**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL**

**ESCOLA POLITÉCNICA**

**ENGENHARIA DE COMPUTAÇÃO**

**Augusto Gabriel de Lara**

**Renan de Souza Silva**

# CONTROLE DE UTILIZAÇÃO DE EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL ATRAVÉS DO RECONHECIMENTO DE IMAGENS POR INTELIGÊNCIA ARTIFICIAL

Porto Alegre

2022

AUGUSTO GABRIEL DE LARA
RENAN DE SOUZA SILVA

# CONTROLE DE UTILIZAÇÃO DE EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL ATRAVÉS DO RECONHECIMENTO DE IMAGENS POR INTELIGÊNCIA ARTIFICIAL

Trabalho de conclusão de curso de graduação apresentado na Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Engenheiro de Computação.

**Orientador: Denis Fernandes**

Porto Alegre
2022.

## RESUMO

O uso de equipamentos de proteção individual (EPI) nas indústrias e nos canteiros de construções torna-se cada vez mais imprescindível para amenizar os danos que eventuais acidentes de trabalho possam causar nos trabalhadores, com isso, é essencial o controle da utilização de EPIs visando deixar o ambiente de trabalho mais seguro. Portanto, este trabalho tem como objetivo propor, com base em referências bibliográficas e do uso da tecnologia de Redes Neurais Artificiais utilizando o algoritmo de reconhecimento e treinamento YOLOv5, um modelo de um sistema que possa processar a RNA com o minicomputador Raspeberry Pi obtendo as imagens através de uma *webcam*, para assim fiscalizar a utilização dos EPI's pelos trabalhadores.



**Palavras-chave**: Equipamentos de Proteção Individual; EPI; Segurança; Redes Neurais; Inteligência Artificial; YOLO.

## ABSTRACT

*The use of personal protective equipment (PPE) in industries and on construction sites has become increasingly essential to mitigate the damage that eventual work accidents may cause to workers, so it is essential to control the use of PPE to the safest working environment. Therefore, this work aims to propose, based on the literature and the use of Artificial Neural Networks technology using the YOLOv5 recognition and training algorithm, a system model that can process the ANN with the Raspeberry Pi minicomputer obtaining the images through a webcam, to monitor the use of PPE by workers.*



***Keywords****: Personal Protective Equipment; PPE; Security; Neural Networks; Artificial Intelligence;* YOLO.

## LISTA DE ILUSTRAÇÕES

Figura 1 - Categorização dos segmentos de IA ..... 12
Figura 2 - Subcampos da IA ..... 13
Figura 3 - Camadas de uma CNN ..... 15
Figura 4 - Arquitetura do YOLOv5 ..... 17
Figura 5 - Sistema embarcado de detecção de EPI's ..... 23
Figura 6 - Exemplo de matriz de confusão ..... 26
Figura 7 - Matriz de confusão gerada ..... 28
Figura 8 - Gráfico de Precisão ..... 29
Figura 9 - Gráfico de Revocação ..... 29
Figura 10 - Gráfico de *F-Score* ..... 30
Figura 11 - Gráfico de mAP ..... 30
Figura 12 - Exemplo de detecção ..... 31
Figura 13 - Exemplo de detecção ..... 31

## LISTA DE TABELAS

Tabela 1 - Classificação de cores dos capacetes.....11
Tabela 2 - Descrição das classes.....20
Tabela 3 - Lista dos Hiperparâmetros .....22
Tabela 4 - Percentuais obtidos.....27
Tabela 5 - Percentuais obtidos por Ferdous M e Ahsan SMM.....32

## LISTA DE SIGLAS

EPI – Equipamento de Proteção Individual.

IA – Inteligência Artificial.

RNA – Rede Neural Artificial.

YOLO – *You Only Look Once*.

RGB – *Red, Green, Blue*

R-CNN – *Region Based Convolutional Neural Network.*

CNN – *Convolutional Neural Network*.

FPS – *Frames per Second*

P – Precisão.

R – Revocação.

AP – *Average Precision* (Precisão Média).

mAP – *Mean Average Precision*.

VM – *Virtual Machine*

**SUMÁRIO**

**1 INTRODUÇÃO ..... 9**
**2 FUNDAMENTAÇÃO TEÓRICA ..... 10**
2.1 EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL ..... 10
2.2 INTELIGÊNCIA ARTIFICIAL ..... 11
**2.2.1 REDES NEURAIS E APRENDIZADO DAS MÁQUINAS ..... 13**
2.3 YOLO ..... 15
2.4 SISTEMAS EMBARCADOS ..... 17
**2.4.1 RASPBERRY PI ..... 18**
**3 DESENVOLVIMENTO ..... 19**
3.1 **ARQUITETURA DO PROJETO** ..... 19
**3.1.1 BANCO DE IMAGENS ..... 19**
**3.1.2 PREPARANDO AS IMAGENS PARA O TREINAMENTO ..... 20**
**3.1.3 EXECUTANDO ALGORITMO PARA CRIAÇÃO DA RNA ..... 21**
**3.1.4 EMBARCANDO A RNA NA RASPEBERRY PI ..... 23**
**4 APLICAÇÕES E RESULTADOS ..... 24**
4.1 MÉTRICAS DE AVALIAÇÃO ..... 24
**4.1.1 PRECISÃO, REVOCAÇÃO E *F-SCORE* ..... 24**
**4.1.2 *AVERAGE PRECISION* E *MEAN AVERAGE PRECISION* ..... 25**
4.2 MATRIZ DE CONFUSÃO ..... 26
4.3 RESULTADOS OBTIDOS ..... 27
4.4 COMPARAÇÃO COM OUTROS TRABALHOS ..... 32
**5 CONCLUSÃO ..... 33**
**REFERÊNCIAS ..... 34**

# 1 INTRODUÇÃO

Atualmente podemos dizer que a tecnologia está presente e auxiliando cada vez mais no nosso dia-a-dia, desde o momento em que acordamos com a ajuda de despertadores e/ou assistentes virtuais que utilizam IoT, sem falar dos *smartphones* que nada mais são do que computadores pessoais poderosos e pequenos que estão conosco todo o tempo.

Mas além de observarmos a tecnologia em nossas vidas pessoais, podemos vê-la cada vez mais em nossos ambientes de trabalho, como nas indústrias, nas escolas, nos canteiros de obras etc. Essas tecnologias estão sendo instaladas para auxiliar colaboradores nas empresas, principalmente na prevenção de acidentes de trabalho, como por exemplo, adaptando sensores nas máquinas. Também temos visto um aumento de tecnologias na área da segurança do trabalho, onde estas auxiliam na fiscalização dos órgãos responsáveis por supervisionarem o uso de EPI's, EPC's e manutenções preventivas de equipamentos.

Muitas destas tecnologias estão baseadas no uso de Redes Neurais Artificiais, que automatizam com eficácia o trabalho humano, inibindo erros e distrações por exemplo. Podemos ver um grande avanço nesse campo, com o aperfeiçoamento dos algoritmos utilizados no treinamento das máquinas para se tornarem cada vez mais eficientes.

Com as Redes Neurais Artificiais, podemos utilizá-las no campo da robótica, por exemplo, para executar tarefas de alta periculosidade dentro de uma indústria ou até menos podemos utilizar o reconhecimento de imagens para detectarmos focos de incêndio.

Por ser um ramo em crescente expansão de recursos e que se acredita ser de alta importância, decidiu-se assim realizar um projeto voltado para este assunto, com o intuito de auxiliar na supervisão do uso de EPI's para a área da construção civil, utilizando uma Rede Neural Artificial com o objetivo de reconhecimento de imagens.

# 2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão abordadas explicações teóricas sobre os diversos temas presentes neste trabalho.

## 2.1 EQUIPAMENTOS DE PROTEÇÃO INDIVIDUAL

Nas indústrias assim como em outros lugares que em algum momento o trabalhador possa realizar alguma atividade que gere potencial risco a sua saúde ou até mesmo possa lhe causar risco de vida, foi visto que para se amenizar o dano que esses eventuais acidentes de trabalho poderiam causar, precisaria ser criado algum mecanismo de segurança para os trabalhadores. Com isso surgiu a criação do Equipamento de Proteção Individual (EPI).

O EPI é todo dispositivo ou equipamento utilizado pelo profissional individualmente, que visam a proteção dele ao desempenhar suas atividades trabalhistas que se enquadrem em algum tipo de risco (PANTALEÃO, 2019).

A utilização destes equipamentos deverá ser feita quando não for possível tomar medidas de proteção coletivas no ambiente de trabalho que sejam viáveis, eficientes e suficientes para eliminarem os riscos ali presentes.

A utilização de EPI pelos trabalhadores foi oficializada pela Norma Regulamentadora 6 (Ministério do Trabalho e Previdência, 2020). Pela norma, temos a seguinte definição:

> 6.1 Para os fins de aplicação desta Norma Regulamentadora - NR, considera-se Equipamento de Proteção Individual - EPI, todo dispositivo ou produto, de uso individual utilizado pelo trabalhador, destinado à proteção de riscos suscetíveis de ameaçar a segurança e a saúde no trabalho.

Existem diferentes classes para os equipamentos de proteção individual, onde as empresas podem disponibilizar uma ou mais dependendo das atividades desempenhadas. Segue abaixo a lista com as classes de proteção e exemplos de EPI’s utilizados em cada uma.

- Proteção Auditiva: abafadores de ruídos, protetores auriculares;
- Proteção Respiratória: máscaras, filtros;
- Proteção Visual: óculos, viseiras;
- Proteção da Cabeça: capacetes;
- Proteção das Mãos e Braços: luvas, mangotes;

- Proteção de Pernas e Pés: sapatos, botinas;
- Proteção Contra Quedas: cintos, cinturões;

Para nosso projeto, selecionamos apenas alguns EPI’s para serem reconhecidos, eles são: óculos; capacetes, sendo estes variados pelas cores azul, vermelha, amarela e branca; e coletes.

Quanto as cores dos capacetes, estas servem não apenas por estética, mas para diferenciar o profissional com sua atribuição e tarefa desempenhada. Veja a descrição de cada cor na Tabela 1.

**Tabela 1 -** Classificação de cores dos capacetes

| Classe | Descrição |
|---|---|
| Branco e Cinza | Mestres de obra, encarregados e engenheiros. |
| Verde | Operários e serventes. |
| Vermelho | Carpinteiros. |
| Preto | Técnicos em segurança do trabalho. |
| Marrom ou Amarelo | Visitantes. |
| Azul | Pedreiros. |
| Laranja | Eletricistas e profissionais de elétrica. |

## 2.2 INTELIGÊNCIA ARTIFICIAL

O termo denotado de Inteligência Artificial (IA) foi criado em 1956 por John McCarthy, um dos fundadores da área de ciência da computação, que o descreveu como uma ciência e engenharia capaz de construir e tornar máquinas inteligentes, em especial computadores inteligentes. Em outras palavras, podemos dizer que IA é a inteligência exibida por máquinas em suas funções, quando estas simulam tarefas associadas a mente humana, como aprendizagem e resolução de problemas.

Podemos descrever sistemas de IA em um agrupamento de quatro dimensões (Russell e Norvig, 2004). Essas dimensões seguem as seguintes abordagens: sistemas que pensam como seres humanos; sistemas que atuam como seres humanos; sistemas que pensam racionalmente; e sistemas que atuam racionalmente. Veja essa categorização na Figura 1.

**Figura 1 -** Categorização dos segmentos de IA


Fonte: Baseado em Luciano Frontino de Medeiros, 2018.

Apesar do termo IA ser muito discutido e procurado dentro do universo da computação, além de ter tido sua origem nessa área também, pode-se dizer que é um tema multidisciplinar. A inteligência artificial é encontrada nas áreas de computação, engenharia, psicologia, filosofia, economia, neurociência, matemática e cibernética, onde o principal objetivo desempenhado é o de construir sistemas que apresentem comportamento inteligente e desempenhem funções com um grau similar ou superior ao de um especialista humano de determinada área.

Dentro do conceito geral da IA, ainda podemos dividir o tema em subcampos, onde cada um possui e desempenha especialidades diferentes. Esses subcampos podem ser divididos em: resolução de problemas; processamento de linguagem natural; robótica; sistemas baseados no conhecimento; processamento de imagem; aprendizagem de máquina; algoritmos genéticos; e redes neurais. Veja os subcampos na Figura 2.

**Figura 2 -** Subcampos da IA


Fonte: Baseado em Liebowitz, 1997; Dym, 1994; Siriram, 1997; Mao e Reddy, 2003.

Vale ressaltar que para o desenvolvimento deste projeto, foi utilizado mais de um subcampo de IA, que foram o Aprendizado de Máquina (*Machine Learning*), Redes Neurais e Processamento de Imagem.

### 2.2.1 REDES NEURAIS E APRENDIZADO DAS MÁQUINAS

Ao falarmos de IA temos um assunto que acaba sendo muito visto e discutido, que são as Redes Neurais Artificiais (RNA's). Podemos dizer que as RNA's são um conceito da computação que visa trabalhar no processamento de dados de maneira semelhante ao cérebro humano, onde temos neurônios trabalhando de forma paralela. Por ter sua arquitetura semelhante à dos neurônios, podemos dizer que se destina a reproduzir o aprendizado por meio do desenvolvimento de sistemas que aprendem com exemplos de treinamento (Alves, 2020). As RNA's são um subconjunto do aprendizado de máquina e estão no centro dos algoritmos de *Deep Learning*, ou seja, realizam o trabalho de aprendizado das máquinas onde possuem uma capacidade de reconhecimento de padrões complexos ou numerosos.

A etapa de treinamento das RNA's conhecida como *Machine Learning* se baseia na ideia onde sistemas possam aprender com os diferentes dados recebidos em suas entradas, identificando padrões e tomando decisões adequadas para cada conjunto de dado, com o mínimo de contato humano (SAS Insights, 2022).

Ao treinarmos uma determinada RNA conseguimos então realizar o processo de identificação de objetos em imagens, por exemplo. Esse processo se caracteriza não somente pela identificação do objeto, mas vai além nos mostrando com precisão a localização que tal objeto se encontra no frame capturado.

Dentro das classes abordadas nas RNA's, a mais indicada para a identificação de objetos é a convolucional, conhecida como *Convolutional Neural Network* (CNN). As CNN's são algoritmos de aprendizado profundo que tem esse nome, pois utilizam a operação matemática da convolução no lugar da multiplicação geral de matrizes em ao menos uma de suas camadas (WIKIPIDEA, 2022). A arquitetura do algoritmo consiste na camada de entrada, nas camadas ocultas (onde se encontra a camada convolucional) e camada de saída. Cada camada recebe uma entrada de dados, onde aplica filtros matriciais e operações, gerando uma saída para a camada seguinte. Ao passarmos uma imagem na entrada, ela irá recebendo caixas delimitadoras e pesos para distinguir os diferentes objetos presentes nela (ALVES, 2018).

Atualmente podemos observar que as CNN's possuem três camadas básicas para a detecção assertiva dos objetos nas imagens, são essas: as camadas de convolução (podendo haver diversas dentro de uma CNN); a camada de *pooling*; e a camada totalmente conectada. Abaixo iremos descrever em maiores detalhes sobre cada uma dessas camadas (INSIGHT, 2021).

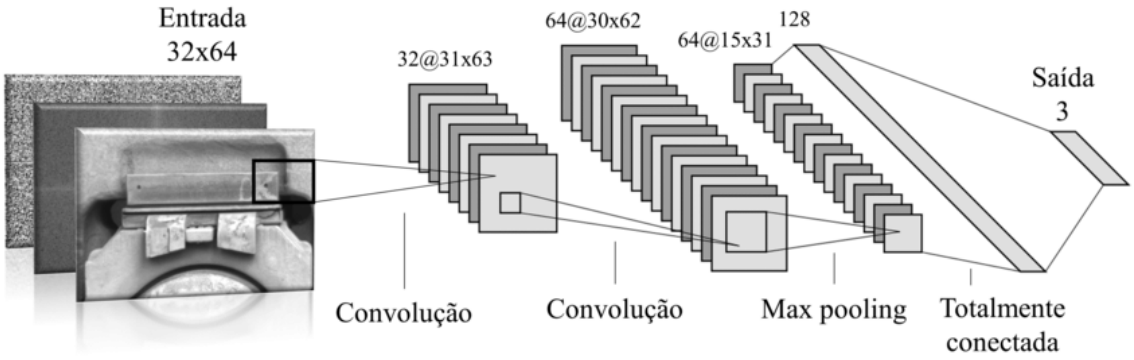
As camadas de convolução possuem como objetivo principal realizar a operação da convolução sobre a entrada e repassar para a próxima camada o resultado. Podemos assemelhar seu processo ao de um neurônio humano no córtex visual, em resposta a um estímulo. Essas camadas são muito importantes devido ao desempenho do processamento dos pesos (filtros) aplicados, visto que para imagens de alta resolução seria necessário um número muito alto de neurônios, devido ao grande tamanho de entrada das imagens onde cada *pixel* é um recurso de entrada relevante. Por exemplo, para uma imagem razoavelmente pequena de tamanho 100x100 temos 10000 pesos para cada neurônio da próxima camada. Utilizando a convolução neste caso, podemos utilizar imagens lado a lado de tamanho 5x5, o que geraria 25 pesos apenas para serem passados para a próxima camada. Em resumo, podemos dizer que a camada de convolução reduz o número de parâmetros livres, permitindo que a rede seja mais profunda e tenha um maior desempenho no processamento dos filtros aplicados.

As CNN's podem incluir camadas de *pooling* locais e/ou globais. Estas camadas reduzem as dimensões dos dados combinando as saídas de *clusters* de neurônios em um único neurônio na próxima camada, ou seja, reduzem os mapas de características por meio de uma unificação de características similares. O *pooling* utiliza filtros pequenos, de tamanho 2x2.

Existem dois algoritmos de *pooling* que são comumente utilizados, o *Max* e o *Average*. O *Max Pooling* utiliza o valor máximo de cada *cluster* de neurônios no mapa de recursos, enquanto o *Average Pooling* utiliza do valor médio.

A camada totalmente conectada (*Fully Connected*) conecta cada um dos neurônios a todos os neurônios das camadas anteriores e posteriores, mas não possuindo conexões ente eles mesmos. Nas CNN's esta camada é evitada por causa das camadas de convoluções.

**Figura 3 -** Camadas de uma CNN




Fonte: Rafael L. Rocha, obtida em: https://www.researchgate.net/figure/Arquitetura-da-rede-neural-convolucional-com-seis-camadas-utilizada-nos-experimentos-A_fig2_337307179

## 2.3 YOLO

YOLO (*You Only Look Once*) é um algoritmo de detecção de imagens em tempo real, sua arquitetura foi planejada para obter velocidade e precisão sobre os objetos apurados.

O algoritmo foi desenvolvido por Redmon em 2015, com uma versão intitulada de YOLOv1 (REDMON, DIVVALA, GIRSHICK e FARHADI, 2016). Após a apresentação da tecnologia para a comunidade científica, novas versões começaram a surgir e sequencialmente nos outros anos foram sendo lançadas novas versões como a YOLOv2 (REDMON e FARHADI, 2016), novamente publicada por Redmon. Nesta versão foram realizadas melhorias nas classes do algoritmo com o objetivo de obter uma melhor velocidade de processamento.

Em 2018 foi lançada a versão YOLOv3 (REDMON e FARHADI, 2018), onde segundo Redmon, esta apresenta uma performance 1000 vezes mais rápida que outros sistemas, como o R-CNN (*Region Based Convolutional Neural Network*) e 100 vezes mais rápida que o *Faster* R-CNN. Para suas análises, YOLOv3 utiliza regressão logística em suas caixas delimitadoras, assim pode-se verificar se há sobreposição sobre a imagem anterior, caso não haja ela é ignorada e a próxima imagem é analisada.

Em 2020 foi lançada uma nova versão intitulada de YOLOv4 (BOCHKOVSKIY, WANG e LIAO, 2020), que veio com o intuito de aprimorar a Precisão Média (AP) e FPS em 10% e 12%.

Pouco tempo após o lançamento do YOLOv4 a comunidade continuou aprimorando o algoritmo e assim foi desenvolvido o YOLOv5 (ULTRALYTICS, 2020). Diferente das versões anteriores esta foi desenvolvida por um grupo de programadores independentes e o código fonte foi disponibilizado no GitHub.

Entre as principais diferenças YOLOv5 destacasse o ambiente que o algoritmo foi construído, passando de *C++* para *Python*. Com a mudança para *Python* foi possível integrar a biblioteca *Pytorch* podendo assim melhorar a performance e extração de métricas.

O treinamento na versão YOLOv5 pode ser executado em quatro tipos de redes diferentes, são elas: YOLOv5s; YOLOv5m*;* YOLOv5l; e YOLOv5x. Os sufixos (*s*, *m*, *l* e *x*) representam a quantidade de quadros por segundo que será realizado o treinamento, onde a ordem de grandeza começa em '*s'* e terminar em '*x'*, podendo assim aumentar a qualidade do treinamento conforme disponibilidade de processamento e *hardware*. Para o desenvolvimento deste projeto foi utilizada a versão YOLOv5s, pois foi a indicada pelos desenvolvedores do algoritmo.

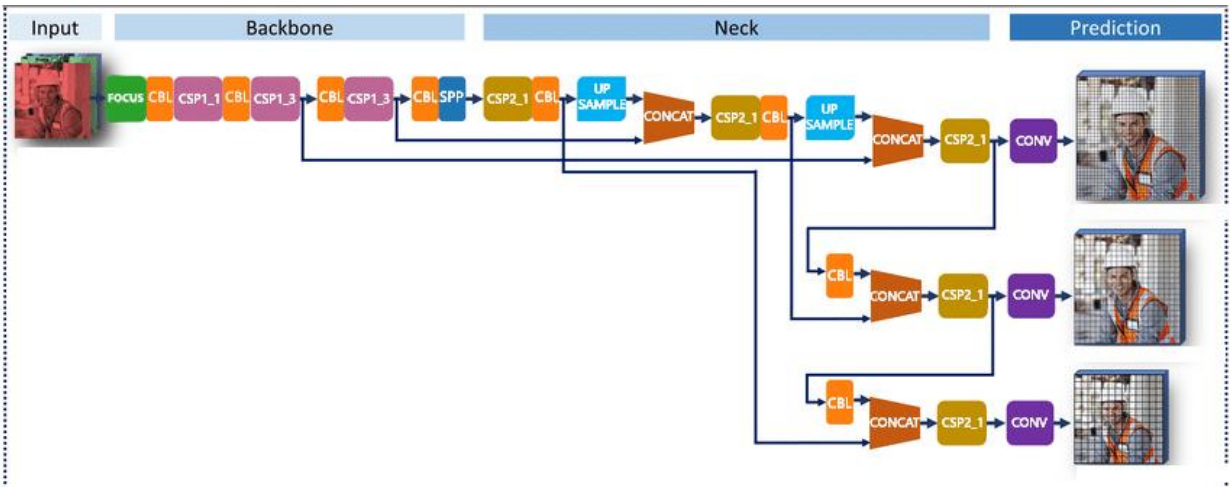
A arquitetura do YOLOv5 é baseada inteiramente em uma CNN, onde temos a camada de entrada (*Input*), as camadas ocultas (*Backbone* e *Neck*) e a camada de saída (*Prediction*) onde realiza a detecção dos objetos na imagem.

Na camada de entrada, recebemos as imagens escolhidas no padrão 640x640x3, onde o 3 significa que ela é separada em RGB.

Nas demais camadas ocultas, temos todo o processamento da imagem, onde são realizadas as operações matemáticas e são gerados os mapas característicos para extração de informações profundas da imagem. As camadas de convolução e *pooling* estão entre estas camadas ocultas.

Por último temos a camada de saída, que possui a função de realizar a identificação final dos objetos desejados e gerar os arquivos com os *labels*. Veja na Figura 4 a visão geral da arquitetura do YOLOv5.

**Figura 4 -** Arquitetura do YOLOv5




Fonte: Baseado em Zhao, 2021.

## 2.4 SISTEMAS EMBARCADOS

Sistemas Embarcados (SE), também referenciados por alguns autores como Sistemas Embutidos, são aqueles que buscam controlar outros equipamentos e dispositivos eletrônicos, podendo realizar uma ou mais tarefas específicas. Sua difusão se deu na década de 60 e hoje estão presentes em diversas áreas da sociedade, como a indústria automotiva, segurança, área médica e diversas outras áreas.

Os Sistemas Embarcados também podem ser utilizados para coletar dados oriundos de diferentes periféricos conectados ao mesmo. Os produtos que levam SE buscam ser confiáveis, seguros, performáticos e de fácil utilização.

Segundo Marwedel (2011), Sistemas Embarcados são caracterizados pelas seguintes definições:

> 1) Sistemas Embarcados devem ser realmente confiáveis. [...]
> 2) Devido a metas de eficiência, os projetos de *software* não podem ser feitos independentemente do *hardware* utilizado. [...]
> 3) Sistemas Embarcados devem atender a diversos requisitos não funcionais como restrições de tempo real, eficiência de energia e requisitos de confiabilidade. [...]
> 4) Sistemas embarcados reais são profundamente concorrentes. [...]
> 5) Linguagens de programação sequencial tradicionais não são a melhor maneira de descrever sistemas simultâneos e cronometrados.

As validações de desempenho de um SE são medidas através de seus requisitos de performance, consumo energético, confiabilidade, conectividade e segurança (MARWEDEL, 2011). A otimização de um sistema embarcado deve ser constante, sempre buscando melhorar a sua eficiência.

### 2.4.1 RASPBERRY PI

Raspberry Pi (RP) é um minicomputador desenvolvido pela Fundação Raspberry Pi (RPi) em 2006, seu objetivo principal era de promover a tecnologia em escolas e universidades. Hoje são encontrados diversos modelos com custos iniciais próximos a $50 e com um poder computacional considerável.

É possível executar diferentes Sistemas Operacionais (SO), os mais comuns são SO's baseados em Linux por sua fácil integração e flexibilidade com linguagens de baixo nível. Para o desenvolvimento deste projeto foi utilizado um RP modelo 4, com 1,5 GHz de CPU e arquitetura ARM v8, 8 GB de memória RAM, quatro portas USB e um controlador Ethernet.

# 3 DESENVOLVIMENTO

Nesta seção, será explicitado com detalhes o processo de desenvolvimento utilizado neste projeto, demonstrando a arquitetura, os *softwares* e os equipamentos que foram utilizados em todo o processo de desenvolvimento e obtenção dos resultados.

## 3.1 **ARQUITETURA DO PROJETO**

Para se ter uma Rede Neural Artificial rodando em um sistema, precisamos efetuar alguns procedimentos como a escolha do banco de imagens, a preparação das imagens, a execução do algoritmo do YOLOv5 e a configuração do sistema, para este possa suportar a execução da RNA.

Em um primeiro momento, foram realizados alguns testes com o ambiente do YOLOv5 para ser possível o entendimento do seu funcionamento e procedimento. Basicamente o algoritmo foi executado através da ferramenta do Google Colab em cima de um vídeo onde detectamos galinhas presentes em um ambiente. Visto seu funcionamento, foi possível então o início do projeto, tendo como primeiro passo a criação do ambiente, onde a RNA de detecção de EPI's seria configurada.

Em seguida, será abordado com detalhes os tópicos necessários para o desenvolvimento.

### 3.1.1 BANCO DE IMAGENS

Dentro de um projeto do ramo de inteligência artificial, principalmente quando falamos de *Machine Learning* e detecção de objetos, antes de qualquer coisa precisamos realizar uma busca por *datasets*, ou seja, banco de dados de imagens. O *dataset* será essencial na etapa do treinamento da RNA e quanto maior e mais diversificado ele for, nos trará um melhor resultado.

Dentro desse banco de imagens podemos ainda executar alguns algoritmos sobre ele, para assim termos distorções, simulando situações inusitadas como nevoeiro, chuva, horário noturno, dentro outros. Isso acaba aumentando ainda mais o número de imagens.

Visto isto, foi realizada uma pesquisa para identificar um *dataset* capaz de suprir a RNA a ser construída, que possuísse um número considerável de imagens. Foi selecionado então o CHVG (FERDOUS e AHSAN, 2022) que possui 8 classes, incluindo 4 de capacetes de cores diferentes (branco, azul, vermelho e amarelo), cabeça de pessoas, coletes, corpo de pessoas e

óculos de segurança. O seu nome CHVG deriva de CH para *Color Hardhat* (capacete colorido), V para V*est* (colete) e G para Glass (vidro).

Para esse conjunto, a maioria das imagens foram extraídas da internet. Além disso, grande parte contém o capacete, pois esse é o principal equipamento de segurança para os trabalhadores se protegerem de possíveis acidentes, e ainda temos a diversidade de cores para os capacetes, onde cada cor representa um papel diferente que o trabalhador desempenha no canteiro de obras. Ainda temos o colete, que ajuda as autoridades a observarem os trabalhadores que estão localizados a distância, e os óculos que protegem os olhos dos trabalhadores de poeira, ar, gás e outros possíveis ferimentos (FERDOUS e AHSAN, 2022).

Um ponto a ser observado é a divisão do *dataset* em relação aos conjuntos de dados para serem treinados, onde teremos 70% para treino, 20% para validação e 10% para o teste, onde cada imagem possui tamanho de 640x640.

### 3.1.2 PREPARANDO AS IMAGENS PARA O TREINAMENTO

Após o banco de imagens ser definido, foi realizada a importação do *dataset* escolhido para o *software* Roboflow (ROBOFLOW, 2022) o qual seria o responsável por preparar as imagens e gerar o arquivo a ser utilizado no algoritmo para o treinamento.

O Roboflow é um *software* que auxilia durante a preparação dos dados a serem analisados, através dele é possível demarcar os *labels* ao redor de objetos presentes nas imagens e assim classificar quais são os objetos a serem preditados. Esse *software* é gratuito, bastando apenas criar uma conta para gerenciar os projetos.

Neste *dataset* escolhido, além dos arquivos de imagens foi fornecido também os arquivos *.xml* contendo as caixas delimitadoras com os *labels* já demarcados. Nesses *labels* são abordadas 8 classes de detecção de objetos, veja na Tabela 2.

**Tabela 2 -** Descrição das classes

| Classe | Descrição |
|---|---|
| *person* | Detectado quando identifica uma pessoa. |
| *head* | Detectado quando identifica a cabeça de uma pessoa. |
| *glass* | Detectado quando identifica um óculos de proteção ou uma viseira. |

| | |
|---|---|
| *vest* | Detectado quando identifica um colete sinalizador. |
| *blue* | Detectado quando identifica um capacete azul. |
| *red* | Detectado quando identifica um capacete vermelho. |
| *white* | Detectado quando identifica um capacete branco. |
| *yellow* | Detectado quando identifica um capacete amarelo. |

Após a importação do *dataset*, o Roboflow fornece um *link* em diferentes formatos para que possam ser utilizados em diversas plataformas, para o projeto foi gerado no formato do YOLOv5.

### 3.1.3 EXECUTANDO ALGORITMO PARA CRIAÇÃO DA RNA

Para a execução da RNA podem ser utilizados diferentes ambientes referenciados pela desenvolvedora do YOLOv5, entre eles AWS, Docker, Kaggle e Colab.

Dentre as opções apresentadas optamos por utilizar o Colab (GOOGLE, 2022) por conta dos recursos fornecidos e de sua fácil usabilidade.

O Colab, também chamado de Google Colaboratory, é um projeto desenvolvido pela Google que busca incentivar o uso de *Machine Learning* de maneira fácil e intuitiva.

Os *Notebooks* (arquivos com código fonte) gerados no Colab são baseados na ferramenta de visualização Jupyter que facilita a integração de linguagens de programação, bibliotecas e gráficos.

Estes *Notebooks* podem ser compartilhados com múltiplos usuários e são executados em Máquinas Virtuais (VM) disponibilizados pela Google, estas máquinas oferecem execução de códigos em *Python*, podendo assim executar facilmente códigos de aprendizagem de máquina.

Após o projeto base ser criado no Colab, foi realizado o clone do repositório do algoritmo YOLOv5 e instalado as dependências necessárias para sua execução tais como Torch, Torchvision, OpenCV, Roboflow entre outras.

Tendo as dependências instaladas foi realizado a importação do *dataset* através do *link* fornecido pelo *software* Roboflow, com isso temos o ambiente preparado para iniciar o treinamento da RNA.

Após a importação do *dataset*, o próximo passo a ser realizado foi o treinamento da RNA, etapa com maior importância dentro do projeto, pois nela foram testados e obtidos os resultados de assertividade da RNA. Para isso foi executado o arquivo do algoritmo de treinamento (train.py), seguido de alguns hiperparâmetros, descritos na Tabela 3, com informações pertinentes ao treinamento.

**Tabela 3 -** Lista dos Hiperparâmetros

| Hiperparâmetros | Descrição |
|---|---|
| *img* | Tamanho das imagens que serão executadas no treinamento. |
| *epochs* | Número de épocas de treinamento. |
| *weights* | Formato do treinamento. |
| *batch* | Tamanho dos lotes de treinamento. |
| *data* | Localização do *dataset*. |
| *patience* | Quantidade de épocas que serão executadas para a normalização. |

O objetivo inicial era executar o treinamento com 3000 *epochs*. Entretanto durante os testes iniciais foi verificado que o treinamento se tornou estável a partir das 459 *epochs*, sem evoluções, por conta da estabilidade foi decidido realizar mais 141 *epochs* totalizando ao final 600 *epochs*,

Por conta da quantidade alta de *epochs* e de imagens do *dataset*, foram necessários maiores recursos computacionais e para isso foi realizada a contratação do serviço *premium* do Colab podendo assim garantir o treinamento desejado.

Após o treinamento ser finalizado, foi realizado a extração das métricas através da execução do arquivo do algoritmo de validação (val.py), nele foram acrescentados os hiperparâmetros *img*, *weights* e *data*, com as informações para a extração dos dados. Desta

forma foi possível extrair as métricas e gráficos detalhados sobre o processo de aprendizado da RNA.

O último passo foi a exportação do arquivo de treinamento no formato *.onnx*, a escolha específica por este formato se deu após uma recomendação da desenvolvedora do YOLOv5 (ULTRALYTICS, 2020), ao qual segundo *benchmarks* é o formato mais performático para ser executado em uma Raspberry pi.

### 3.1.4 EMBARCANDO A RNA NA RASPEBERRY PI

Seguindo o objetivo de embarcar a RNA em um dispositivo móvel e de fácil usabilidade, foi optado a utilização do minicomputador Raspberry Pi 4B+, com 8 GB de memória RAM, sistema operacional Raspbian X64 e armazenamento interno via SD CARD de 32 GB. Outro acréscimo foi o acoplamento de uma *webcam* USB a Raspberry Pi, com o objetivo de realizar detecções simultâneas.

Após termos os equipamentos conectados e o sistema operacional funcional, foi realizada a instalação de dependências e pacotes necessários para a execução do algoritmo YOLOv5.

Com o ambiente configurado, foi realizada então a importação do treinamento com o formato *.onnx* gerado no Colab para o repositório principal do projeto, localizado dentro da Raspberry. Em seguida foi executado o arquivo do algoritmo de detecção (detect.py) com os hiperparâmetros *img*, *weights* e *source* (dado que será inspecionado na detecção).

Para a execução da RNA indicando a leitura de *frames* capturados via *webcam* foi necessário que o hiperparâmetro *source* fosse iniciado com o valor 0.

Além de verificar *frames* diretos da *webcam* também é possível detectar objetos através de vídeos e imagens estáticas. Para isso é necessário indicar no hiperparâmetro o caminho para o arquivo.

**Figura 5 -** Sistema embarcado de detecção de EPI's

# 4 APLICAÇÕES E RESULTADOS

Nessa seção será abordado com maiores detalhes os conceitos de métricas utilizadas para a validação do projeto e sobre os resultados obtidos com o treinamento da RNA. Além disso, serão comparados os resultados com os do artigo selecionado como base para o projeto.

## 4.1 MÉTRICAS DE AVALIAÇÃO

Para a validação final do projeto e assim verificar se a RNA criada está atendendo ao seu propósito, identificando corretamente o uso de EPI's nas imagens, foram obtidas algumas métricas de validação. Essas métricas são colhidas ao final do treinamento da RNA, onde são vistas em cima do subgrupo do *dataset* responsável pela validação, que possui o equivalente a 340 imagens.

Essas métricas são geradas através de números que representam porcentagens, onde temos como valor mínimo 0 (0%) e valor máximo 1 (100%). Quanto mais próximo de 1 se encontrar o valor obtido, melhor será o desempenho da RNA quanto ao reconhecimento dos objetos.

Dentre as métricas que podemos obter através do algoritmo do YOLOv5, resolvemos selecionar a Precisão (P - *Precision*), a Revocação (R - *Recall*), a *F-Score*, a Precisão Média (AP - *Average Precision*) e a média das Precisões Médias (mAP - *Mean Average Precision*).

### 4.1.1 PRECISÃO, REVOCAÇÃO E *F-SCORE*

Para a obtenção de cada métrica, são utilizados alguns parâmetros: Verdadeiros Positivos (VP); Verdadeiros Negativos (VN); Falsos Positivos (FP); e Falsos Negativos (FN) (LAPIX, 2022).

Consideramos como VP todos os objetos de interesse que são classificados na classe correta. Ex: Identificou o capacete branco da imagem com a classe *white*.

Consideramos como VN quando não é detectado nenhum objeto na imagem e de fato ele não existe. Ex: Uma imagem mostrando apenas um canteiro de obras, sem nenhuma pessoa, não é marcado nenhum *label*.

Consideramos como FP os objetos que não são do interesse de avaliação, classificados na classe correta. Ex: Pessoa na imagem sem nenhum EPI sendo identificada com a classe *person*.

Por último, consideramos como FN os objetos que são detectados e marcados com a classe incorreta, ou objetos que aparecem na imagem e não são demarcados com os *labels*. Ex: Uma pessoa aparece na imagem e é marcada com a classe *white* ou não é marcada com nada.

Visto os parâmetros utilizados, podemos agora classificar os tipos de métricas e o que cada uma leva em consideração para seu cálculo (LAPIX, 2022).

A Precisão nos diz a probabilidade de um objeto encontrado na imagem estar marcado corretamente.

$$P = \frac{VP}{VP + FP}$$

A Revocação nos diz a probabilidade de um objeto encontrado estar dentre as classes de estudo, ou seja, é a proporção de positivos identificados corretamente.

$$R = \frac{VP}{VP + FN}$$

O *F-Score* nos mostra a média harmônica entre a Precisão e a Revocação, onde nos diz que quanto mais próximo de 1 seu resultado for, maior será a assertividade da RNA.

$$F - Score = \ 2 * \frac{P * R}{P + R}$$

### 4.1.2 *AVERAGE PRECISION* E *MEAN AVERAGE PRECISION*

A medida da Precisão Média (AP) é calculada utilizando os valores de P e R. Na prática montamos um gráfico Precisão X Revocação para conseguirmos então realizar a medida da área abaixo da curva. Este dado é muito importante e útil quando temos diversas classes no nosso *dataset*, pois com ele podemos visualizar melhor o resultado em um todo e comparar entre as classes para vermos se a RNA é eficaz. Para calcularmos essa medida, podemos utilizar a interpolação de 11 pontos iguais no eixo de Revocação (0, 0.1, 0.2, 0.3, ..., 1), junto dos valores máximos de Precisão para o ponto de interpolação da Revocação (LAPIX, 2022).

$$AP = \frac{1}{11} * \sum_{R} \max\left(P(R)\right)$$

Já o mAP é calculado realizando a média dos APs obtidos entre as classes.

$$AP = \frac{1}{N} * \sum_{i=1}^{N} AP_i$$

## 4.2 MATRIZ DE CONFUSÃO

A matriz de confusão, também conhecida como matriz de erro ou tabela de confusão, é uma tabela que nos permite a visualização do desempenho de uma RNA quanto a sua assertividade. Ela nos dá a visão correta de como nosso *dataset* se comportou nas classificações corretas dos objetos quanto as classes, gerando assim nossos parâmetros (VP, VN, FP, FN) (JUNIOR, 2021).

Cada linha da matriz representa os tipos de classes previstas, enquanto cada coluna representa os tipos de classes reais.

Pode-se dizer que é um tipo especial de tabela de contingência com duas dimensões (real e prevista), e conjuntos idênticos de classes em ambas as dimensões, onde cada combinação de dimensão e classe é uma variável na tabela de contingência.

Na Figura 6, podemos ver um exemplo de tabela de confusão, onde são extraídos os parâmetros e o que cada um significa.

**Figura 6 -** Exemplo de matriz de confusão

| | | Classificação Real | |
|---|---|---|---|
| | | **P** | **N** |
| **Classificação Prevista** | **P** | **VP** | FP |
| | **N** | FN | **VN** |

Fonte: Autor desconhecido, obtida em https://pt.wikipedia.org/wiki/Matriz_de_confus%C3%A3o

## 4.3 RESULTADOS OBTIDOS

Ao final do projeto, especificamente ao final do treinamento da RNA, foram obtidos os percentuais para as métricas selecionadas e descritas anteriormente. Nessa etapa também foram gerados os gráficos para cada uma das métricas e a matriz de confusão.

Como explicado no item 3.1.3, esse treinamento para a RNA foi realizado utilizando 600 *epochs*, mas o algoritmo compreendeu que o melhor resultado de treinamento foi obtido com 459 *epochs*, não havendo alterações de performance após este número.

Com isso, foi obtido para a RNA no geral uma taxa P de 88,8%, R de 82,4%, *F-Score* de 85,48% e mAP de 86,9%. Vale ressaltar também que para a Precisão a classe que obteve um maior percentual foi a *person* (92,2%) e o com o menor percentual foi a *glass* (81%), já para a Revocação a classe com maior percentual foi a *blue* (89,5%) e com o menor percentual foi a *glass* (67%). Podemos observar com isso que a identificação de EPI do tipo óculos de proteção, com a classe *glass*, possui um menor percentual de assertividade em todas as métricas testadas.

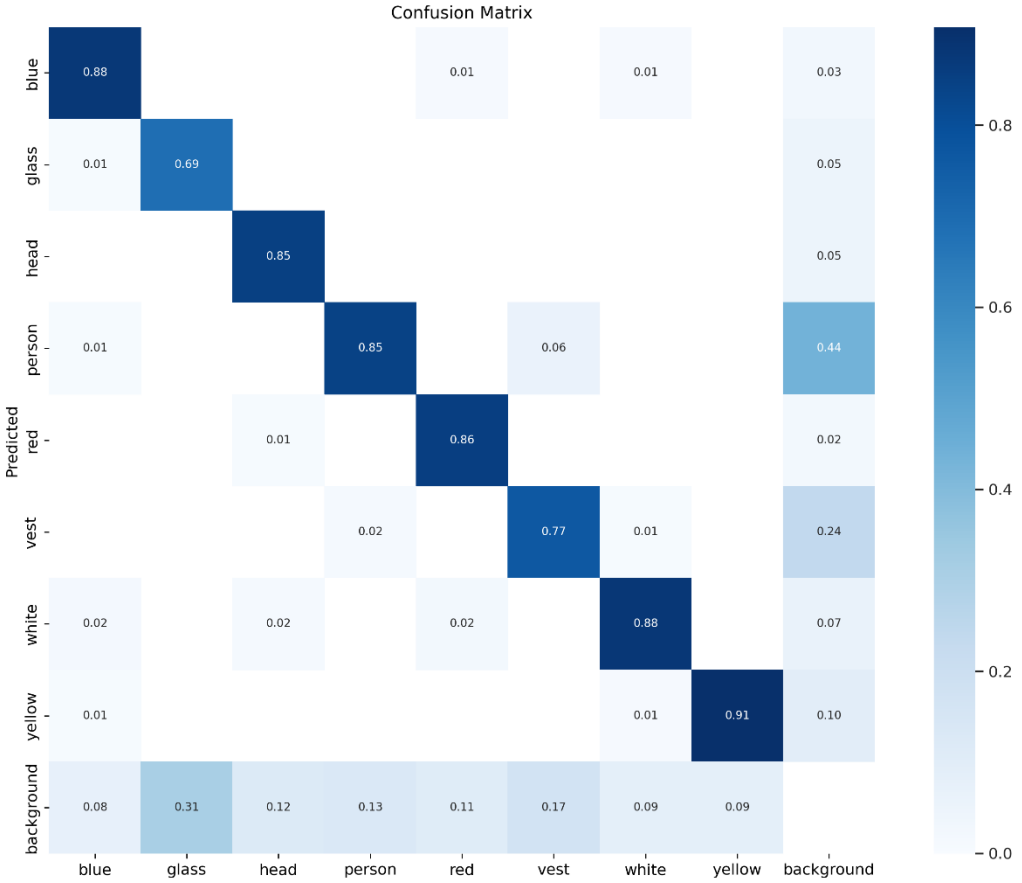
Esses resultados obtidos com o treinamento, foram os mesmos obtidos com o sistema embarcado. Veja na Tabela 4 a relação das métricas obtidas.

**Tabela 4 -** Percentuais obtidos

| **Classe** | **P (%)** | **R (%)** | ***F-Score* (%)** | **mAP (%)** |
|---|---|---|---|---|
| *all* | 88,8 | 82,4 | 85,48 | 86,9 |
| *blue* | 87,9 | 89,5 | 88,7 | 89,8 |
| *glass* | 81 | 67 | 73,34 | 69,1 |
| *head* | 89,3 | 84,7 | 86,94 | 88,6 |
| *person* | 92,2 | 81,7 | 86,63 | 89,4 |
| *red* | 88,4 | 81,9 | 85,02 | 89,5 |
| *vest* | 91,3 | 80,3 | 85,45 | 86,1 |
| *white* | 91,8 | 85,3 | 88,43 | 91,5 |
| *yellow* | 88,5 | 89 | 88,75 | 91,3 |

Observando a matriz de confusão na Figura 7, podemos ver o percentual dos parâmetros de VP, VN, FP e FN, onde temos que a maior taxa de VP é da classe *yellow* com 91% e a menor taxa de VP se dá para a classe *glass* com 69%.

**Figura 7 -** Matriz de confusão gerada




Os gráficos a seguir mostram, respectivamente, a Precisão (Figura 8), a Revocação (Figura 9), a *F-Score* (Figura 10) e o mAP (Figura 11) obtidos juntamente dos percentuais equivalentes de cada um.

Podemos observar nos gráficos de P e R, que eles são o oposto um do outro. Vemos também a média harmônica entre P e R na curva do gráfico F1. Por último, podemos ver no gráfico PxR (mAP) que as curvas se possuem o comportamento de ziguezague no eixo P, o que dificulta a leitura sendo preciso assim a realização do cálculo de área através da interpolação de 11 pontos.

**Figura 8 -** Gráfico de Precisão

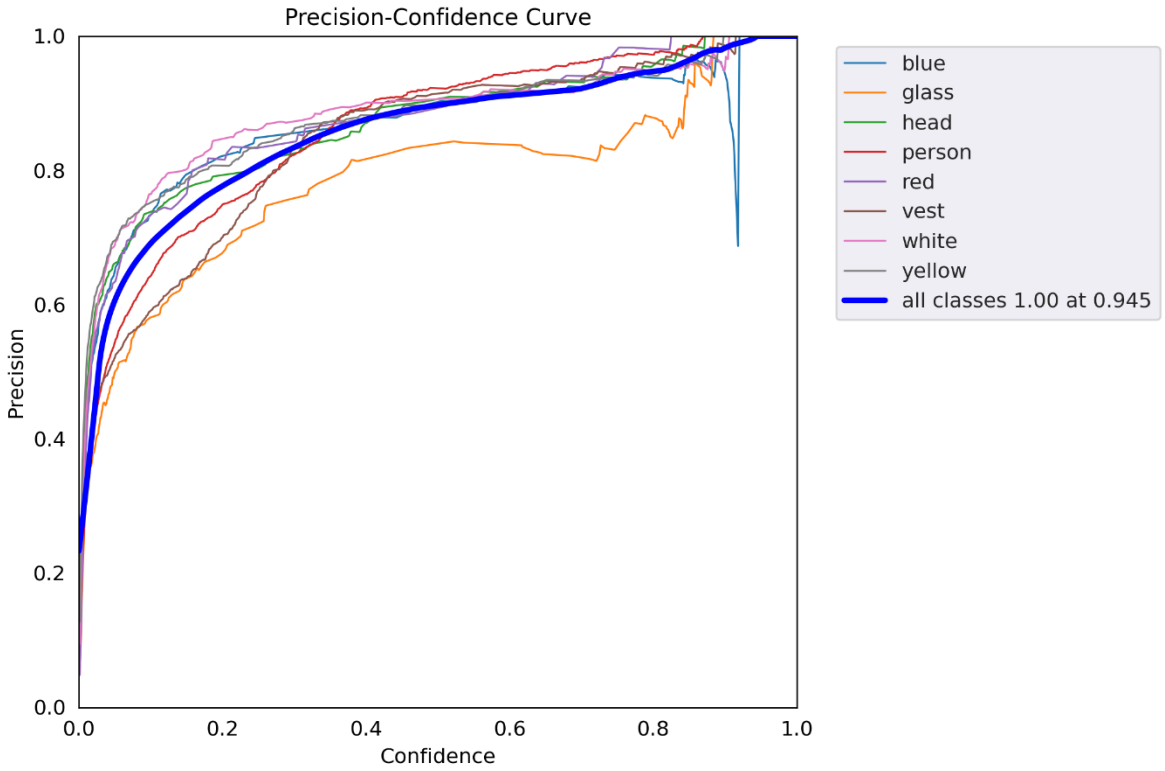



**Figura 9 -** Gráfico de Revocação

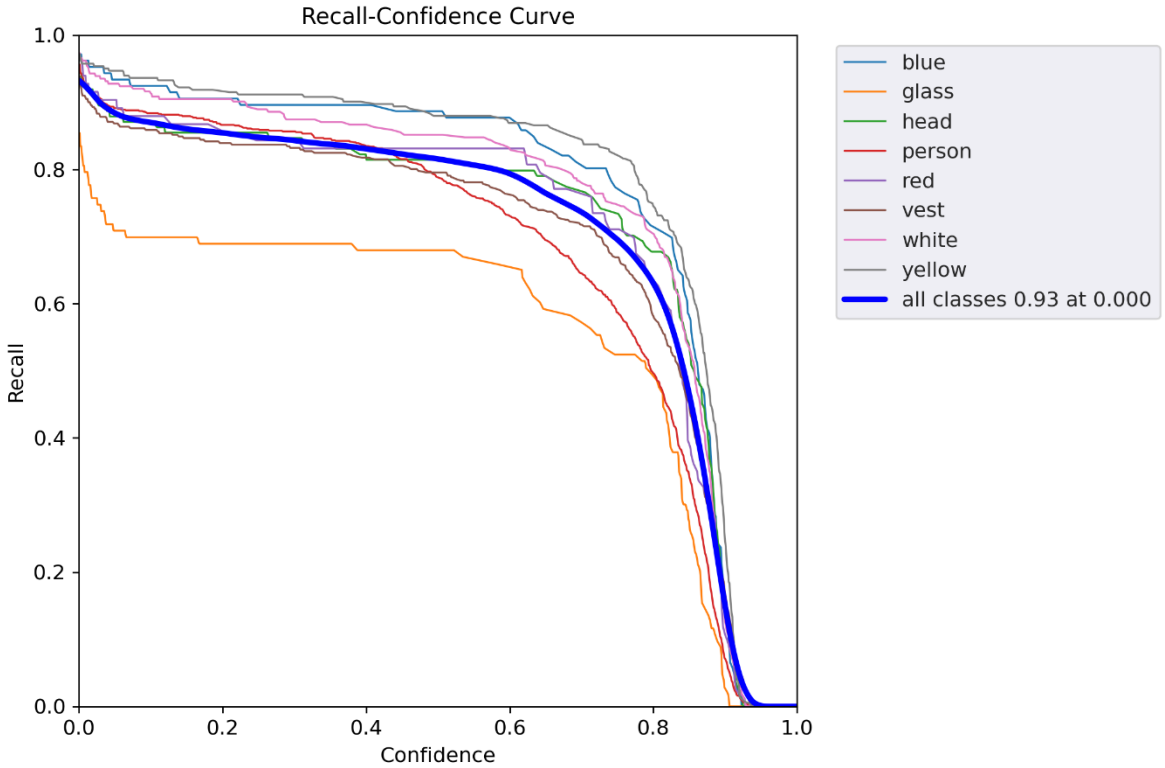


**Figura 10 -** Gráfico de *F-Score*

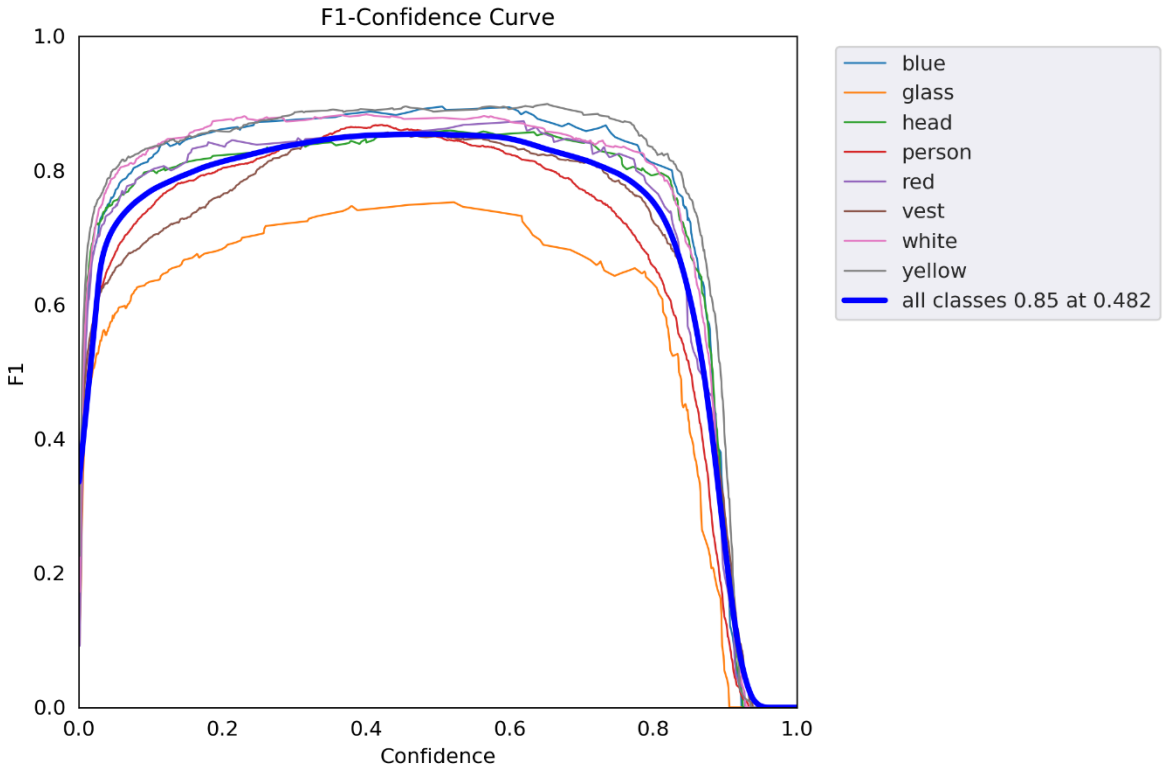
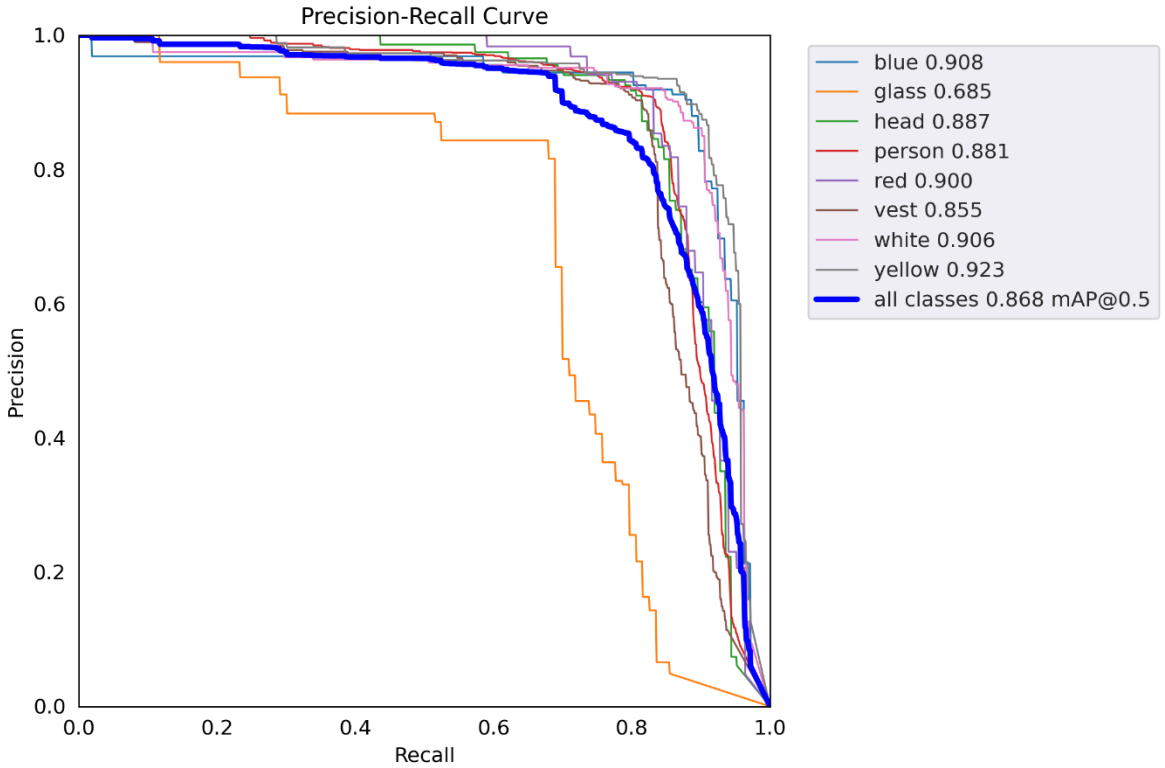


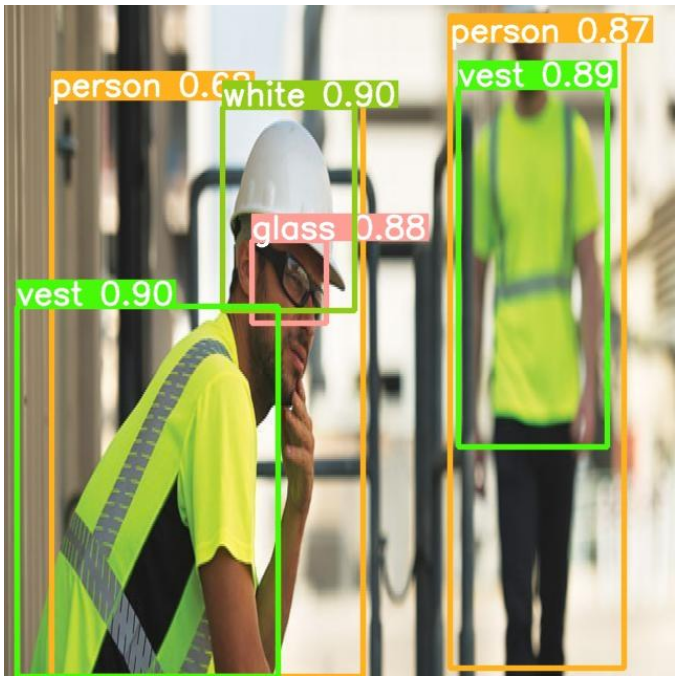

**Figura 11 -** Gráfico de mAP



Para constatação do funcionamento e detecção correta da RNA, o algoritmo foi executado sobre algumas imagens obtidas na internet de canteiros de obra além de testarmos utilizando uma *webcam* para realizar a detecção em tempo real. Veja abaixo, na Figura 12 e Figura 13, alguns exemplos do funcionamento da RNA.

**Figura 12 -** Exemplo de detecção


**Figura 13 -** Exemplo de detecção



## 4.4 COMPARAÇÃO COM OUTROS TRABALHOS

Para fins de autenticação e validação da RNA, foi selecionado um artigo (FERDOUS e AHSAN, 2022) como referência que descreve uma RNA com funcionamento muito semelhante ao da RNA construída no projeto, com o mesmo intuito de detecção de EPI's. Esse artigo também foi o responsável por disponibilizar o *dataset* com as imagens para o nosso treinamento.

Vale ressaltar que o desenvolvimento da RNA mencionada neste artigo, foi realizado utilizando a versão do algoritmo YOLOX e não a versão YOLOv5.

Para a obtenção dos resultados, Ferdous e Ahsan realizaram seu treinamento com 3 conjuntos de dados além de utilizarem todos os modelos da arquitetura do YOLOX, mas consideraram para sua verificação apenas a métrica de mAP. Visto isso, vamos comparar os resultados de mAP para o primeiro conjunto de dados utilizado no modelo de arquitetura '*s'*, obtidos por Ferdous e Ahsan, observe na Tabela 5.

**Tabela 5 -** Percentuais obtidos por Ferdous M e Ahsan SMM

| **Classe** | **mAP (%)** |
|---|---|
| *all* | 87,99 |
| *blue* | 90,01 |
| *glass* | 79,95 |
| *head* | 80,21 |
| *person* | 93,19 |
| *red* | 89,3 |
| *vest* | 93,55 |
| *white* | 91,12 |
| *yellow* | 86,25 |

Em comparação com os resultados de mAP obtidos pelo treinamento da RNA (Tabela 4), podemos ver que para algumas classes o algoritmo do projeto obteve um percentual mais elevado e para outras um percentual menor. Mas de maneira geral, os resultados se assemelham podendo-se constatar então que a RNA criada tem uma boa eficiência e os resultados obtidos são compatíveis a outros projetos realizados.

# 5 CONCLUSÃO

O presente projeto visou a integração da tecnologia de uma Rede Neural Artificial para auxiliar na fiscalização de Equipamentos de Proteção Individual entre os trabalhadores de um canteiro de obras.

Visto isto, foi proposto a utilização do algoritmo de treinamento e criação de RNA's YOLOv5, pois após um estudo referente foi constatado que no tempo atual seria o algoritmo mais eficiente e recomendado pela comunidade tecnológica.

Na etapa de treinamento, foram enfrentados alguns desafios quanto ao *hardware* para rodar essa tarefa, pois devido ao alto consumo de processamento seria inviável a utilização de nosso equipamento pessoal para executar um treinamento maior com mais *epochs*. Para isso, foi optado por utilizar a VM disponibilizada pelo Google Colab que utiliza de seu poder de GPU para processar a tarefa. Ainda nesta etapa, descobrimos que o próprio algoritmo nos mostra a quantidade máxima de *epochs* que devem ser consideradas para a execução do treinamento para este obter a máxima eficiência, onde chegamos ao número 459, pois após isto a RNA não teve alterações nos seus padrões de reconhecimento.

Além disso, foi encontrado um desafio na etapa em que a RNA foi embarcada em um minicomputador Raspeberry, pois ele possui poucos recursos computacionais para o processamento de imagens o que ocasiona em uma perda de *frames* ao interligarmos uma câmera *webcam* para capturar as imagens a serem processadas. Outro ponto relevante foram as dificuldades encontradas ao instalar os pacotes para a preparação do ambiente, pois a arquitetura ARM da RP ainda não possui total integração com o YOLOv5.

Contudo, podemos afirmar com base nos resultados obtidos ao final da simulação e validação da RNA, que foi alcançado o objetivo. Isso pode ser visto ao analisarmos e compararmos as métricas de P, R e mAP com a do artigo de referência.

Em um próximo momento, é previsto o aperfeiçoamento do *dataset* de treinamento, inserindo maiores variações de imagens para serem utilizadas no treinamento e assim melhorar a detecção das classes que tiveram um percentual de eficiência um pouco abaixo da média.

## REFERÊNCIAS

FERDOUS, Md; AHSAN S. M. M. **PPE Detector: a YOLO - Based Architecture To Detect Personal Protective Equipment (PPE) For Construction Sites**. Disponível em: https://doi.org/10.7717/peerj-cs.999. Acesso em: 8 nov. 2022.

PANTALEÃO, Sergio Ferreira. **EPI - Equipamento De Proteção Individual - Não Basta Fornecer É Preciso Fiscalizar**. Disponível em: http://www.guiatrabalhista.com.br/tematicas/epi.htm. Acesso em: 8 nov. 2022.

MINISTÉRIO DO TRABALHO E PREVIDÊNCIA. **Equipamentos de Proteção Individual – EPI**. Disponível em: https://www.gov.br/trabalho-e-previdencia/pt-br/composicao/orgaos-especificos/secretaria-de-trabalho/inspecao/seguranca-e-saude-no-trabalho/copy_of_equipamentos-de-protecao-individual-epi. Acesso em: 8 nov. 2022.

FRONTINO, Luciano de Medeiros. **Inteligência Artificial Aplicada: Uma Abordagem Introdutória**. 1ª. Editora Intersaberes, 2018.

DE BRITTOS, Valdati. **Inteligência Artificial – IA**. 1ª. Editora Contentus, 2020.

RUSSEL, S. J.; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro: Campus, 2004.

ALVES, Priscila M. **Inteligência Artificial e Redes Neurais**. Disponível em: https://www.ipea.gov.br/cts/pt/central-de-conteudo/artigos/artigos/106-inteligencia-artificial-e-redes-neurais. Acesso em: 8 nov. 2022.

SAS INSIGHTS. **Machine Learning: O Que É E Qual Sua Importância?** Disponível em: https://www.sas.com/pt_br/insights/analytics/machine-learning.html?gclid=Cj0KCQjwqc6aBhC4ARIsAN06NmMCbkZy29fKrCQ6fNgz5EGASStXt9r5BhetWXdSg38FzC3oIv0R2vkaAsu7EALw_wcB. Acesso em: 8 nov. 2022.

DE CARVALHO, André Ponce de Leon F. **Redes Neurais Artificiais**. Disponível em: https://sites.icmc.usp.br/andre/research/neural/#:~:text=Camada%20de%20Entrada%3A%20onde%20os,final%20%C3%A9%20conclu%C3%ADdo%20e%20apresentado. Acesso em: 8 nov. 2022.

ALVES, Gisely. **Entendendo Redes Convulacionais (CNNs)**. Disponível em: https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184. Acesso em: 8 nov. 2022.

WIKIPEDIA. **Convolutional Neural Network**. Disponível em: https://en.wikipedia.org/wiki/Convolutional_neural_network. Acesso em: 8 nov. 2022.

INSIGHT. **Aprenda a Criar e Treinar Uma Rede Neural Convolucional (CNN)**. Disponível em: https://insightlab.ufc.br/aprenda-a-criar-e-treinar-uma-rede-neural-convolucional-cnn. Acesso em: 8 nov. 2022.

ROCHA, Rafael L. **Image Inspection of Railcar Structural Components: An approach through Deep Learning and Discrete Fourier Transform - Scientific Figure on ResearchGate**. Disponível em: https://www.researchgate.net/figure/Arquitetura-da-rede-

neural-convolucional-com-seis-camadas-utilizada-nos-experimentos-A_fig2_337307179. Acesso em: 26 out. 2022.

ZHAO, Yifan. **Fast Personal Protective Equipment Detection for Real Construction Sites Using Deep Learning Approaches - Scientific Figure on ResearchGate**. Disponível em: https://www.researchgate.net/figure/Architectures-of-YOLO-v4-and-YOLO-v5s_fig4_351625277. Acesso em: 8 nov. 2022.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. **You Only Look Once: Unified, Real-Time Object Detection**. Disponível em: https://arxiv.org/abs/1506.02640. Acesso em: 8 nov. 2022.

REDMON, Joseph; FARHADI, Ali. **YOLO9000: Better, Faster, Stronger.** Disponível em: https://arxiv.org/abs/1612.08242v1. Acesso em: 8 nov. 2022.

REDMON, Joseph; FARHADI, Ali. **YOLOv3: An Incremental Improvement.** Disponível em: https://arxiv.org/abs/1804.02767v1. Acesso em: 8 nov. 2022.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. Disponível em: https://arxiv.org/abs/2004.10934v1. Acesso em: 8 nov. 2022.

ULTRALYTICS. **YOLOv5.** Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 8 nov. 2022.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; ET AL. **PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems** 32. Disponível em: https://dl.acm.org/doi/10.5555/3454287.3455008 Acesso em: 5 nov. 2022.

WANG, Z.; WU, Y.; YANG, L.; THIRUNAVUKARASU, A.; EVISON, C.; ZHAO, Y. **Fast Personal Protective Equipment Detection for Real Construction Sites Using Deep Learning Approaches.** Disponível em: https://www.researchgate.net/publication/351625277_Fast_Personal_Protective_Equipment_Detection_for_Real_Construction_Sites_Using_Deep_Learning_Approaches. Acesso em: 5 nov. 2022.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. **Rich feature hierarchies for accurate object detection and semantic segmentation**. Disponível em: https://ieeexplore.ieee.org/document/6909475 Acesso em: 5 nov. 2022.

ROBOFLOW. **Give Your Software The Sense Of Sight**. Disponível em: https://roboflow.com/. Acesso em: 10 de out. de 2022.

GOOGLE. **Colaboratory**. Disponível em: https://research.google.com/colaboratory/faq.html. Acesso em: 8 nov. 2022.

B. M. RANDLES, I. V. PASQUETTO, M. S. GOLSHAN, AND C. L. BORGMAN. **Using the Jupyter notebook as a tool for open science: An empirical study.** https://ieeexplore.ieee.org/document/7991618

LAPIX, Image Processing and Computer Graphics Lab. **Visão Computacional, Métricas, Mean Average Precision**. Disponível em: https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/visao-computacionalmetricasmean-average-precision/#:~:text=A%20precis%C3%A3o%20m%C3%A9dia%20(average%20precision%20%E2%80%93%20AP)%20%C3%A9%20a%20%C3%A1rea,com%20o%20aumento%20de%20TP. Acesso em: 8 nov. 2022.

WIKIPEDIA. **Matriz de Confusão**. Disponível em: https://pt.wikipedia.org/wiki/Matriz_de_confus%C3%A3o. Acesso em: 8 nov. 2022.

JUNIOR, Jose R F. **Matriz de Confusão**. Disponível em: https://pt.linkedin.com/pulse/matriz-de-confus%C3%A3o-jose-r-f-junior. Acesso em: 8 nov. 2022.

MICROSOFT, Learn. **Matrizes de Confusão**. Disponível em: https://learn.microsoft.com/pt-br/training/modules/machine-learning-confusion-matrix/2-confusion-matrices. Acesso em: 8 nov. 2022.

MARWEDEL, Peter. **Embedded System Design**, 2. ed. Dortmund: Springer Publishing, 2011.