra a Figura 30.

Figura 30 - Arquivos dentro do pipeline

```
train_input_reader {
  label_map_path: "Tensorflow\\workspace\\annotations\\label_map.pbtxt"
  tf_record_input_reader {
    input_path: "Tensorflow\\workspace\\annotations\\train.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "Tensorflow\\workspace\\annotations\\label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "Tensorflow\\workspace\\annotations\\test.record"
  }
}
```

Fonte: O autor (2023)

Após configurar o modelo pré-treinado, tem-se todos os dados necessário para treinar o sistema, A Figura 31 mostra o comando em que inicia o treinamento.

Figura 31 – Comando de treinamento

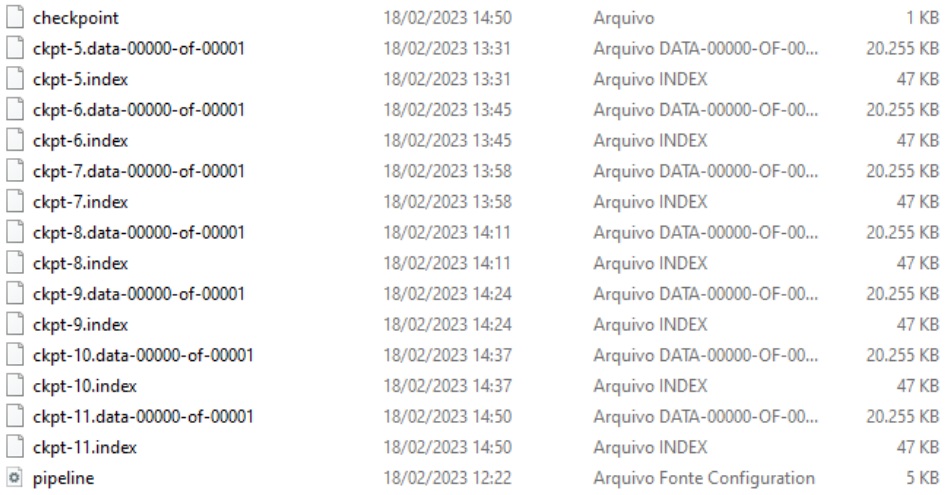
```
python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace
\models\modelo_final4 --pipeline_config_path=Tensorflow\workspace\models\modelo_final4\pipeline.confi
g --num_train_steps=10000
```

Fonte: O autor (2023)

O comando inicia um código em Python com o nome de ‘‘model_main_tf2.py’ disponibilizado na API oficial do Tensorflow para detecção de objeto, disponível em: https://github.com/tensorflow/models/tree/master/research/object_detection. Além do código é necessário indicar o diretório em que será salvo os arquivos de treinamento através de ‘’--model_dir=’, o diretório do arquivo pré-treinado, através do comando ‘--pipeline_config_path’ e a quantidade de ‘steps’ em que o treinamento ocorrerá através do comando ‘--num_train_steps’ de acordo com a Figura 31.

Ao todo foram utilizados 10000 steps de treinamento, que geram checkpoints, salvando a evolução do treinamento, sendo possível avaliar o desempenho em várias etapas até o modelo final, de acordo com a Figura 32.

Figura 32 – Diretório do projeto



Fonte: O autor (2023)

5 RESULTADOS E DISCUSSÕES

Está etapa tem como objetivo avaliar os resultados obtidos dos treinamentos dos sistemas 1 e 2 na Tabela 3, onde o sistema 1 utilizou 683 imagens com o EPI e 705 sem o EPI para o treinamento, ao sistema 2 adicionou-se mais imagens, resultando em 980 com EPI e 979 sem EPI.

Os testes utilizaram-se imagens iguais para avaliar o desempenho comparando a precisão que o sistema está detectando o objeto a ser detectado, bem como serão realizados teste de detecção em tempo real.

5.1 TESTE COM IMAGEM

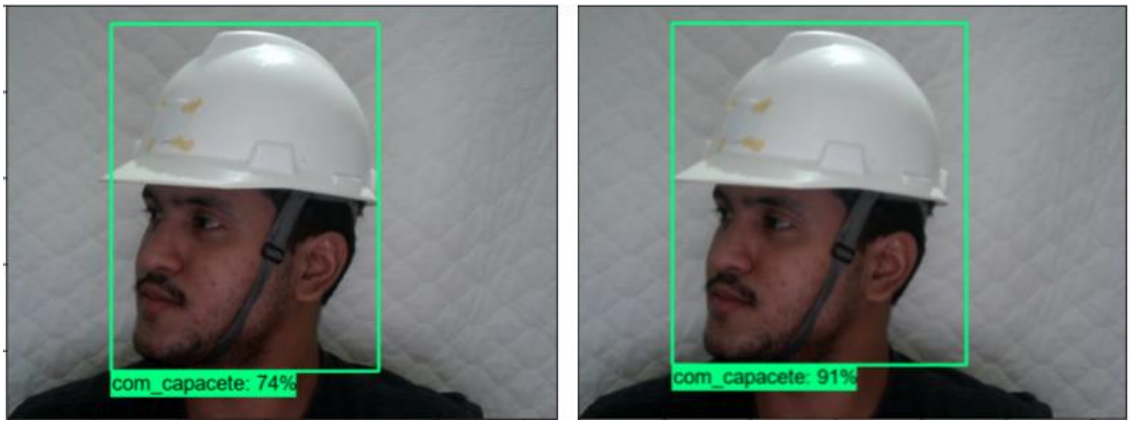
A Figura 33 compara o desempenho do sistema 1 e o sistema 2. Foi possível observar que o desempenho do sistema 2 apresentou maior precisão. De acordo com Ludemir (2021), os sistemas de aprendizado de máquina são orientados a dados e aprendem a partir de grandes volumes de dados.

Figura 33 - Sistema 1 x sistema 2 sem EPI

Fonte: O Próprio Autor (2022)

A Figura 34 apresenta a comparação dos sistemas 1 e 2 com EPI.

Figura 34 - Sistema 1 x sistema 2 com EPI



Fonte: O Próprio Autor (2022)

Notou-se que o sistema 2 apresentou maior precisão por ter maior volume de dados.

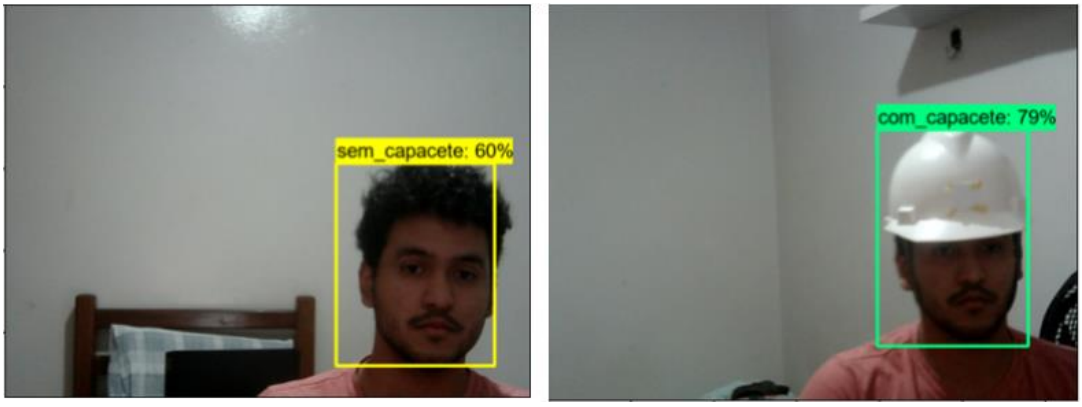
Os sistemas mostraram-se eficientes em detectar chapéu como a classe sem EPI, como mostra a figura 35.

Figura 35 - Sistema 1 x sistema 2 com chapéu

Fonte: O Próprio Autor (2021)

Segundo LIU, Wei et al (2016), a posição no campo de observação da câmera também importa, sendo uma característica do sistema SSD que foi utilizado no projeto. A Figura 36 demostra que os dados de treinamentos fornecidos não possuem muitos exemplares no qual o objeto de detecção está nos cantos da imagem, então ao avaliar a imagem os resultados tendem a ter uma menor precisão.

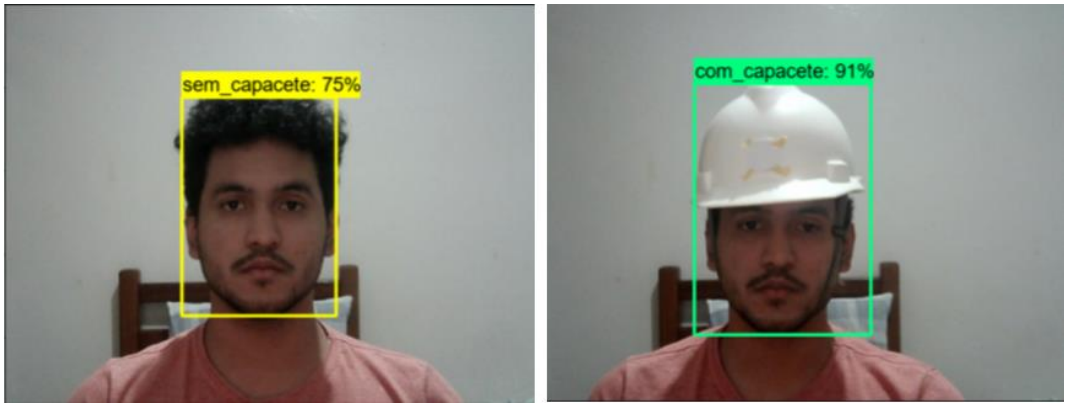
Figura 36 - Detecção com imagem descentralizada no sistema 2



Fonte: O Próprio Autor (2021)

O projeto visa permitir o acesso a áreas restritas. A maior parte das imagens de treinamento fornecem dados em que o objeto de detecção está centralizado. Imagens que se enquadram nesse conjunto possuem resultados melhores como mostra a figura 37.

Figura 37 - Detecção centralizada no sistema 2

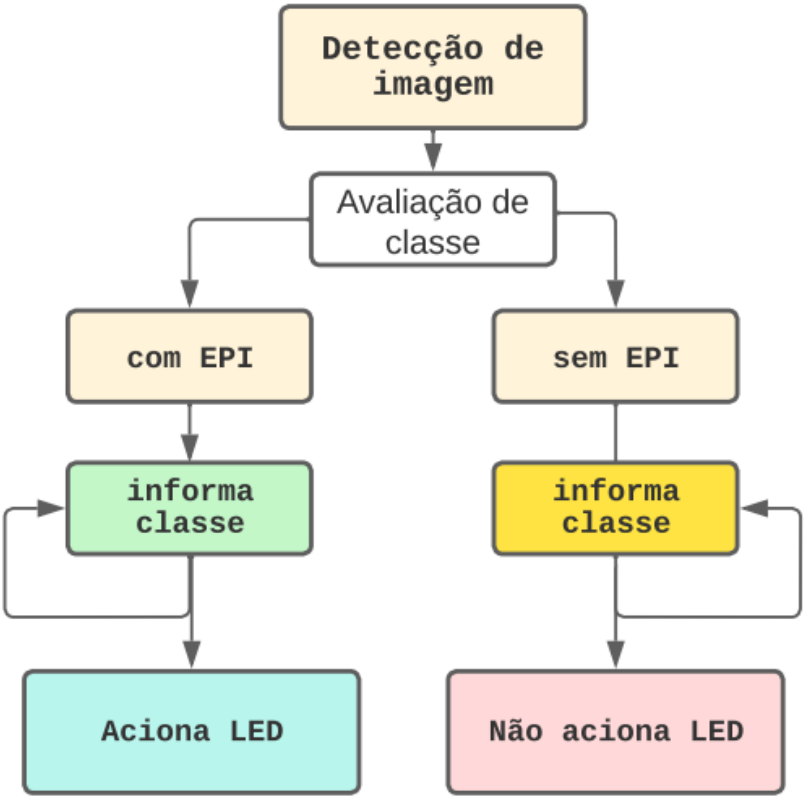


Fonte: O Próprio Autor (2021)

5.2 FUNCIONAMENTO DO SISTEMA

O código que realiza a detecção em tempo real, disponível no Anexo 1, funciona de acordo com a figura 38.

Figura 38 - Diagrama do sistema de detecção



Fonte: O Próprio Autor (2021)

A detecção em tempo real teve resultado satisfatório apenas no sistema 2. O sistema 1 um possuí dificuldade em detectar o EPI, de forma constante, pois o código de avaliação em tempo real funciona através de laço de repetição, ou seja, avalia infinitamente o status enquanto estiver ativo. Para permitir ou negar o acesso, adicionou-se uma variável de incremento para que o acionamento do LED ocorresse apenas após quinze ciclos de detecção bem-sucedidas e uma precisão acima de 80%.

6 CONCLUSÕES FINAIS

Este trabalho, teve como objetivo desenvolver um sistema capaz de avaliar se está ou não sendo feito o uso capacete para acessar áreas restritas, de forma a garantir a segurança dos trabalhadores automatizando o controle de acesso e facilitar adequação das empresas e dos trabalhadores as normas de segurança do trabalho, pois de acordo com a norma é função de ambos garantir que o trabalho seja realizado de forma segura.

Para isso utilizaram-se equipamentos de baixo custo como teclado, mouse, Webcam de computador, desktop pessoal, LED, resistor e protoboard e ferramentas gratuitas como Tensorflow, Python, Jupyter Notebook, banco de dados.

O projeto foi desenvolvido em três etapas principais: Seleção de dados, treinamento e teste. A etapa que exigiu maior tempo foi a seleção de dados, pois não existe um banco de dados adequado para o projeto desenvolvido. Além disso realizou-se a etapa de detecção em tempo real, na qual de fato realiza o controle, através do acionamento após detectar que o funcionário está com o EPI.

O resultado foi satisfatório, pois o projeto cumpriu a função de realizar o controle de acesso. Foram realizados dois sistemas onde o primeiro sistema possui o menor volume de dados e o segundo possui os dados do primeiro sistema com a adição de mais dados. O incremento de dados permitiu a comparação dos sistemas, onde o sistema 2 apresentou-se mais eficiente e preciso em ambas as classes. No sistema dois avaliou-se o desempenho da arquitetura utilizada baseada em SSD. As comparações realizadas reforçaram que quanto maior o volume de dados adequados, melhor é a precisão do sistema.

Como trabalho futuros, pretende-se o desenvolvimento em minicomputadores de placa única como Raspberry, na qual é suportada pelo Tensorflow. Planeja-se também o incremento do banco de dados e o desenvolvimento de um outro sistema que monitore o ambiente de trabalho e gere dados de como o equipamento de proteção individual está sendo utilizado no ambiente de acesso restrito.

REFERÊNCIAS BIBLIOGRÁFICAS

ALPAYDIN, Ethem. **Introduction to machine learning**. MIT press, 2020. Acesso em: 27 de nov. 2022.

ARDUINO, **Hardware**, 2022. Disponível em <https://docs.arduino.cc/hardware/uno-rev3. Acesso em: 27 de nov. 2022.

AYODELE, Taiwo Oladipupo. **Types of machine learning algorithms**. New advances in machine learning, v. 3, p. 19-48, 2010. Acesso em 27 de nov. 2022.

BARELLI, F. **Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV**. Casa do Código, 2018. ISBN 9788594188588. Disponível em: .<https://books.google.com.br/books?id=CA5ZDwAAQBAJ>. Acesso out 30 de 2022.

BATISTA, Gustavo Enrique de Almeida Prado Alves**. Pré-processamento de dados em aprendizado de máquina supervisionado**. 2003. Tese (Doutorado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, University of São Paulo, São Carlos, 2003. doi:10.11606/T.55.2003.tde-06102003-160219. Acesso em 11 de nov de 2022.

BEIS, 1999. **Indexing Without Invariants in 3D Object Recognition**. Acesso em: 30 de dez. 2022.

BORGES, Luiz Eduardo. **Python para desenvolvedores: aborda Python 3.3**. Novatec Editora, 2014. Acesso em 17 de nov de 2022.

CHAPELLE, Olivier; SCHOLKOPF, Bernhard; ZIEN, Alexander. **Semi-supervised learning** (chapelle, o. et al., eds.; 2006)[book reviews]. IEEE Transactions on Neural Networks, v. 20, n. 3, p. 542-542, 2009. Acesso em 16 de nov de 2022.

DA SILVA, Josenildo C.; VIEIRA, Raimundo Osvaldo. **Introdução às Redes Neurais Profundas com Python**. Sociedade Brasileira de Computação, 2022. Acesso em 17 de nov de 2022.

D'ADDARIO, Miguel. **Inteligência Artificial: Tratados, aplicações, usos e futuro.** Babelcube Inc., 2022. Acesso em: 28 de dez de 2022.

FENG, S. et al. **Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment.** Nat Commun, v.12, p.748, 2021. Disponível em: <https://doi.org/10.1038/s41467-021-21007-8>. Acesso em 16 de nov de 2022.

FERNANDES, Fernando Timoteo; CHIAVEGATTO FILHO, Alexandre Dias Porto. **Perspectivas do uso de mineração de dados e aprendizado de máquina em saúde e segurança no trabalho**. Revista Brasileira de Saúde Ocupacional, v. 44, 2019. Acesso em: 27 de dez.2022.

FERREIRA, Michael Henrique Souza. **Reconhecimento facial para detecção de emoções utilizando redes neurais convolucionais com tensorflow**. 2021. Acesso em: 23 de dez de 2022.

GONÇALVES, Steffani Bez Batti et al. **Prevalência e fatores associados aos acidentes de trabalho em uma indústria metalmecânica**. Revista Brasileira de Medicina do Trabalho, v. 16, n. 1, p. 26-35, 2018. Acesso em: 19 de jan.2023.

GONZAGA, Lucas Augusto. **Aplicações da visão computacional utilizando Python**. 2017. 118 f. . Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) – Universidade Federal de Uberlândia, Uberlândia, 2017. Disponível em: < https://repositorio.ufu.br/handle/123456789/25519>. Acesso em: out 30 de 2022.

HUANG, Hao. **Implementação de algoritmos de TensorFlow™ para detetar patologias cardíacas**. 2019. Tese de Mestrado. Acesso em: 23 de dez de 2022.

IZBICKI, Rafael; SANTOS, Tiago Mendonça dos. **Aprendizado de máquina: uma abordagem estatística**. 1. ed. São Carlos, SP: [s.n.], 2020. ISBN: 978-65-0002-410-4. Acesso em 17 de nov de 2022.

LIU, Wei et al. Ssd: **Single shot multibox detector. In: Computer Vision–ECCV** 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016. p. 21-37. Acesso em: 24 de dez.2022.

LUDERMIR, Teresa Bernarda. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências**. Estudos Avançados, v. 35, p. 85-94, 2021. Acesso em 4 de dez de 2022. Acesso em: 29 de dez. 2022.

Marques, Lucas**. Sistema de detecção de objeto personalizado utilizando inteligência artificial**. 2020. 66 f. (Graduação em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia do Amazonas – Campus Manaus Distrito Industrial. Manaus, 2020. Disponível em <http://repositorio.ifam.edu.br/jspui/handle/4321/645>. Acesso em 28 de dez de 2022.

Ministério do Trabalho e Previdência. **Normas Regulamentadoras – NR**. Governo do Brasil, 2023. Disponível em: https://www.gov.br/. Acesso em: 29 jan. 2023.

Ministério do Trabalho e Previdência. **Quantidade de acidentes do trabalho, por situação do registro e motivo, segundo a parte do corpo atingida – 2019**. 2022. Governo do Brasil, 2022. Disponível em: https://www.gov.br/. Acesso em: 18 de fev. 2023.

OPENCV. **About**, 2022. Disponível em: https://opencv.org/about/. Acesso em: 27 de nov. 2022.

PYTHON, **HOME**. Disponível em: https://www.python.org/. Acesso em: 27 de nov. 2022.

RANDLES, Bernadette M. et al**. Using the Jupyter notebook as a tool for open science: An empirical study**. In: 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL). IEEE, 2017. p. 1-2. Acesso em 17 de nov. 2022.

RUDEK, M.; Coelho, L. S.; Canciglieri J. O., **Visão Computacional Aplicada a Sistemas Produtivos: Fundamentos e Estudo de Caso**. In: XXI Encontro Nacional De Engenharia De Produção, 2001. Disponível em < https://www.researchgate.net/publication/228549623_Visao_Computacional_Aplicada_a_Sistemas_Produtivos_Fundamentos_e_Estudo_de_Caso>. Acesso em 31 out de 2022.

S. Aradi, "**Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles," in IEEE Transactions on Intelligent Transportation Systems**, vol. 23, no. 2, pp. 740-759, Feb. 2022. Disponível em <doi: 10.1109/TITS.2020.3024655>. Acesso em: 27 de nov. 2022.

TENSORFLOW. **Por que o TensorFlow**, 2022. Disponível em <https://www.tensorflow.org/about/case-studies>. Acesso em: 27 de nov. 2022.

TORFI, A. et al. **Natural language processing advancements by deep learning: A survey**. arXiv preprint arXiv:2003.01200 (2020). Acesso em 16 de nov de 2022.

TST. **O que é acidente do trabalho?** [s.d.]. Disponível em: https://www.tst.jus.br/web/trabalhoseguro/o-que-e-acidente-de-trabalho. Acesso em: 18 de fev. 2023.

ZHU, Xiaojin Jerry. **Semi-supervised learning literature survey**. 2005. Acesso em: 20 de nov 2022.

APÊNDICE A – Código de detecção em tempo real

```
#Inclusão das bibliotecas necessárias
import os
import pyfirmata
import time
import object_detection
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
#Configuração de para comunicação entre Python e Arduino
pin = 13
port = 'COM3'
board = pyfirmata.Arduino(port)
y=0



#Utilizando a biblioteca os, configuração dos diretórios do projeto
CUSTOM_MODEL_NAME = 'modelo_final4'
LABEL_MAP_NAME = 'label_map.pbtxt'
paths = {
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME)
 }
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models',
CUSTOM_MODEL_NAME, 'pipeline.config'),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}




# Carrega o modelo treinado ao código
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'],
is_training=False)


# Carrega o checkpoint desejado
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
```

```
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-
11')).expect_partial()



@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections


#Configurações iniciais para detecção em tempo real
import cv2
import numpy as np
from matplotlib import pyplot as plt


category_index =
label_map_util.create_category_index_from_labelmap(files['LABELMAP'])



cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))


#Loop para avaliação de classe
while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections


    detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)
```

```
    label_id_offset = 1
    image_np_with_detections = image_np.copy()


    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=1,
                min_score_thresh=.75,
                agnostic_mode=False)


    cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800,
600)))
    #avaliação de classe
    # x[0] = 0 x[1] = 1, com capacete, variável de controle de acesso.
    x = detections['detection_classes']
    z = detections['detection_scores']
    print(x[0],x[1])
    board.digital[13].write(0)


    if x[0] == 0 and x[1] == 1:
        y= y+1

    if x[0] == 1 or x[1] == 0 or z[0]<.80:
        y = 0
    if y>15:
        board.digital[13].write(1)


    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
```