

INSTITUTO FEDERAL DO ESPÍRITO SANTO  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE CONTROLE E  
AUTOMAÇÃO

**GUSTAVO GREGÓRIO FERNANDES SANTOS**

**DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO EM TEMPO REAL  
DE UTILIZAÇÃO DE EPI A PARTIR DE APRENDIZADO DE MÁQUINA**

SERRA  
2023

GUSTAVO GREGÓRIO FERNANDES SANTOS

**DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO EM TEMPO REAL  
DE UTILIZAÇÃO DE EPI A PARTIR DE APRENDIZADO DE MÁQUINA**

Dissertação apresentada ao Programa de Pós-graduação em Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, como requisito parcial para a obtenção do título de Mestre em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Flávio Garcia Pereira

Coorientador: Prof. Dr. Adilson Ribeiro Prado

SERRA

2023

## Dados Internacionais de Catalogação na Publicação (CIP)

---

S237d Santos, Gustavo Gregório Fernandes  
2023 Desenvolvimento de um sistema de monitoramento em tempo real de  
utilização de epi a partir de aprendizado de máquina / Gustavo Gregório  
Fernandes Santos. - 2023.  
68 f.; il.; 30 cm

Orientador: Prof. Dr. Flávio Garcia Pereira.  
Coorientador: Prof. Dr. Adilson Ribeiro Prado.

Dissertação (mestrado) - Instituto Federal do Espírito Santo, Programa  
de Pós-graduação em Engenharia de Controle e Automação, 2023 .

1. Automação. 2. Machine learning. 3. Aprendizado de máquina. 4.  
EPI. 5. YOLO. 6. Segurança no trabalho. I. Pereira, Flávio Garcia. II.  
Instituto Federal do Espírito Santo. III. Título.

CDD 629.895

---

Bibliotecário: Valmir Oliveira de Aguiar - CRB6/ES 566

**MINISTÉRIO DA EDUCAÇÃO**  
**INSTITUTO FEDERAL DO ESPÍRITO SANTO**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

**GUSTAVO GREGÓRIO FERNANDES SANTOS**

**“DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO EM TEMPO REAL DE  
UTILIZAÇÃO DE EPI A PARTIR DE APRENDIZADO DE MÁQUINA”**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo, como requisito parcial para obtenção de título de Mestre em Engenharia de Controle e Automação.

Aprovado em 12 de março de 2024

**COMISSÃO EXAMINADORA**

---

**Prof. Dr. Flávio Garcia Pereira**  
Professor Orientador

---

**Prof. Dr. Adilson Ribeiro Prado**  
Professor Coorientador

---

**Prof. Dr. Luiz Alberto Pinto**  
Examinador da Banca

---

**Prof. Dr. Vítor Faiçal Campana**  
Examinadora da Banca

## **RESUMO**

Equipamento de Proteção Individual (EPI) é o equipamento de uso individual utilizado pelo trabalhador, cujo principal propósito é protegê-lo contra riscos capazes de prejudicar sua saúde e segurança, além de reduzir os custos para o empregador com substituições de pessoal, demissões e processos de indenização. No entanto, muitas vezes, seja por negligência ou desconforto, há resistência ao seu uso e/ou à remoção do equipamento durante a realização das atividades. Diante desse problema, o departamento de Saúde, Segurança e Meio Ambiente (SMS) deve, portanto, inspecionar e monitorar o uso adequado do equipamento de proteção pessoal pelos trabalhadores na maioria das vezes. Como alternativa para auxiliar o departamento de segurança na verificação, demanda e quantificação do uso de equipamentos de proteção no local de trabalho, este projeto apresenta um modelo de aprendizado de máquina baseado na arquitetura *You-Only-Look-Once (YOLO)* para verificar a conformidade dos trabalhadores em relação ao seu comportamento de segurança em tempo real, utilizando imagens/vídeos de um sistema de segurança instalado em um local movimentado dentro de uma indústria. O algoritmo utiliza a abordagem de detecção de trabalhadores e EPIs básicos, como capacete, luvas e óculos, simultaneamente, por meio de aprendizado profundo previamente treinado por um conjunto de dados de imagens de trabalhadores em diferentes tipos de ambientes de trabalho e, em seguida, verifica se cada caixa delimitadora gerada está na posição correta, confirmando assim se o trabalhador está utilizando EPI ou não. Posteriormente, o plano inclui o desenvolvimento de um programa que utiliza o modelo de aprendizado de máquina pré-treinado da rede YOLO para prever se um trabalhador está utilizando EPI ou não, verificando a viabilidade de aplicar esse processo de detecção em vídeo em tempo real.

Palavras-chave: EPI. Segurança. Aprendizado de máquina. Aprendizado profundo. YOLO.

## **ABSTRACT**

Protective Equipment (PPE) is the equipment for individual use used by the worker where its main purpose is to protect against risks capable of jeopardizing their health and safety, in addition also reducing costs to the employer with personnel replacements, dismissals and indemnity processes. However, many times, either through negligence or discomfort, there is resistance to its use and/or the removal of the equipment during the performance of activities. In view of this problem, the HSE (Health, Safety and Environment) department must therefore inspect and monitor the proper use of workers' personal protective equipment mostly of the time. As an alternative to assist the security department in verifying, demand and quantifying the use of protective equipment in the workplace, this project presents a machine learning model based on the You-Only-Look-Once (YOLO) architecture to verify the workers' compliance regarding their safety behavior in real time, using images / video of a security system installed in a busy place within an industry. The algorithm uses the approach of detecting workers and basic PPE as helmet, gloves, goggles simultaneously by deep learning previously trained by a image dataset of workers in different types of labour ambient, and next verifies that each bounding box generated is in the correct position, thus confirming if the worker is carrying PPE or not. Later, the plan includes developing a program that utilizes the pre-trained machine learning model from YOLO to predict whether a worker is using PPE or not. Verifying the viability of applying this detection process in real-time video streams.

**Keywords:** PPE. Safety. Machine learning. Deep Learning. YOLO.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de classificação.	19
Figura 2 – Exemplo simples de localização de objetos.	20
Figura 3 – Exemplo de uma arquitetura de CNN comum.	21
Figura 4 – Exemplo de operação de convolução.	22
Figura 5 – Exemplo de operação de <i>pooling</i> .	23
Figura 6 – Modelo YOLO.	25
Figura 7 – Intersection Over Union (IOU)	26
Figura 8 – Arquitetura YOLOv4.	27
Figura 9 – Arquitetura YOLOv8.	29
Figura 10 – Desempenho da rede YOLOv4	30
Figura 11 – Desempenho da rede YOLOv8	30
Figura 12 – Fluxograma das etapas do trabalho	32
Figura 13 – Cálculos das coordenadas da caixa delimitadora para modelo de rede neural YOLO	33
Figura 14 – Anotação de trabalhadores portando ou não os EPIs pelo programa LabelImg	34
Figura 15 – Aquivos .txt gerado pelo programa, relacionando a anotação com a coordenada da localização do objeto.	34
Figura 16 – Quantificação das anotações divididas entre treino e validação na proporção 70/30.	36
Figura 17 – Configurações aplicadas no arquivo ".cfg".	37
Figura 18 – Configurações aplicadas no arquivo ".cfg".	38
Figura 19 – Edição do arquivo "obj.names"	38
Figura 20 – Edição do arquivo "obj.data"	39
Figura 21 – Recorte da visualização da interação do treinamento e suas respecti- vas métricas	41
Figura 22 – Habilitação de data augmentation no arquivo de configuração da rede.	43
Figura 23 – Filtro de "hue" aplicado a imagem de uma mão.	44
Figura 24 – Configurações aplicadas no arquivo "yolov8.yaml".	45
Figura 25 – Configurações aplicadas no arquivo "model.yaml".	46
Figura 26 – Diretório dos arquivos para treinamento e validação	46

Figura 27 – Recorte da visualização da interação do treinamento e suas respectivas métricas. . . . .	49
Figura 28 – Melhores modelos apresentados em gráfico de mAP@50% e Average Loss versus interações. . . . .	53
Figura 29 – Gráfico da evolução da rede discretizado por classes. . . . .	55
Figura 30 – Treinamentos realizados durante os testes com a rede YOLOv8. . .	56
Figura 31 – Matriz de confusão do melhor modelo da rede YOLOv8 . . . . .	58
Figura 32 – Resultado do melhor modelo em imagens avulças. . . . .	60
Figura 33 – Resultado do melhor modelo em imagens avulças. . . . .	61
Figura 34 – Resultado do melhor modelo em imagens avulças. . . . .	61
Figura 35 – Resultado da inferencia da rede rodando tempo real de 1 a 2 metros.	62
Figura 36 – Resultado da inferencia da rede rodando tempo real de 2 a 4 metros.	62
Figura 37 – Falsos negativos e não detecção durante predição com webcam em tempo real quando afastado acima de 4 metros. . . . .	63

## **LISTA DE TABELAS**

Tabela 1 – Quantidade de rótulos no <i>dataset</i> completo . . . . .	35
Tabela 2 – Valores utilizados nos treinamentos da YOLOv8 . . . . .	50
Tabela 3 – Precisão de cada classe nas interações correspondentes. . . . .	54
Tabela 4 – Tempo de processamento da rede YOLOv8 em cada teste. . . . .	55
Tabela 5 – Valores máximos de mAP obtidos em cada teste . . . . .	57
Tabela 6 – Precisão de cada classe nos melhores modelos de ambas redes . .	59

## **LISTA DE SIGLAS**

SMS	<i>Saúde, Meio ambiente e Segurança</i>
NR	<i>Norma Regulamentadora</i>
MTE	<i>Ministério do trabalho e Emprego</i>
CNN	<i>Convolutional Neural Networks</i>
YOLO	<i>You Only Look Once</i>
mAP	<i>Mean average precision (média de precisão média)</i>
mAP50	<i>(Média de precisão média para IoU de 0,5)</i>
EPI	<i>Equipamento de proteção Individual</i>
CNN	<i>Convolutional Neural Network</i>
R-CNN	<i>Region-based Convolutional Neural Network</i>
R-FCN	<i>Region-based Fully Convolutional Network</i>
IOU	<i>Intersection over Union</i>
COCO	<i>Common Objects in Context</i>
SSD	<i>Single Shot Multibox Detection</i>
RFID	<i>Radio Frequency Identification</i>
HUE	<i>Grau de luminosidade, escuridão ou intensidade de uma cor</i>

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>10</b>
1.1	OBJETIVOS	14
1.2	TRABALHOS RELACIONADOS	15
1.3	ESTRUTURA DO TRABALHO	18
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>19</b>
2.1	TAREFAS DE PROCESSAMENTO DE IMAGENS BASEADA EM VI- SÃO COMPUTACIONAL	19
2.1.1	<b>Classificação de imagem</b>	<b>19</b>
2.1.2	<b>Localização de objetos</b>	<b>20</b>
2.2	CONVOLUTIONAL NEURAL NETWORKS (CNN)	20
2.3	YOLO	24
2.3.1	<b>YOLOv4</b>	<b>26</b>
2.3.2	<b>YOLOv8</b>	<b>28</b>
2.3.3	<b>Comparação de desempenho da YOLOv4 e YOLOv8</b>	<b>29</b>
<b>3</b>	<b>MATERIAIS E MÉTODOS</b>	<b>31</b>
3.1	ETAPA 1 - AQUISIÇÃO DE DADOS	31
3.2	ETAPA 2 - ANOTAÇÃO DOS DADOS	32
3.3	ETAPA 3 - QUANTIFICAÇÃO DAS ANOTAÇÕES NO BANCO DE DADOS	34
3.4	ETAPA 4 - ADAPTAÇÃO DAS REDES YOLOv4 E YOLOv8 PARA A PROBLEMATICA DO PROJETO	35
3.4.1	<b>Configurações iniciais da rede YOLOv4</b>	<b>36</b>
3.4.2	<b>Treinando a rede YOLOv4</b>	<b>39</b>
3.4.3	<b>Configurações da rede YOLOv4 para obter modelos com maior acurácia</b>	<b>42</b>
3.4.4	<b>Configurações iniciais da rede YOLOv8</b>	<b>44</b>
3.4.5	<b>Treinando a rede YOLOv8</b>	<b>46</b>
3.4.6	<b>Configurações da rede YOLOv8 para obter modelos com maior acurácia</b>	<b>49</b>
3.5	IMPLEMENTAÇÃO DO ALGORITMO DE IDENTIFICAÇÃO DO EPI QUADRO A QUADRO EM UM VÍDEO	51

<b>4</b>	<b>RESULTADOS E DISCUSSÕES</b>	<b>53</b>
4.1	RESULTADOS DA YOLOV4	53
4.2	RESULTADOS DA YOLOV8	55
4.3	COMPARAÇÃO DOS RESULTADOS DAS REDES YOLOV4 E YOLOV8	58
4.4	RESULTADOS DO MELHOR MODELO APLICADO EM IMAGENS	60
4.5	RESULTADOS DO MELHOR MODELO APLICADO EM VIDEOS (WEBCAM E CAMERAS IP)	62
<b>5</b>	<b>CONSIDERAÇÕES FINAIS</b>	<b>64</b>
	<b>REFERÊNCIAS</b>	<b>66</b>

## 1 INTRODUÇÃO

Na grande maioria das atividades laborais, dá mais simples a mais complexa, sempre existirá o risco de algum tipo de acidente ocorrer, acidentes nos quais podem trazer danos a integridade do trabalhador, podendo provocar afastamento temporário, impossibilitar o trabalhador de exercer as suas atividades laborais normais, ou até mesmo vir a óbito (VIANA, 2014).

A fim de evitar tais injurias ao trabalhador e quando não existir medidas preventivas capazes de eliminar os riscos do ambiente em que se desenvolve a atividade, de outra forma, quando as medidas de proteção coletiva não forem viáveis, eficientes e suficientes para a atenuação de acidentes do trabalho e/ou de doenças profissionais e do trabalho o uso dos equipamentos de proteção individual (EPI) se faz necessário. Os EPIs são a primeira barreira individual para amortização de alguma lesão caso ocorra algum acidente ou exposição a algum risco (TOSMANN, 2019).

Nas diretrizes fundamentais do trabalho, o trabalhador tem o direito de gozar de uma qualidade de vida saudável e prazerosa (TRENTIN, 2016). Com o objetivo de reduzir os riscos e acidentes de trabalho, surgiu então a Segurança no Trabalho, uma ciência voltada para criar métodos preventivos e medidas técnicas afim de erradicar, ou pelo menos mitigar a ocorrência de acidentes de trabalho (TRENTIN, 2016).

No Brasil, foram criadas as Normas Regulamentadoras de segurança do trabalho, as NRs, totalizando 37 atualmente. Elas são normas cujo conteúdo abrange direitos, deveres e obrigações a serem cumpridos por empregadores e trabalhadores, sendo de uso obrigatório tanto para empresas privadas, quanto para as empresas públicas que estão sob o regime da CLT. As normas trazem as imposições e exigências mínimas de segurança para o trabalho, abordadas em diferentes tópicos de acordo com cada NR (TRENTIN, 2016).

Na Portaria nº 3.214, de 8 de junho de 1978, que aprova as Normas Regulamentadoras, do Capítulo V, Título II, da Consolidação das Leis do Trabalho, relativas à Segurança e Medicina do Trabalho, destaca-se a NR-6 Equipamentos de Proteção Individual EPI. Tal norma estabelece diversas regulamentações evidenciando responsabilidades do

empregador, empregado e fabricantes quanto ao seu fornecimento, uso e certificação dos EPIs (BRASIL, 2018).

A norma determina que é obrigação do empregador fornecer o EPI de forma gratuita ao trabalhador para o desempenho de suas funções dentro da empresa, bem como o empregado deve utilizar apenas para a finalidade a que se destina e cumprir as determinações do empregador sobre o uso adequado (BRASIL, 2018).

A Norma Regulamentadora 06 define um Equipamento de Proteção Individual - EPI todo dispositivo ou produto, de uso individual utilizado pelo trabalhador, que se destina à proteção de riscos que têm o potencial de ameaçar a saúde do trabalhador no ambiente de trabalho (BRASIL, 2018). De acordo com a NR 06, no item 6.6.1, cabe ao empregador quanto ao EPI:

- (a) adquirir o adequado ao risco de cada atividade;
- (b) exigir seu uso;
- (c) fornecer ao trabalhador somente o aprovado pelo órgão nacional competente em matéria de segurança e saúde no trabalho;
- (d) orientar e treinar o trabalhador sobre o uso adequado, guarda e conservação;
- (e) substituir imediatamente, quando danificado ou extraviado;
- (f) responsabilizar-se pela higienização e manutenção periódica; e,
- (g) comunicar ao Ministério do Trabalho e Emprego (MTE) qualquer irregularidade observada.

Ainda de acordo com a norma, no item 6.6.7, cabe ao empregado quanto ao EPI:

- (a) usar, utilizando-o apenas para a finalidade a que se destina;
- (b) responsabilizar-se pela guarda e conservação;
- (c) comunicar ao empregador qualquer alteração que o torne impróprio para uso; e,

(d) cumprir as determinações do empregador sobre o uso adequado.

Dentre as responsabilidades mencionadas, fica evidente que no subitem (a) do item 6.6.7, o empregado tem a obrigação de utilizar o EPI para a sua atividade laboral na qual o foi conduzido. Porém segundo Nascimento *et al.* (2015), por motivos como esquecimento, incômodo, negligência ou falta de instrução adequada, geram resistência ao seu uso durante a execução do trabalho, cabendo ao empregador seguir o subitem (b) do item 6.6.1, e para isso o mesmo deve contar com uma equipe instruída de pessoas capacitadas em segurança para aquisição, distribuição, acondicionamento e instrução e conscientização de uso dos EPIs.

As empresas devem seguir à risca as normas regulamentadoras para preservar a segurança e saúde dos trabalhadores; caso ocorra descumprimento elas estão sujeitas a sanções do Governo Federal (OCUPACIONAL, 2018).

O Ministério do Trabalho é o órgão mais atuante quanto à fiscalização. O mesmo é responsável por verificar se as regras estão sendo respeitadas; caso não estejam, as sanções variam desde ações reclamatórias, ações civis públicas até o pagamento de multas. As principais sanções são (OCUPACIONAL, 2018):

- Multas aplicadas pelo Ministério do Trabalho;
- Embargo da obra ou interdição do estabelecimento, máquinas ou equipamentos;
- Pagamento de adicionais de insalubridade e periculosidade;
- Risco ou perigo de morte ou à saúde do trabalhador, é caracterizado como Crime de Perigo (artigo 132 do Código Penal);
- Dano físico ao trabalhador, o caso é caracterizado como Lesão Corporal (artigo 129, §6º, do Código Penal).
- Morte do trabalhador decorrente do descumprimento das normas de segurança, o caso é tratado como um homicídio (artigo 121 do Código Penal).

Apesar de existir tais normas evidenciando a grande importância do uso do EPI, tanto para assegurar a integridade do empregado quanto ao empregador, por muitas vezes existem resistência ao seu uso, ou a retirada do equipamento durante a execução da atividades, o uso incorreto, ou não uso, por diversos motivos pontuados como, esquecimento, incômodo, negligência, falta de informação correta sobre o uso, limitação de mobilidade e até falta de sensibilidade (FIORAVANTE; BARONI, 2012).

Para garantir o uso dos EPIs se faz necessário uma equipe de segurança bem instruída, porém para satisfazer os requisitos modernos de gestão de segurança, contando apenas com a supervisão manual tradicional se torna um trabalho árduo. Neste contexto, é importante estudar sobre a detecção automática e reconhecimento de objetos no ambiente laboral (LI *et al.*, 2020), atrelado a isso atualmente os métodos de detecção de objetos via visão computacional e inteligência artificial vem aumentando de forma expressiva, principalmente por meio das Redes Neurais Convolucionais. Tal rede tem a função de extrair padrões e características de um conjunto de imagens, por meio de um treinamento prévio e ela consegue classificar essas características à um objeto específico, e posteriormente realizar a detecção desses mesmos objetos em imagens posteriormente expostas a está rede (GONZAGA; ALMEIDA, 2020).

Tendo em vista a problemática do não uso de EPIs vinculado a dificuldade de fiscalização, o presente trabalho tem por objetivo o desenvolvimento de um sistema de identificação automático de EPIs básicos a partir de estratégias de aprendizado de máquina, utilizando redes neurais convolucionais (CNN). Tal sistema auxiliará a equipe de segurança do trabalho a verificar, advertir e quantificar o uso dos equipamentos de proteção no meio laboral. Os dados gerados serão utilizados para auxiliar a empresa nas tomadas de decisões e aumentar a performance de segurança.

Os EPIs definidos para serem identificados pela rede neural convolucional foram:

**O capacete de segurança:** tem a funcionalidade de proteger o crânio do colaborador contra impactos durante o processo de trabalho, seja este proveniente de queda de objetos até a batida contra estruturas no nível da cabeça. O capacete é composto basicamente por dois elementos sendo constituído pela carcaça e pelo arnês. A carcaça ou cocha, é a parte externa com formato oval e concavo feito de material

plástico termoendurecível possui características como boa resistência a altas e baixas temperaturas e a produto químico. O arnês é composto pela alça ajustável, carneira e a alça de ajuste traseira, tais elementos são responsáveis pela absorção da energia proveniente do impacto e mantem a cabeça do colaborador na posição e afastada da carcaça com distância entre 6 a 19 milímetros.(GOUVÊA, 2018).

**O Óculos de proteção:** tem a função de preservar o globo ocular de injurias provocadas por particulados em geral, respingos, vapores químicos, metais em fusão, alta ou baixa luminosidade e radiações ultravioleta (UVA e UVB). Ele é composto por uma armação que contorna toda a face e é sustentada pelas hastes que se encaixam nas orelhas, que se apoiam sobre o nariz e sustentam as lentes e a armação como um todo. As lentes são feitas em policarbonato incolor ou em diferentes tonalidade, permitindo uma visão periférica e sem distorcer a imagem, resistente a impacto de partículas, respingos e borrifos de produtos químicos (ALMEIDA, 2019).

**A luva de proteção:** é um equipamento de proteção individual destinado a proteger as mãos contra os riscos relacionados com ações mecânicas, contato com corrente elétrica, superfícies quentes, superfícies frias, agentes biológicos, agentes químicos, radiação, ou qualquer combinação dessas (LIMA, 2019).

## 1.1 OBJETIVOS

O objetivo geral deste trabalho é utilizar o aprendizado de máquina baseada na Rede Neural Convolucional de Disparo Único (*Single Shot Detection*) chamada *YOLO* (*You only look once*) para a detecção dos EPIs - Capacete, Luva e Óculos de proteção, além de desenvolver um programa para quantificar as detecções com a finalidade de gerar indicadores de conformidade de segurança de cada detecção. Para isso, os seguintes objetivos específicos foram traçados:

- Aquisitar em banco de dados imagens já existentes;
- Realizar a anotação dos EIPs nas imagens;
- Segregação proporcional dos dados anotados em validação e teste;

- Configurar da rede neural YOLOv4 e YOLOv8;
- Modificar de hiper parâmetros da rede buscando melhores métricas;
- Definir o melhor modelo;
- Desenvolver programa para testar a rede em tempo real;

## 1.2 TRABALHOS RELACIONADOS

Na literatura a utilização de CNN é amplamente utilizada para a detecção de EPIs. Por exemplo, Li *et al.* (2020) preocupados com a segurança dos colaboradores envolvidos em construções civis eles desenvolveram um método de detecção de capacetes de segurança utilizando redes convolucionais, com o objetivo de contribuir com o monitoramento do uso do capacete no canteiro de obras, os autores consideram a verificação manual como demorada, passível de erro e não suficiente para satisfazer padrões modernos de construção. A metodologia empregada foi utilizar redes neurais convolucionais YOLO e SSD para realizar a detecção do EPI. O projeto atingiu uma precisão média de 95% mAP@50.

Chen e Demachi (2020), desenvolveram um projeto que tem por metodologia principal utilizar a rede YOLOv3 para identificar capacete e máscara e utilizar um programa de extração de keypoints corporais (OpenPose) para determinar se o capacete e a máscara estão nos lugares corretos, permitindo assim a entrada nas instalações nucleares em descomissionamento em Fukushima no Japão. O trabalho atingiu uma precisão média de 97.64%.

Natha, Behzadanb e Paala (2020), propõe como o objetivo fazer a detecção automática da vestimenta de trabalho e de capacete utilizando a metodologia de aprendizado profundo utilizando a rede YOLOv3. O artigo faz um contra ponto entre os métodos já existentes para análise de utilização de EPIs, no qual comenta que são categorizadas em dois tipos, baseadas em sensores e baseadas em visão. Um exemplo utilizando sensores, inclui o uso de etiquetas de Identificação por Radiofrequência (RFID) instaladas em cada componente do EPI e verificação das etiquetas com um scanner na entrada de um local de trabalho para controlar se os trabalhadores estão usando

o EPI adequado. Outro exemplo inclui o uso de uma rede de área local (LAN) para verificar os RFIDs instalados nos componentes do EPI, que monitoram continuamente a conformidade com o EPI enquanto os funcionários estão trabalhando. No entanto é afirmado que a abordagem baseada em sensores requer um investimento significativo na compra, instalação e manutenção de redes de sensores, o que pode levar a preferência de utilização de métodos de detecção por visão computacional com a utilização de aprendizado profundo. Foi atingido no processo a precisão média de 72.3% na detecção de vestimenta, capacete e trabalhador.

Em uma abordagem diferente, porém totalmente ligada com segurança Fang *et al.* (2019), desenvolveram um método que não faz o reconhecimento de EPIs, mas sim a identificação do posicionamento seguro do trabalhador no canteiro de obras. Os autores elaboraram, utilizando aprendizado profundo, mais precisamente uma rede neural chamada de Mask Region Based Convolutional Neural Network (R-CNN), um método que verifica se o trabalhador está muito próximo da borda de alguma estrutura, sendo ela com guarda corpo ou não, identificando assim a potencialidade da ocorrência de algum acidente de queda em altura, foi atingindo uma precisão de 75% na identificação do comportamento seguro do colaborador perante a um ambiente de risco.

Os autores Cabrejos e Roman-Gonzalez (2023), em seu projeto propuseram a detecção de vários Equipamentos de Proteção Individual, incluindo capacete, colete, luvas, óculos de proteção, botas e até mesmo a identificação da própria pessoa. Durante o desenvolvimento do projeto, foi realizado um estudo cuidadoso para determinar qual versão da YOLOv5 seria mais adequada para sua problemática. Como resultado, concluíram que a versão YOLOv5s atendia às suas necessidades, uma vez que fornecia uma combinação de boa precisão e velocidade satisfatória. O projeto alcançou uma precisão de 98,13% a uma taxa de 13 quadros por segundo na detecção dos EPIs mencionados.

A pesquisa realizada por Lo, Lin e Hung (2023) teve como objetivo aprimorar a segurança no local de trabalho empregando algoritmos de aprendizado profundo baseados na YOLOv3, YOLOv4 e YOLOv7 para a detecção em tempo real da conformidade com o uso de Equipamentos de Proteção Individual (EPIs). Os EPIs escolhidos para foram

capacete e colete, com o seu uso ou não uso, totalizando quatro classes. No projeto foram utilizadas um grande conjunto de 11.000 fotos para treinamento e validação. Foi utilizado técnicas de pré-processamento de dados, incluindo aumento de dados. Os autores mencionam em seu artigo que o modelo obtido foi utilizando a rede YOLOv7 superando os demais em termos de precisão e velocidade de detecção, com valores de mAP de 97,29% e 25,02 FPS.

A pesquisa coordenada por Ahmed *et al.* (2023) abordou a detecção de Equipamentos de Proteção Individual (EPIs), com foco no capacete, e a identificação de quatro modelos diferentes de capacetes, cada um distinto por sua cor (branco, azul, vermelho e amarelo). Além disso, o estudo visou também a identificação de pessoas, coletes e óculos de proteção. O projeto utilizou um conjunto de dados composto por 1.189 imagens, divididas em 430 imagens de treinamento, 172 de validação e 115 para testes.

Durante o desenvolvimento do projeto os autores realizaram a comparação entre os algoritmos Faster RCNN e YOLOv5 para a detecção dos EPIs propostos. Segundo eles, no contexto específico abordado, o modelo treinado pela rede Faster RCNN apresentou um desempenho superior ao da rede YOLOv5, atingindo um mAP50 de 86%.

De acordo com os autores Lee *et al.* (2023), os trabalhos na indústria da construção civil são intrinsecamente perigosos e resultam em um maior número de acidentes e fatalidades em comparação com outros setores industriais. Com o intuito de zelar pela segurança e integridade dos trabalhadores, o estudo propôs a verificação das condições seguras dos trabalhadores por meio da tarefa de visão computacional de segmentação utilizando a rede YOLACT, uma CNN própria para segmentação de objetos.

A segmentação visa destacar os colaboradores que não estão em conformidade com as normas de segurança, marcando-os com a classe *unsafe* (inseguro), enquanto os trabalhadores que seguem as normas de segurança utilizando vestimenta completa e capacete são segmentados com a classe *safe* (seguro). O conjunto de dados utilizado foi coletado a partir de diversas fontes, incluindo o Google Images, câmeras de vigilância e smartphones utilizados nos canteiros de obras, com um total de 1031 amostras destinadas ao treinamento e 257 amostras para fins de validação.

Os resultados obtidos pelo modelo YOLACT demonstraram um desempenho satisfatório, com um mAP50 (média de precisão média para IoU de 0,5) de 66,4 na determinação do status de segurança dos trabalhadores. Para futuros trabalhos, os autores têm a intenção de aprimorar a capacidade do modelo a fim de identificar possíveis situações de risco, estabelecendo correlações entre o ambiente de construção e a segurança dos trabalhadores.

### 1.3 ESTRUTURA DO TRABALHO

Este trabalho se divide em cinco capítulos, o primeiro contém o resumo e a introdução com o contexto do problema proposto. O segundo se trata das referências bibliográficas em que o projeto se baseou. O terceiro capítulo descreve a metodologia empregada para a confecção da pesquisa. No capítulo quatro são apresentados os resultados do projeto e, por fim, no último capítulo se definem as conclusões e propostas futuras.

No capítulo 2 é discorrido sobre os métodos que serão utilizados nesse trabalho. Inicialmente é dada uma visão geral sobre os algoritmos de classificação CNN. Em seguida é falado sobre os métodos de extração de características e demarcação de objetos em imagens utilizando a rede convolucional profunda YOLO.

O capítulo 3 apresenta as etapas da metodologia deste trabalho. É apresentado como os dados foram aquisitados e como foi o processos de categorização dos dados. Apresenta também o processamento necessário para classificar as imagens identificando cada tipo de EPI e a conformidade com a segurança. Por fim são apresentados parâmetros e arquiteturas dos modelos de classificação utilizado pela YOLOv4 e YOLOv8 e também é descrito o processo de validação, quais métricas são utilizadas e como os algoritmos foram treinados.

Já no capítulo 4 são mostrados os resultados do treinamento feito com a metodologia de validação proposta. Nessa parte somente são mostradas as estratégias que deram melhor resultado.

Finalmente, no capítulo 5 são feitas as considerações finais do trabalho juntamente com as propostas de melhorias futuras.

## 2 REFERENCIAL TEÓRICO

Neste capítulo, serão expostas as técnicas e instrumentos empregados durante a etapa experimental do projeto. Serão discutidos tanto os aspectos teóricos quanto o algoritmos de aprendizado de máquina e extratores de características utilizados.

### 2.1 TAREFAS DE PROCESSAMENTO DE IMAGENS BASEADA EM VISÃO COMPUTACIONAL

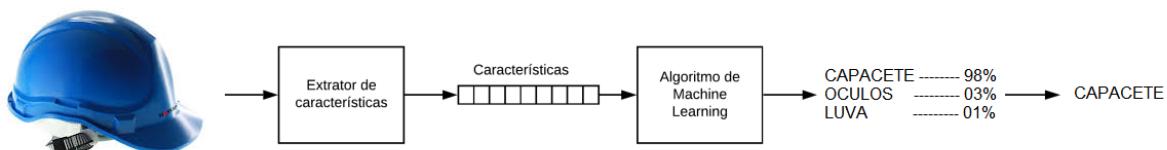
As tarefas de interpretação de imagens são um conjunto de técnicas e algoritmos da área de visão computacional que permitem a extração de informações e significados a partir de imagens digitais. Essas tarefas podem incluir a classificação de imagens, detecção de bordas, identificação de padrões, reconhecimento e localização de objetos, segmentação de imagens, entre outras (SINHA; PANDEY; PATTNAIK, 2018).

Neste trabalho são usadas apenas as tarefas de classificação e localização de objetos.

#### 2.1.1 Classificação de imagem

Nesta tarefa o objetivo é classificar uma imagem de acordo com o rótulo a que ela realmente pertence. A classificação de imagem é uma tarefa da visão computacional que consiste em atribuir um ou mais rótulos em uma imagem digital, com base em suas características visuais. Essas categorias podem ser pré-definidas ou aprendidas a partir dos próprios dados com as redes neurais. Essas redes são treinadas a partir de centenas ou milhares de imagens e seus respectivos rótulos, e são capazes de identificar padrões visuais complexos e sutis nas imagens. A figura 1 exemplifica esta tarefa.

Figura 1 – Exemplo de classificação.

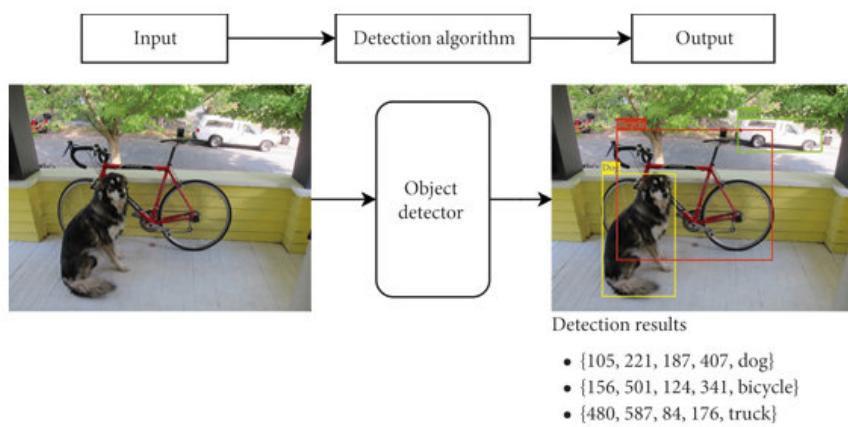


Fonte: Elaborado pelo próprio Autor (2023)

### 2.1.2 Localização de objetos

Nesta tarefa o objetivo é segmentar o objeto desejado na imagem. Para isso é necessário ter imagens com os objetos e máscaras dessas imagens onde só aparecem os objetos desejados para treinamento do algoritmo de classificação. Assim como na seção 2.1.1 é usado um extrator de características para transformar a imagem em um vetor de características. Por fim, o algoritmo de *machine learning* classifica cada pixel da imagem com a classe a que ele representa, gerando assim uma máscara da imagem original. A figura 2 exemplifica esta tarefa.

Figura 2 – Exemplo simples de localização de objetos.



Fonte: Hassan Yasser Khalil (2020)

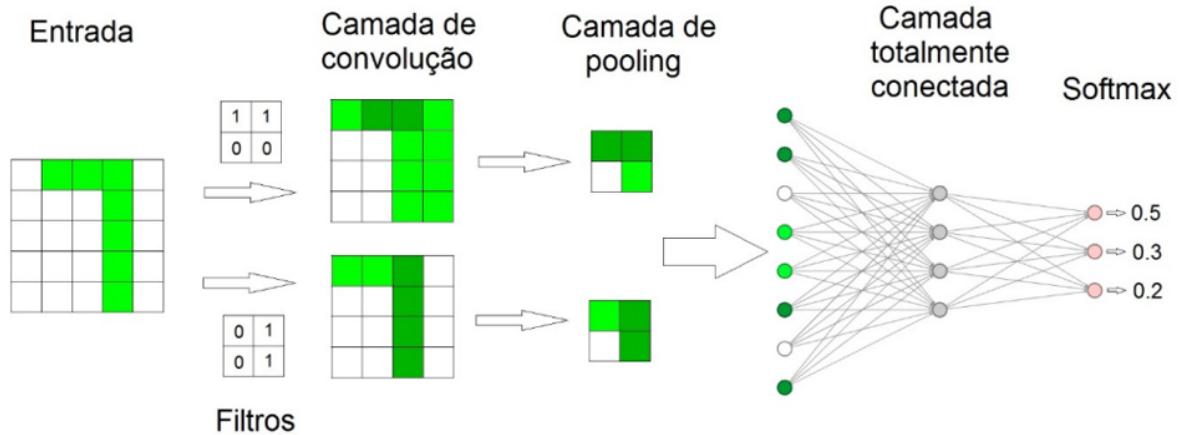
## 2.2 CONVOLUTIONAL NEURAL NETWORKS (CNN)

Uma Rede Neural Convolucional, ou em inglês Convolutional Neural Network – CNN, é uma variação de uma rede Perceptrons de múltiplas camadas (*Multilayer Perceptron - MLP*), elas são uma especialização de Rede Neural Profunda (Deep Neural Network), o processamento de dados é feito a partir de matrizes multidimensionais originarias de uma imagem, a rede aplica diversos filtros de dados visuais, mantendo a relação entre os pixels da imagem ao longo do processamento da rede.

A CNN foi proposta pela primeira vez por Lecun et al. (1998). Em 2012, as CNNs se tornaram o estado da arte nas tarefas de classificação de imagens quando Krizhevsky et al. (2012) chamou a atenção para o famoso modelo “AlexNet”.

A estrutura básica de uma CNN é mostrada na figura 3.

Figura 3 – Exemplo de uma arquitetura de CNN comum.



Fonte: Ebermam (2018)

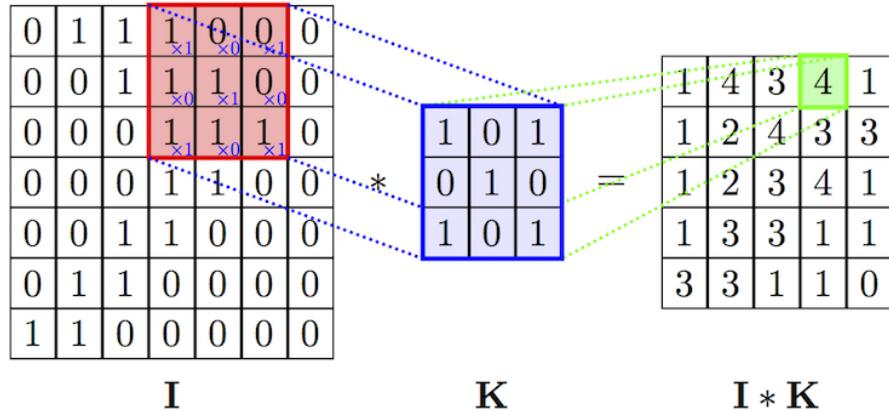
A principal função de uma rede neural convolucional é a extração de características do objeto de entrada, em que para realizar essa tarefa são aplicadas diversas funções matemáticas no decorrer da rede a partir operações com matrizes, que são chamados de kernels. Os kernels funcionam basicamente como filtros, que realizam diferentes tipos de transformações nos dados a cada seguimento da imagem, varrendo a imagem por completo gerando assim um mapa de característica. Essa ação dá o nome ao tipo de rede e é chamada de convolução (VARGAS; CARVALHO; VASCONCELOS, 2019).

A equação simplificada da convolução com os dados de entrada gerando um mapa de característica é representada na equação 2.1 (VO, 2018).

$$Sada = K * I + b \quad (2.1)$$

A figura 4 exemplifica a equação simplificada de convolução, envolvendo então a multiplicação do valor do kernel (filtro) "K", em azul, pelos valores das entradas da imagem "I", representado em vermelho, e em seguida, os produtos são somados para obter o resultado saída, representado em verde. O termo b é o bias, ele é adicionado a essa soma ponderada, se ajustando nos diferentes níveis de intensidade e deslocamento no conjunto de dados, se assemelhando ao peso de um neurônio, permitindo que a que a camada modele melhor os dados (VO, 2018).

Figura 4 – Exemplo de operação de convolução.



Fonte: Vo (2018)

Posteriormente ao filtro de convolução, é aplicado uma função de ativação, esta realiza transformações nos dados recebidos em cada neurônio (VARGAS; CARVALHO; VASCONCELOS, 2019). A função de ativação propaga o gradiente de forma eficiente fazendo com que não ocorra problema denominado "*vanishing gradient*" (PASCANU; MIKOLOV; BENGIO, 2013) e resolve o efeito de cancelamento que pode ter nas camadas seguintes. A função de ativação mais comum é a *Relu* e é definida pela equação 2.2

$$R(z) = \max(0, z) = \max(0, W * x + b) \quad (2.2)$$

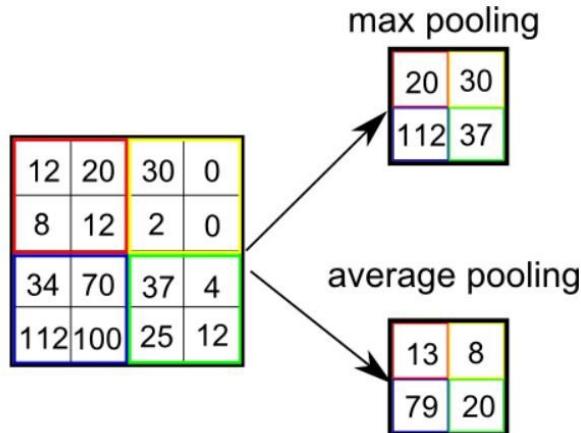
onde  $z$  são os dados advindos da convolução de um *kernel*  $W$  com os dados de entrada  $x$  mais o *bias*.  $R(z)$  já é a resposta da função de ativação.

Em sequência os dados são passados por uma camada de agrupamento, que são comumente chamadas de *pooling*, e esta tem a função de diminuir o tamanho dos mapas de características mantendo as informações mais relevantes, redimensionar os dados da rede, por conseguinte, aumentar a agilidade da rede e criar invariância espacial (VARGAS; CARVALHO; VASCONCELOS, 2019).

Os métodos mais comuns são *pooling* de máximo e *pooling* de média. A figura 5 mostra os dois exemplos de *pooling* comentados.

Após o *pooling*, pode haver uma série de concatenação de mais camadas convolu-

Figura 5 – Exemplo de operação de *pooling*.



Fonte: Saha (2018)

cionais, de funções de ativação e de *pooling*. No final dessa sequência, ocorre a vetorização da última camada de *pooling*, transformando-a em um vetor unidimensional. Esse vetor resultante é então encaminhado para a próxima etapa (EBERMAM, 2018).

Para que uma rede neural convolucional consiga realizar uma tarefa de classificação, é preciso que o vetor unidimensional seja passado por no mínimo uma camada de neurônios totalmente conectada, essa camada traça os caminhos de decisões a partir dos dados vindos dos filtros das camadas anteriores de cada classe de resposta. Essas camadas são parecidas com uma *Multilayer Perceptron* (MLP), com a única diferença que em uma MLP comum, os dados de entrada são dados brutos. Já em uma CNN, a entrada são os dados vetorializados dos mapas de ativação gerados pela concatenação das camadas convolucionais, funções de ativação e *pooling* (VARGAS; CARVALHO; VASCONCELOS, 2019).

Por fim, para obter uma saída, é necessário aplicar uma função de ativação ao final da camada densa. A quantidade de neurônios no final da camada densa é definida pelo número de classes do problema e a função de ativação mais utilizada é a *softmax*, onde sua principal função é transformar a saída final da camada densa em valores de (0,1), onde a soma de todos os valores da saída serão iguais a 1. Assim esses novos valores podem ser interpretados como uma probabilidade. A função *softmax* é descrita pela equação 2.3:

$$y_j = \frac{e^{y_{(j)}}}{\sum_{i=1}^n e^{y_i}} \quad (2.3)$$

Onde a saída  $y$  do neurônio  $j$  passa a ser o valor da mesma dividido pelo somatório de todas as saídas dos neurônio da camada final (EBERMAM, 2018).

### 2.3 YOLO

As redes neurais convolucionais estão sendo abrangemente usadas para classificação de imagens e detecções de objetos. No momento, as redes baseadas em redes neurais convolucionais capazes de realizar detecção de objetos em tempo real são SSD (Single Shot Detector), YOLO (You-OnlyLook-Once), R-FCN (regionbased fully convolutional network), e RetinaNet (NATHA; BEHZADANB; PAALA, 2020).

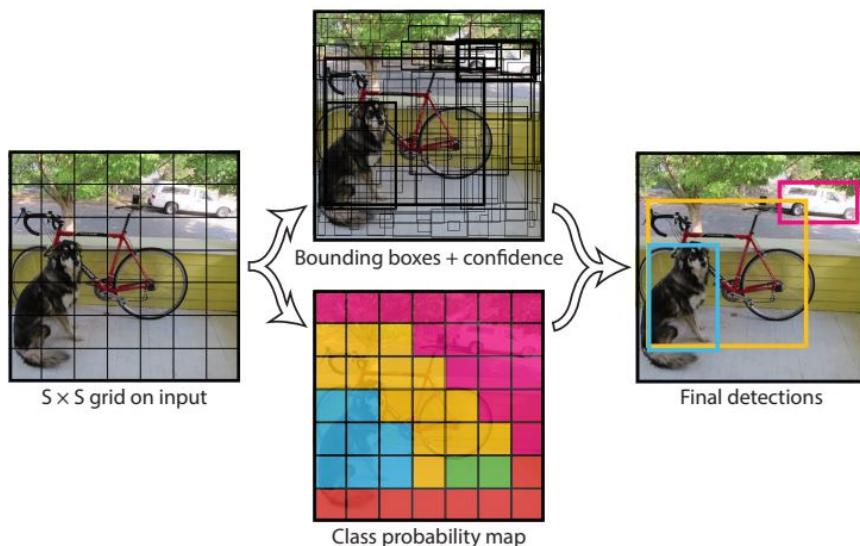
Segundo Natha, Behzadanb e Paala (2020). o desafio de obter computação em tempo real é manter uma precisão satisfatória. Embora na maioria das vezes, os algoritmos rápidos comprometam significativamente a precisão para obter computação em tempo real, até o momento, apenas YOLO é mais rápida e mais precisa do que as alternativas descritas acima.

A rede YOLO foi proposta pela primeira vez por Joseph Redmon et al. A YOLO é uma rede neural profunda com 24 camadas convolucionais seguidas por 2 camadas totalmente conectadas. As primeiras camadas convolucionais são responsáveis por extrair características da imagem, enquanto as camadas totalmente conectadas utilizam regressão para prever as probabilidades de saída bem como as coordenadas da caixa delimitadora, realizando essa operação para cada pixels da imagem varrendo a imagem inteira durante o treinamento e o tempo de teste. A rede utiliza recursos da imagem completa para prever cada caixa delimitadora (REDMON, 2016).

A imagem de entrada é dividida em quadrados de tamanho  $S \times S$ . Cada quadrado na grade prevê uma quantidade de caixas delimitadoras ( $B$ ) atrelado ao valor de confiabilidade ( $C$ ). A confiança é definida como a predição do objeto ( $\text{Pr}(\text{Objeto})$ ) multiplicado pela sobreposição da união (IOU), que mostra o quanto confiante o modelo está ao colocar o objeto naquela caixa e o quanto preciso a rede acredita que seja

(REDMON, 2016).

Figura 6 – Modelo YOLO.



Fonte: REDMON (2016)

Se não houver nenhum objeto na célula, as pontuações de confiança devem ser zero.

Se o objeto existir, a pontuação de confiança deve ser igual à interseção sobre a união (*Intersection over Union - IOU*) (REDMON, 2016).

Para calcular o IoU, é considerada a área de interseção entre as caixas delimitadoras e as caixas delimitadoras de referência (ground truth), bem como a área total coberta pelas duas caixas delimitadoras - também conhecida como União (PADILLA; NETTO; SILVA, 2020).

A interseção dividida pela União é o IoU, o que significa que é a razão entre a sobreposição e a área total, estimando o quão próxima a caixa delimitadora de previsão está da classe original. Conforme mostrado na figura 7 (PADILLA; NETTO; SILVA, 2020).

Por fim, cada caixa delimitadora é representada por cinco valores:  $bx$ ,  $by$ ,  $bw$ ,  $bh$  e confiança. A saída tem o formato de um tensor de  $S \times S$  ( $5 \times B + C$ ). Quando vários quadros estão prevendo o mesmo objeto, a rede utiliza uma técnica de supressão não máxima para encontrar o quadro mais adequado (BOCHKOVSKIY; WANG; LIAO, 2020).

A rede YOLO é disponibilizado em várias versões, indo desde a YOLOv1 até a YO-

Figura 7 – Intersection Over Union (IOU)

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$

Fonte: REDMON (2016)

LOv3, desenvolvidas pelos autores originais REDMON (2016), seguindo pela YOLOv4 desenvolvida por Bochkovskiy, Wang e Liao (2020) e na sequência existem as versões YOLOv5 até YOLOv8, que foram desenvolvidas pela organização chamada Ultralytics LLC. Cada uma dessas versões apresentam suas características e desempenho próprio.

O projeto em questão se baseou em dois tipos de rede YOLO, a YOLOv4 e a YOLOv8.

### 2.3.1 YOLOv4

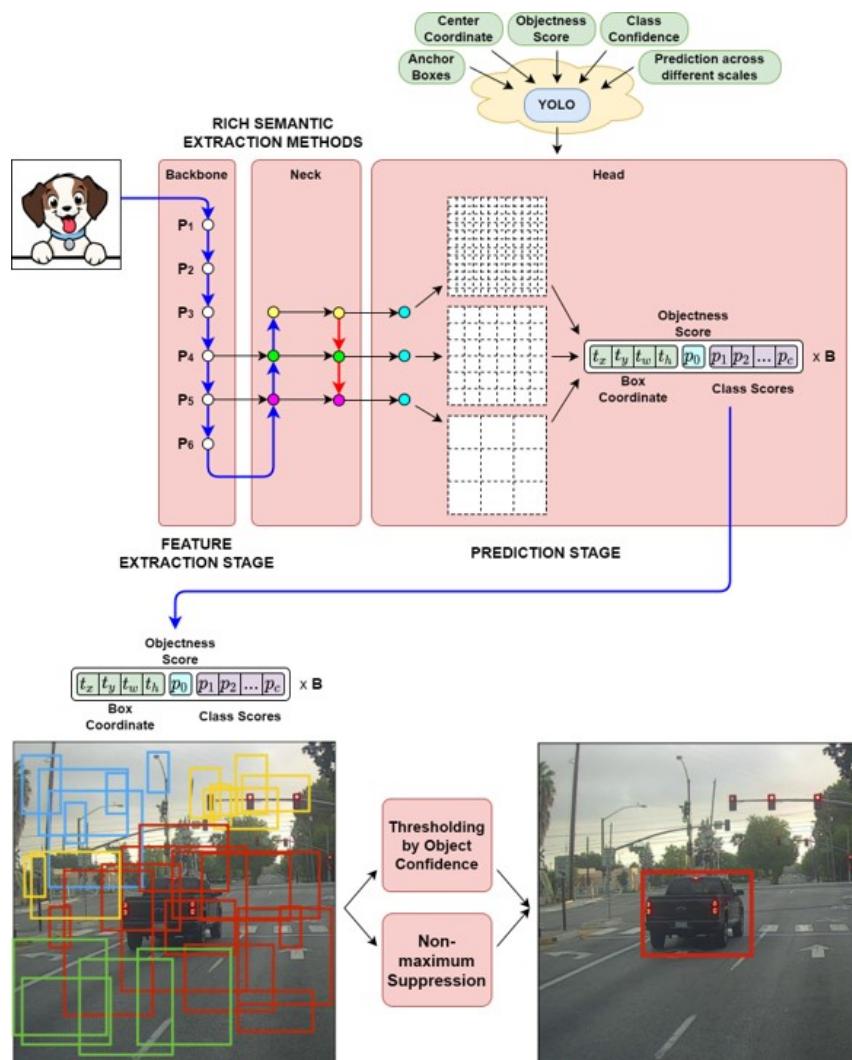
A YOLOv4 foi proposta por Bochkovskiy, Wang e Liao (2020) como uma melhoria das versões anteriores. Ele segue a mesma teoria da primeira rede YOLO, mas inclui uma combinação de mudanças no *design* arquitetônico e nas metodologias de treinamento, com uma rede convolucional muito mais complexa (BOCHKOVSKIY; WANG; LIAO, 2020).

A arquitetura de alto nível da rede YOLOv4 é composta por três partes principais. A primeira parte é a *backbone*, que utiliza a CSPDarknet53 CNN como extrator de características. A *backbone* é responsável por extrair características significativas da imagem de entrada (BOCHKOVSKIY; WANG; LIAO, 2020).

A segunda parte é o *neck*, que consiste em um conjunto de camadas adicionais usadas para extrair diferentes características das camadas da "backbone". Isso ajuda a aumentar o poder de representação da rede, capturando características em múltiplas escalas (BOCHKOVSKIY; WANG; LIAO, 2020).

A terceira parte é a *head*, também conhecida como *dense prediction*, que realiza a detecção de objetos e a regressão das caixas delimitadoras. Ela utiliza as características extraídas pela *backbone* e *neck* para prever as coordenadas da bounding box como valores relativos à célula da grade. Esses valores relativos são decodificados em coordenadas absolutas na imagem. Após a decodificação das bounding boxes, é realizado a supressão de não máximos (non-maximum suppression) para eliminar detecções duplicadas ou de baixa confiança. E, por fim, são associadas aos objeto detectados os rótulos de cada classe. E atrelado a classe a probabilidade mais alta de acordo com um limiar previamente definido para produzir os resultados finais de detecção, incluindo as caixas delimitadoras previstas (BOCHKOVSKIY; WANG; LIAO, 2020).

Figura 8 – Arquitetura YOLOv4.



Fonte: Bochkovskiy, Wang e Liao (2020)

### 2.3.2 YOLOv8

É a mais recente versão da YOLO e traz uma série de melhorias incrementais em relação às versões anteriores, como o aumento de precisão e velocidade.

O YOLOv8 foi lançado em janeiro de 2023 pela Ultralytics. O YOLOv8 oferece cinco versões dimensionadas: YOLOv8n (nano), YOLOv8s (pequeno), YOLOv8m (médio), YOLOv8l (grande) e YOLOv8x (extra grande). Diferente das outras redes YOLO, ela realiza várias tarefas de visão computacional, como detecção de objetos, segmentação, estimativa de pose, rastreamento e classificação (TERVEN; CORDOVA-ESPARZA, 2023).

A arquitetura macro da YOLOv8 também consiste em *backbone*, *neck* e *head* sendo representada na figura 9.

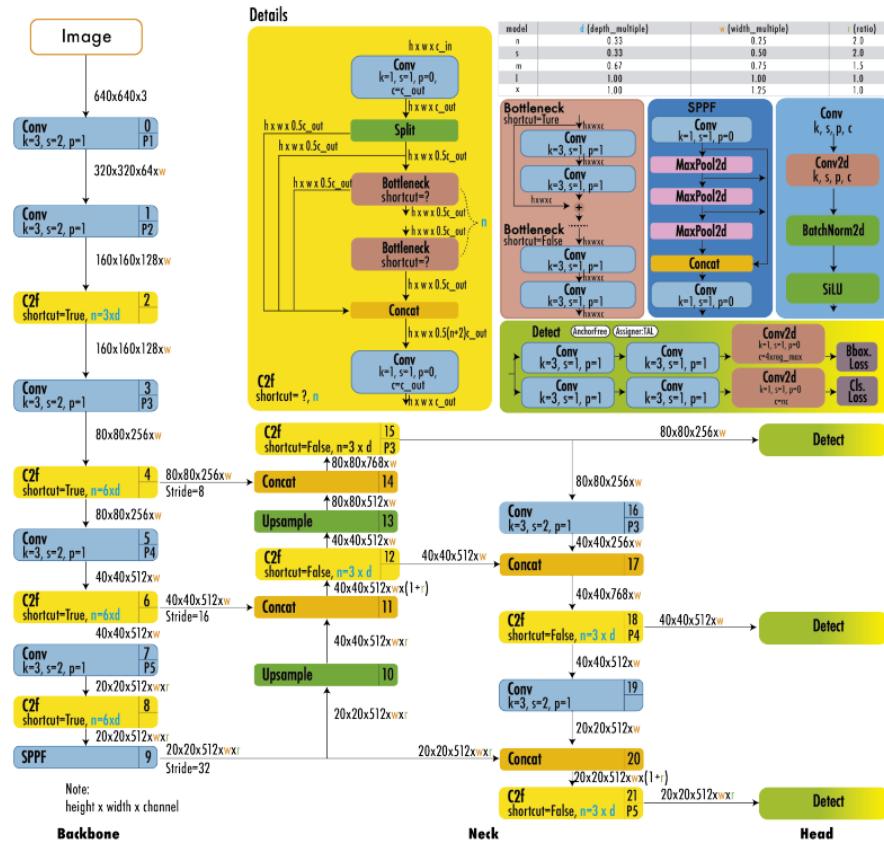
A camada *backbone* é similar a rede YOLOv4 com uma CNN baseada na CSPDarknet53 como extrator de característica porém com mudanças em sua estrutura com a inclusão de um módulo chamado C2f que combina características de alto nível com informações contextuais para melhorar a precisão da detecção (TERVEN; CORDOVA-ESPARZA, 2023).

A camada de *neck* da YOLOv8, também extrai diferentes características de diferentes camadas da *backbone* e tamém utiliza o módulo C2f (TERVEN; CORDOVA-ESPARZA, 2023).

Na camada final conhecida como *head*, é onde ocorre a predição das classes dos objetos bem como sua localização. Na arquitetura do YOLO, geralmente existem várias camadas de saída (*output heads*), porém a YOLOv8 tem apenas uma, pois a rede não trabalha com o metodo de detecção utilizando Ancoras (*anchor boxes*) de diferentes tamanhos, ao invés disso a rede YOLOv8 utiliza o mecanismo de detecção sem âncora (*anchor free detection*) que prevê a localização diretamente o centro de um objeto, em vez de um deslocamento em relação a uma caixa-âncora conhecida. Na camada *head* para a definição da probabilidade, se uma bouding box contém um objeto, é utilizado a função de ativação *sigmoid*, e para a definição de qual classe o objeto corresponde, é

utilizado a função *softmax* (REIS *et al.*, 2023).

Figura 9 – Arquitetura YOLOv8.



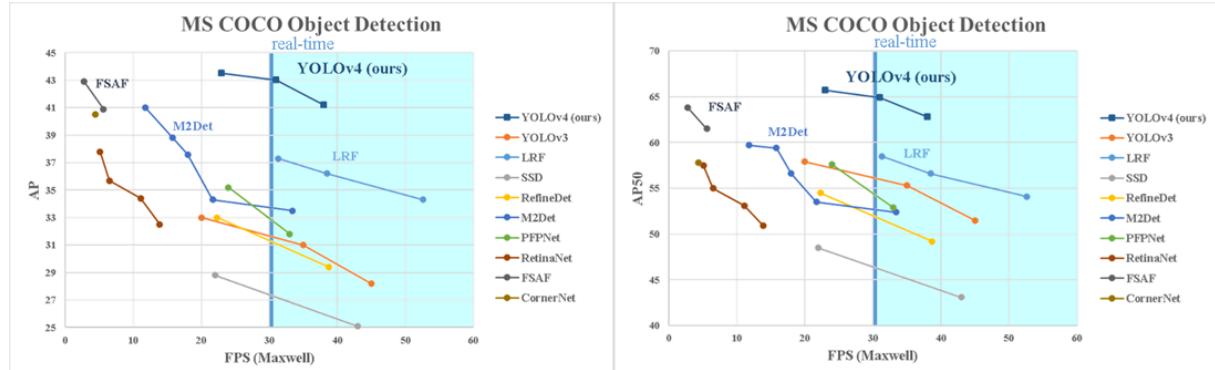
Fonte: Terven e Cordova-Esparza (2023)

### 2.3.3 Comparação de desempenho da YOLOv4 e YOLOv8

Como evidenciado na Figura 10, a YOLOv4 supera as versões anteriores da YOLO, bem como outras redes, comprovando sua superioridade quando testada em diversos ranges de FPS, alcançando uma precisão de 43% em AP50:95 e 65,7% em AP50 com 30fps a partir do conjunto de dados Microsoft COCO (BOCHKOVSKIY; WANG; LIAO, 2020).

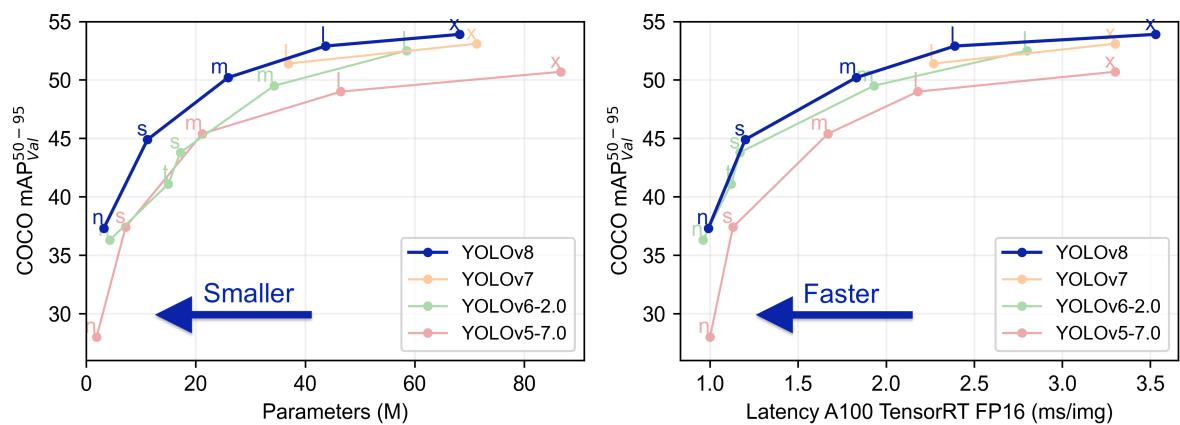
Já a figura 11 demonstra que, ao comparar o YOLO-v8 com as versões YOLO-v7, YOLO-v6 e o YOLOv5, treinados com uma resolução de imagem de 640, todas as variantes do YOLO-v8 apresentam melhor rendimento com um número semelhante de parâmetros, indicando que as modificações em sua arquitetura resultaram no aumento da eficiência da rede (HUSSAIN, 2023).

Figura 10 – Desempenho da rede YOLOv4



Fonte: Bochkovskiy, Wang e Liao (2020)

Figura 11 – Desempenho da rede YOLOv8



Fonte: Hussain (2023)

### 3 MATERIAIS E MÉTODOS

Este capítulo aborda a metodologia utilizada na execução deste trabalho. Os EPIs definidos para a identificação nesse projeto foram o capacete, óculos de proteção e luva, a decisão da identificação desses equipamentos de proteção se deu ao fato de que, durante a revisão bibliográfica, estes são os mais utilizados no meio laborativo em diversos segmentos industriais, e além de serem responsáveis pela proteção dos principais membros e órgãos do corpo humano, onde intactos promovem o bem estar do colaborador durante sua vida cotidiana e trabalhista. O capacete pois protege a cabeça a lesões que podem provocar sequelas ou até óbito, os óculos de proteção pois protege os olhos de exposição a particulados que podem causar feridas ou até perda de visão, e luva pois protege de lesões nas mãos, sendo cortes ou até esmagamento.

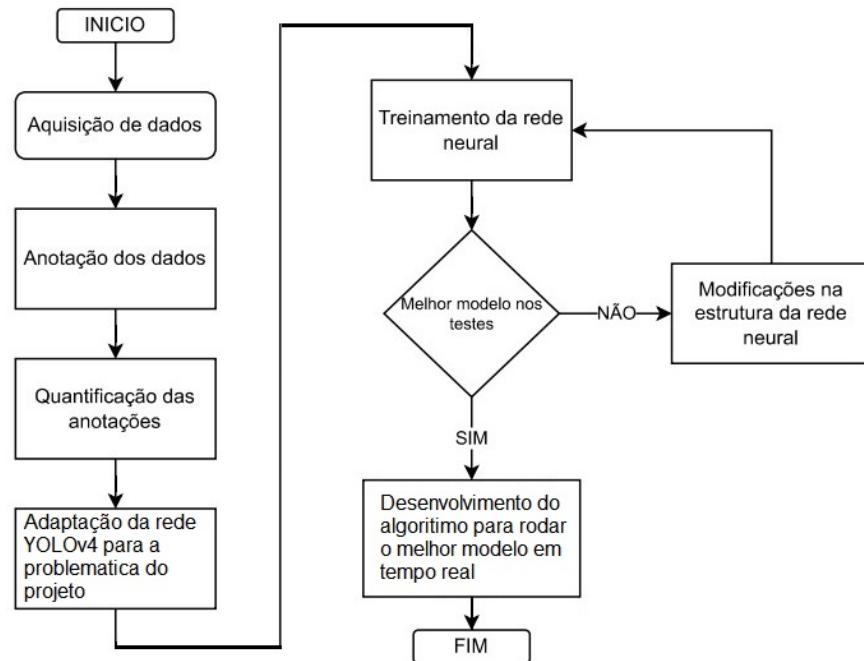
O projeto foi dividido em seis etapas principais: a aquisição de imagens, a anotação de classes de cada EPI, a quantificação das anotações no banco de dados, modelagem do algoritmo de rede neural capaz de classificar e delimitar pessoas usando ou não EPI, modificações nos parâmetros da rede neural a fim de obter o melhor desempenho e, por fim, o desenvolvimento de um algoritmo que utiliza a rede já treinada para identificar EPI quadro a quadro em um vídeo simulando uma rede de CCTV.

#### 3.1 ETAPA 1 - AQUISIÇÃO DE DADOS

O conjunto de dados usado para a elaboração do projeto contém 7555 imagens de pessoas em um ambiente de trabalho diverso usando ou não Equipamentos de Proteção Individual (EPIs) de diversos modelos, formas e cores.

Todas as imagens estão em domínio público, sendo que 7041 imagens foram adquiridas do site "<https://public.roboflow.com/>"(ROBOFLOW, 2018) e as outras 514 foram adquiridas por meio de pesquisa no Google Imagens. As resoluções das imagens variam de 151x184 a 7360x4912. Devido ao conjunto de imagens ser grande e exclusivo de pessoas em ambiente laboral não foi necessário realizar a coleta de imagens em campo.

Figura 12 – Fluxograma das etapas do trabalho



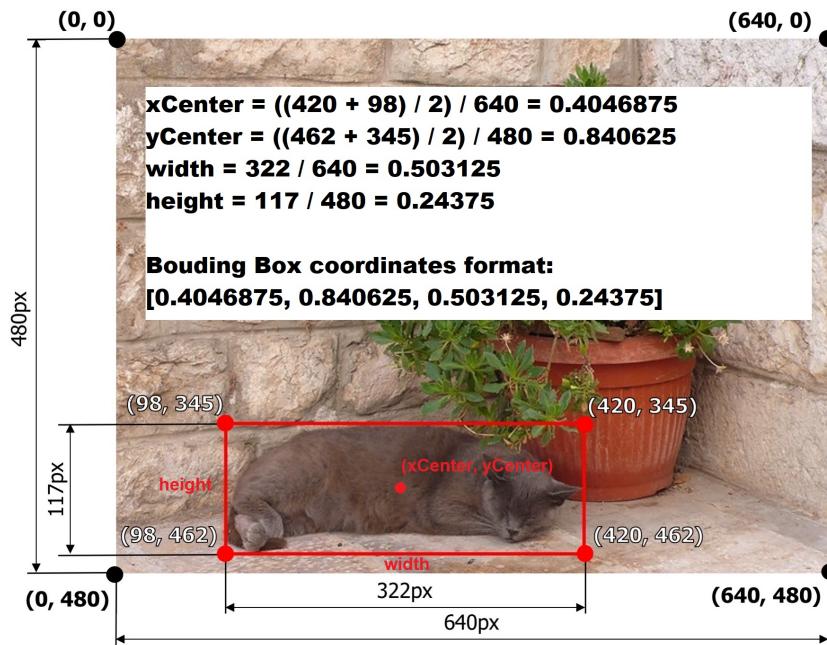
Fonte: Elaborado pelo próprio Autor (2023)

### 3.2 ETAPA 2 - ANOTAÇÃO DOS DADOS

Assim como qualquer outra rede neural convolucional, o YOLO também precisa ser alimentado com dados pré-classificados. Por padrão, essa rede utiliza para cada imagem no banco de dados um arquivo ".txt" com o mesmo nome do arquivo da imagem. Este arquivo de texto é responsável por informar à rede as classes dos objetos, bem como suas localizações na imagem. Para localizar o objeto na imagem, o YOLO trabalha com caixas delimitadoras, em um tipo específico de coordenada. A caixa delimitadora é representada por quatro valores [xCentro, yCentro, largura, altura]. Todos os valores são normalizados com a largura máxima e a altura máxima. A Figura 13 detalha um exemplo do cálculo do padrão de anotação de um objeto no modelo em que a rede YOLO necessita para funcionar de forma correta.

As classes de objetos no arquivo de texto são anotadas com números inteiros. No caso deste projeto, foram definidos 6 tipos diferentes de classes, variando de 0 a 5, onde cada classe é anotada, respectivamente, como "sem capacete", "com capacete", "sem luva", "com luva", "sem óculos" e "com óculos".

Figura 13 – Cálculos das coordenadas da caixa delimitadora para modelo de rede neural YOLO



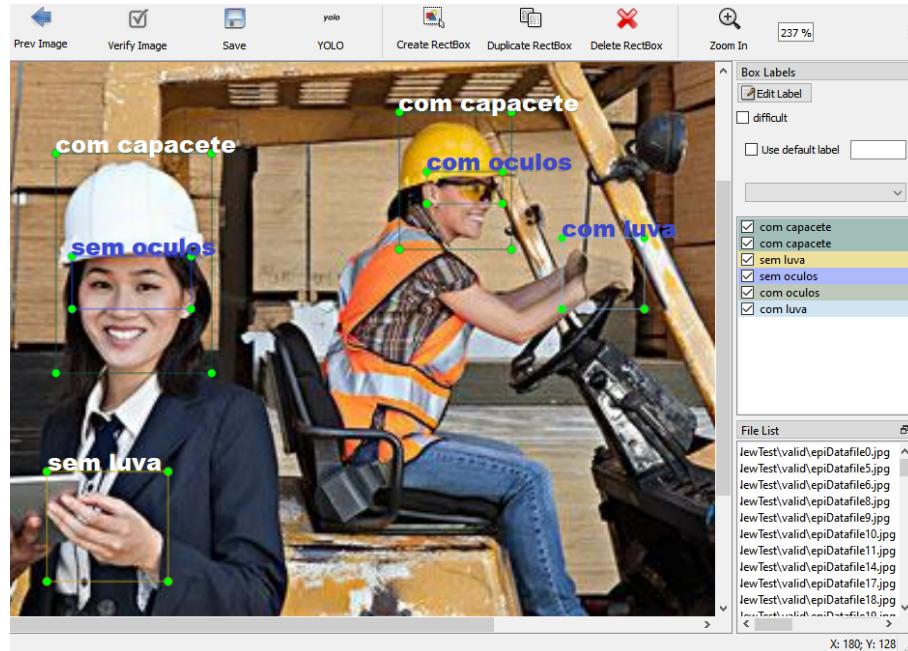
Fonte: Albumentations (2019)

Para anotar cada objeto encontrado nas 7555 imagens no formato de coordenadas padrão da rede YOLO, foi necessário utilizar o programa chamado *LabelIMG*, um software de código aberto exclusivo para anotação de imagens escrita em Python e que utiliza Qt, um framework C++ multiplataforma que é comumente usado para desenvolver interfaces gráficas. A ferramenta foi criada por Tzutalin (2017), e pode ser encontrada no repositório da Github.

No *software* cada imagem é aberta uma por uma. Em seguida o programa permite selecionar manualmente o objeto que deseja ser identificado, é utilizado a opção de demarcar o objeto com uma *bouding box*, selecionando assim o objeto em sua totalidade respeitando o seu limite para de certa forma especificar sua localização exata na imagem, o rotulo é atribuído conforme mencionado anteriormente como se pode observar na figura 14.

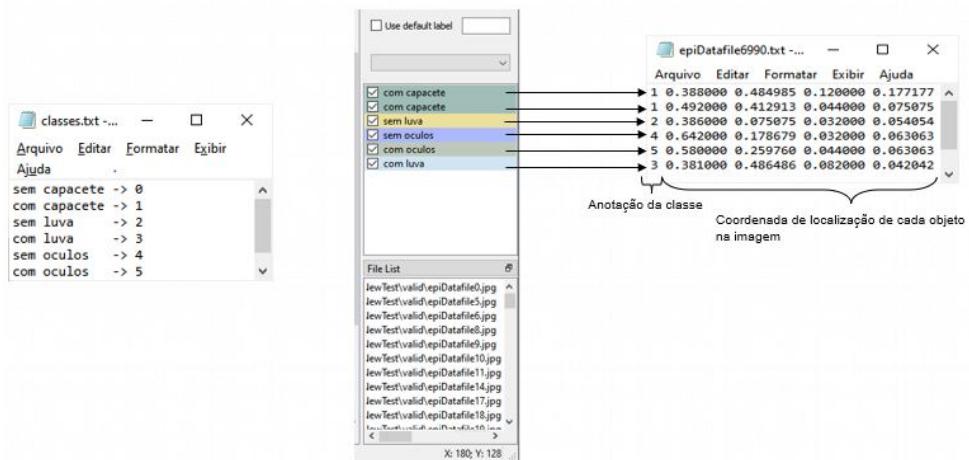
O programa então salva as coordenadas e o tipo dos objetos anotados em um arquivo ".txt" de mesmo do arquivo aberto. Observe na figura figura 15.

Figura 14 – Anotação de trabalhadores portando ou não os EPIs pelo programa LabelImg



Fonte: Elaborado pelo próprio Autor (2023)

Figura 15 – Arquivos .txt gerado pelo programa, relacionando a anotação com a coordenada da localização do objeto.



Fonte: Elaborado pelo próprio Autor (2023)

### 3.3 ETAPA 3 - QUANTIFICAÇÃO DAS ANOTAÇÕES NO BANCO DE DADOS

Depois da anotação de todo o banco de dados, um algoritmo Python foi desenvolvido para quantificar as classes em cada imagem em todo o *dataset*, a fim de auxiliar na etapa de particionamento do banco de dados em um conjunto de treinamento e um conjunto de validação.

Foram anotados as seguintes quantidades totais de objetos no *dataset* de acordo com os rótulos:

Tabela 1 – Quantidade de rótulos no *dataset* completo

Anotação	Quantidade
Sem Capacete	6888
Com Capacete	20021
Sem Luva	5562
Com Luva	3658
Sem Óculos	2886
Com Óculos	743

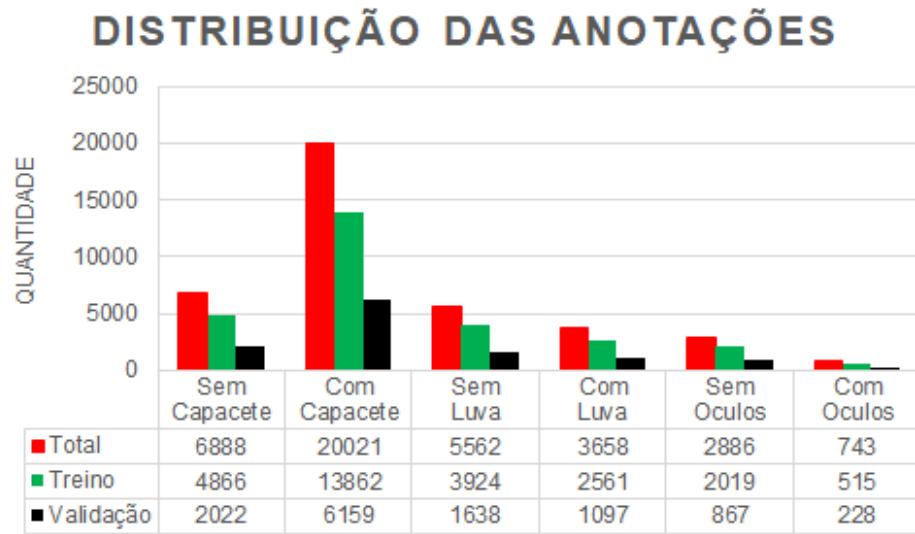
Fonte: Elaborado pelo próprio Autor (2023)

O YOLO requer por padrão o particionamento do *dataset* em validação e treinamento. Este conjunto foi dividido em 70% para treinamento e 30% para validação. Essa proporção não foi aplicada ao número total de imagens, devido ao fato de que existem imagens com várias quantidades de objetos da mesma classe e de classes diferentes, então probabilisticamente, o conjunto de treinamento e validação estaria desbalanceado em relação às classes, levando a um aprendizado ineficiente. Para evitar esse problema, o algoritmo filtrou as classes de imagens para que todas as classes em seus conjuntos de treinamento e validação sejam distribuídas em uma proporção muito próxima de 70/30. A Figura 18 mostra a quantidade total de anotações de cada classe distribuída pelo conjunto de treinamento/validação.

### 3.4 ETAPA 4 - ADAPTAÇÃO DAS REDES YOLOV4 E YOLOV8 PARA A PROBLEMATICA DO PROJETO

Para o desenvolvimento do projeto foi utilizada a plataforma on-line Google Collaboratory (Uma plataforma de nuvem que permite aos usuários executar código Python ou outras linguagens de programação em um ambiente Jupyter Notebook). Essa escolha foi feita devido à facilidade de instalação e configuração dos *frameworks* responsáveis por construir as redes. Para o YOLOv4, o *Darknet*, um *framework* de rede neural de código aberto escrito em C e CUDA e para o YOLOv8 o *PyTorch* também de código escrito em python que oferece uma ampla gama de ferramentas e bibliotecas

Figura 16 – Quantificação das anotações divididas entre treino e validação na proporção 70/30.



Anotação	Treino	Validação
Sem Capacete	70,64%	29,36%
Com Capacete	69,24%	30,76%
Sem Luva	70,55%	29,45%
Com Luva	70,01%	29,99%
Sem Óculos	69,96%	30,04%
Com Óculos	69,31%	30,69%

Fonte: Elaborado pelo próprio Autor (2023)

que facilitam a criação e treinamento de redes neurais. Ambos podem ser clonados rapidamente dos repositórios dos servidores GitHub dos autores originais.

O Google Colaboratory também fornece acesso limitado aos GPUs dos servidores Google, que durante o treinamento da rede reduz consideravelmente o tempo de processamento.

#### 3.4.1 Configurações iniciais da rede YOLOv4

Para a utilização da rede YOLOv4 de forma correta e satisfatória se fez necessário realizar uma sequência de configurações iniciais a fim de atender a demanda proposta.

Em primeiro momento foi necessário clonar o repositório do *darknet* criado por Boch-

kovskiy, Wang e Liao (2020), que contém todos os arquivos e ferramentas necessárias para fazer o treinamento do modelo e montar o *framework* utilizando as linhas de código abaixo:

```
1 !git clone https://github.com/AlexeyAB/darknet
2 !make
```

Uma vez instalado e montado, foi necessário realizar modificações no arquivo de configuração ("yolov4.cfg") da rede yolo para a demanda do projeto. O arquivo de configuração vem como default para treinar objetos de um conjunto de dados usualmente chamado de COCO, ou *Common Objects in Context* em inglês, sendo um conjunto de dados de imagens e rótulos de objetos para aprendizado de máquina contendo mais de 80 categorias de objetos, incluindo pessoas, animais, veículos e objetos cotidianos.

`max_batches = 2000 * 6` (Número de classes existente no projeto)

`steps = max_batches * 0.8, max_batches * 0.9`

Figura 17 – Configurações aplicadas no arquivo ".cfg".

---

```
21 burn_in=1000
22 max_batches = 12000
23 policy=steps
24 steps=9600,10800
25 scales=.1,.1
```

Fonte: Elaborado pelo próprio Autor (2023)

Em cada camada Yolo do arquivo ".cfg" foi alterado o valor:

`classes = Número de classes = 6`

E antes de cada camada Yolo, foi alterado o filtro da camada convolucional para:

`filters = (Número de classes + 5) * 3 = (6 + 5)*3 = 33`

Feita essas alterações, o arquivo é salvo como "yolov4\_custom.cfg" na pasta do principal do projeto.

Figura 18 – Configurações aplicadas no arquivo ".cfg".

```

959 [convolutional]
960 size=1
961 stride=1
962 pad=1
963 filters=33
964 activation=linear
965
966
967 [yolo]
968 mask = 0,1,2
969 anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76,
970 classes=6
971 num=9
972 jitter=.3
973 ignore_thresh = .7
974 truth_thresh = 1

```

Fonte: Elaborado pelo próprio Autor (2023)

Em seguida foi preciso criar dois arquivos, chamados obj.names e obj.data e salvá-lo dentro da pasta do Colab.

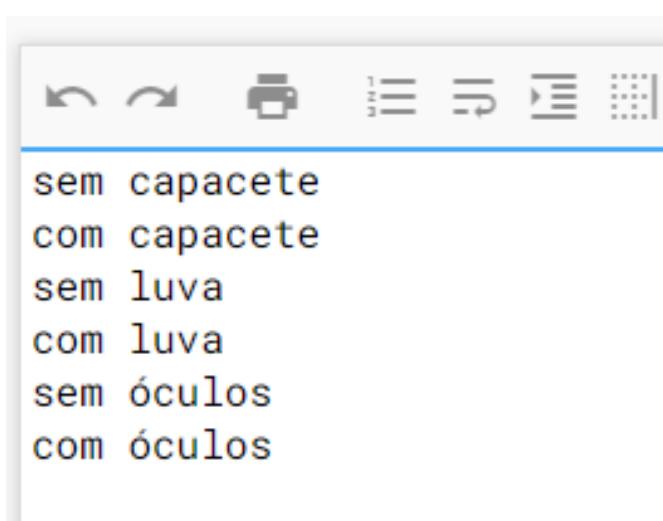
```

1 !touch obj.names
2 !touch obj.data
3 !cp obj.names /content/gdrive/MyDrive/YOLOppe/obj.names
4 !cp obj.data /content/gdrive/MyDrive/YOLOppe/obj.data

```

"obj.names" sendo um arquivo ".txt" que contém os nomes das classes utilizadas no treinamento da rede, como descrito na figura 19

Figura 19 – Edição do arquivo "obj.names"



Fonte: Elaborado pelo próprio Autor (2023)

Para o arquivo "obj.data" foi inserido as informações sobre o local do diretório onde serão armazenados os arquivos utilizados para o treinamento.

Figura 20 – Edição do arquivo "obj.data"

```

classes = 6
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /yolo/

```

Fonte: Elaborado pelo próprio Autor (2023)

Ambos também salvos na pasta principal do projeto.

Para finalizar as configurações iniciais, foi preciso compactar os datasets de treino e validação para os arquivos "obj.zip" e "valid.zip" respectivamente e salvá-las na pasta principal do projeto. Em seguida para cada arquivo compactado foi gerado um arquivo ".txt" contendo os nomes de todas as imagens do *dataset*, tal arquivo é utilizado para acessar a imagem pela rede durante o treinamento.

### 3.4.2 Treinando a rede YOLOv4

Após as configurações a rede está pronta para iniciar o processo de aprendizado. Foi preciso então, clonar o repositório do *darknet* novamente e ativar o suporte da GPU/CUDA modificando o arquivo *Makefile* responsável por montar o *framework*, utilizando o código abaixo.

```

1 !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
2 !sed -i 's/GPU=0/GPU=1/' Makefile
3 !sed -i 's/CUDNN=0/CUDNN=1/' Makefile

```

Posteriormente foi necessário montar o framework, extrair as imagens dos arquivos compactados e copiar os arquivos criados conforme mostrado na seção anterior para a pasta de dados e configuração do *darknet*, utilizando as linhas e código abaixo:

```

1 !make
2 !unzip /yolo/obj.zip -d ./data/
3 !unzip /yolo/valid.zip -d ./data/
4 !cp /yolo/yolov4_custom.cfg ./cfg/
5 !cp /yolo/obj.names ./data
6 !cp /yolo/obj.data ./data
7 !cp /yolo/train.txt ./data
8 !cp /yolo/test.txt ./data

```

Para o treinamento, foi utilizado uma técnica chamada transferência de aprendizado, onde inicialmente foi utilizada uma base convolucional pública já treinada com o conjunto de dados COCO, a base convolucional consegue identificar pessoas e cerca de 80 objetos do meio cotidiano. Essa técnica reduz o tempo de processamento de uma nova rede que identifica objetos personalizados, no caso do projeto os EPIs, pois os pesos da camada de convolução já estão pré-calculados, não sendo necessário partir do zero.

Para baixar a base convolucional, comumente chama de *weight*, ou peso em português se fez necessário utilizar o código abaixo:

```

1 !wget https://github.com/AlexeyAB/darknet/releases/download/
1 darknet_yolo_v3_optimal/yolov4.conv.137

```

Após baixarmos o peso, foi possível utilizá-lo para iniciar o treinamento da rede para identificação das anotações do problema proposto no projeto, utilizando o comando:

```

1 !./darknet detector train data/obj.data cfg/yolov4_custom.cfg
1 yolov4.conv.137 -dont_show -map | tee /yolo/results.log

```

O *darknet* permite a visualização do treinamento em tempo real, permitindo acompanhar as principais métricas durante a evolução treinamento. Como se pode observar na figura 21 sendo uma parte do resultado dinâmico do treinamento, é evidenciado as seguintes métricas:

- **mAP**: *Mean average precision* ou média aritmética das precisões, em português,

é uma medida de quão bem um modelo pode detectar objetos em imagens. É calculada fazendo a média das pontuações de precisão e recall para cada classe de objeto.

- **Average loss:** É uma medida de quão bem o modelo está se saindo no geral. Quanto menor a perda média, melhor o desempenho do modelo. Ela é calculada fazendo a média das perdas para cada região em uma imagem.
- **IOU loss:** Mede o quão bem a caixa delimitadora prevista pelo modelo se sobrepõe à caixa delimitadora real (ground truth). A caixa delimitadora real é aquela desenhada manualmente ao redor do objeto na imagem. Um valor perfeito de IOU seria 1.0, enquanto um valor de 0.0 significaria que a caixa delimitadora prevista não se sobrepõe à caixa delimitadora real de forma alguma.
- **Class loss:** Mede o quão bem o modelo prevê a classe correta para o objeto na região. Um valor perfeito de perda de classe seria 0.0, enquanto um valor de 1.0 significaria que o modelo previu a classe errada para o objeto.
- **Total loss:** É definido como a soma das perdas para cada região em uma imagem.

```
(next mAP calculation at 10286 iterations)
9966: 1.971451, 2.857120 avg loss, 0.000130 rate, 10.386220 seconds, 637824 images, 8.602023 hours left
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.873675), count: 1,
class_loss = 0.065885, iou_loss = 9.318471, total_loss = 9.384356
```

Figura 21 – Recorte da visualização da interação do treinamento e suas respectivas métricas

Fonte: Elaborado pelo próprio Autor (2023)

Como configurado anteriormente a cada 326 interações aproximadamente é realizado o cálculo do mAP e salvo de forma automática a base convolucional referente a este cálculo na pasta raiz do projeto como "yolov4\_custom\_last.weights". Se em um determinado momento da interação, o cálculo de mAP da base convolucional sobressair dos demais durante o processo de treinamento, ele é salvo de forma exclusiva como yolov4\_custom\_best.weights. Ambos os arquivos podem ser utilizados para continuar o treinamento, com o comando abaixo:

```
1 !./darknet detector train data/obj.data cfg/yolov4_custom.cfg
```

```
1 /yolo/yolov4_custom_best.weights -dont_show -map |  
1 tee /yolo/results.log
```

Em muitos momentos devido a limitação de uso da GPU do *Google Colab* foi necessário reiniciar o processo de treinamento utilizando tais arquivos. O darknet ainda permite salvar cada linha de interação contendo as métricas mencionadas anteriormente em um arquivo de log para avaliações posteriores ao treinamento.

Para verificação dos resultados e definição do melhor modelo foi utilizado a métrica Mean Average Precision - mAP at 50% (sendo uma versão do mAP que considera apenas os objetos que são detectados com uma precisão de pelo menos 50%) e average loss (perda média).

### **3.4.3 Configurações da rede YOLOv4 para obter modelos com maior acurácia**

Para buscar um modelo com melhor acurácia, foi utilizada uma técnica chamada de Data Augmentation, essa técnica de processamento de dados é comumente usada em aprendizado de máquina e visão computacional para aumentar a quantidade e a variedade de dados, aplicando uma série de transformações às imagens criando dessa forma novas imagens sintéticas. Porem essas transformações devem ser realizadas de forma coerente para não afetar a integridade dos dados originais e prejudicar o treinamento.

A rede YOLOv4 permite realizar o uso dessa técnica de forma bem rápida e eficiente, basta habilitar antes de iniciar o treinamento no arquivo customizado de configuração. A figura 22 mostra como foi implementado a técnica no projeto em questão.

- *angle*: Gira a imagem em um ângulo específico.
- *saturation e exposure*: Ajusta a intensidade dos pixels para simular diferentes condições de iluminação.
- *blur*: Introduzir ruído aleatório na imagem para torná-la mais robusta.
- *hue*: Modifica as cores da imagem para simular diferentes ambientes.

Figura 22 – Habilitação de data augmentation no arquivo de configuração da rede.

```

1 -> [net]
2 batch=64
3 subdivisions=64
4 # Training
5 width=416
6 height=416
7 channels=3
8 momentum=0.949
9 decay=0.0005
10 #data augmentation
11 angle=30
12 saturation = 1.5
13 exposure = 1.5
14 mosaic=1
15 blur=1
16 hue=.1
17 #cutmix=1

```

Fonte: Elaborado pelo próprio Autor (2023)

Para a fase de identificação do melhor modelo treinado foram habilitadas e desabilitadas a augmentação que modifica a cor (*hue*) das imagens do *dataset* de forma aleatória em cada treinamento de dimensão diferente da imagem. Tais testes foram feitos pois, como comentado anteriormente, existiu a preocupação de ser prejudicial ao treinamento pois em geral as cores estão relacionadas a informações semânticas dos objetos presentes na imagem, e como existem diversos EPIs de diversas cores a informação semântica pode ser perdida, no que tange se perder informação quando a pele do colaborador se tornar colorida de forma aleatória existindo a possibilidade de interpretação pela rede se a pessoa estaria usando EPI ou não. Na figura 23 segue um exemplo de como a augmentação "HUE" altera a imagem, fazendo com que a rede possa interpretar um uso de EPI luva de diversas cores.

Foi também em alterado a dimensão da entrada da imagem na rede neural, utilizando as seguinte dimensões 416x416, 608x608 e 832x832, realizando essas alterações na linha 5 e 6 da figura 22, a alteração tem como o objetivo verificar se existe ganhos significativos no desempenho do modelo e a precisão da detecção, visto que é esperado que exista ganhos em precisão média, aumentando o tamanho da resolução da imagem

Figura 23 – Filtro de "hue" aplicado a imagem de uma mão.



Fonte: Elaborado pelo próprio Autor (2023)

de entrada, pois os detalhes dos objetos ficarão mais claros durante o processamento da rede, porém em contra partida pode acarretar o incremento do tempo de treinamento da rede.

Foi então idealizado realizar treinamentos intercalando a inclusão ou a não inclusão da augmentação "hue" com a alteração da dimensão de entrada da rede, buscando assim verificar qual modelo treinado atingiu o mAP mais elevado, para que por fim possa ser utilizado como modelo principal para o teste prático, utilizando processamento em tempo real.

#### **3.4.4 Configurações iniciais da rede YOLOv8**

Inicialmente, foi fundamental clonar o repositório da Ultralytics no GitHub, que contém os arquivos e ferramentas essenciais para treinar a YOLOv8, em seguida, é necessário importar esses arquivos para o ambiente de programação utilizando os códigos abaixo:

```

1 !git clone https://github.com/ultralytics/ultralytics.git
2 import ultralytics
3 from ultralytics import YOLO
4 from IPython import display
5 from IPython.display import display, Image
  
```

Após a instalação, foi necessário editar os arquivos de configuração da rede YOLOv8 para se adequar a demanda do projeto. Para rede, foi preciso configurar dois principais arquivos, o primeiro é o *yolov8.yaml*, no qual se deve passar a quantidade de classes que o projeto customizado irá detectar, no caso do projeto 6 classes, o mesmo vem configurado inicialmente em 80 objetos pois o mesmo foi utilizado para o treinamento na identificação dos objetos da COCO.

O arquivo original está localizado no diretório "/ultralytics/cfg/models/v8/yolov8.yaml" na pasta clonada do GitHub. Para fins de treinamento personalizado, foi necessário editá-lo conforme mostrado na Figura 24. Inicialmente, esse arquivo está configurado com 80 objetos, pois foi utilizado originalmente pelos criadores da rede YOLOv8 para treinar a identificação de objetos na base de dados COCO. Após configurado, foi preciso copiar o arquivo para o diretório raiz do ambiente de programação.

Figura 24 – Configurações aplicadas no arquivo "yolov8.yaml".

```
# Ultralytics YOLO 🚀, AGPL-3.0 license
# YOLOv8 object detection model with P3-P5 ✨

# Parameters
nc: 6 #<- Alterado de 80 para 6.
scales: # model compound scaling constants,
# [depth, width, max_channels]
n: [0.33, 0.25, 1024] # YOL0v8n summary:
s: [0.33, 0.50, 1024] # YOL0v8s summary:
m: [0.67, 0.75, 768] # YOL0v8m summary:
l: [1.00, 1.00, 512] # YOL0v8l summary:
x: [1.00, 1.25, 512] # YOL0v8x summary:
```

Fonte: Elaborado pelo próprio Autor (2023)

O segundo arquivo que requer configuração é o *model.yaml*. Este arquivo desempenha um papel crucial, pois informa à rede a localização diretório das imagens e dos arquivos com as coordenadas dos objetos para treinamento e validação. Além disso, no *model.yaml*, também é editado os nomes das classes específicas do projeto, conforme ilustrado na Figura 25.

Dando sequencia as configurações iniciais, foi preciso então copiar as imagens e arquivos ".txt" com as coordenadas dos objetos em relação as imagens para os diretórios indicados no arquivo *model.yaml*, as imagens e arquivos de texto para treinamento e validação foram copiadas do Drive para as pastas *images* e *labels* dentro do diretório

Figura 25 – Configurações aplicadas no arquivo "model.yaml".

```

path: ../datasets/ # Diretório raiz
train: images/train # Imagens de treinamento, relativo ao diretório 'path'
val: images/val # Imagens de validação, relativo ao diretório 'path'

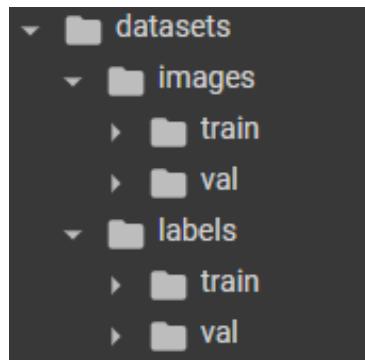
names:
  0: sem capacete
  1: com capacete
  2: sem luva
  3: com luva
  4: sem oculos
  5: com oculos

```

Fonte: Elaborado pelo próprio Autor (2023)

principal *datasets* localizado na pasta raiz do ambiente de programação conforme figura 26

Figura 26 – Diretório dos arquivos para treinamento e validação



Fonte: Elaborado pelo próprio Autor (2023)

### 3.4.5 Treinando a rede YOLOv8

Para treinar a rede YOLOv8 foi empregada também a técnica de transferência de aprendizado, como na rede YOLOv4, foi utilizado um modelo pré treinado nos moldes da rede YOLOv8 com o conjunto de dados COCO. Para baixar o modelo bastou utilizar o código abaixo:

```
1 model = YOLO('yolov8n.pt')
```

Ao contrário da rede YOLOv4, na YOLOv8, a quantidade de épocas, a inclusão ou exclusão de técnicas de aumento de dados (data augmentation) e o número de lotes (batches) podem ser editados diretamente no comando de início do treinamento. Isso torna a YOLOv8 mais flexível, pois não é necessário editar sempre o arquivo de

configuração e remontar o framework por completo quando se deseja alterar qualquer hiperparâmetro ou estratégia de aumento de dados, como era necessário no YOLOv4.

A YOLOv8 conta com CLI (*Command Line Interface*) ou interface de linha de comando que permite a execução de comandos em uma única linha de código sem a necessidade de codificar em Python. As execuções das tarefas podem ser feitas diretamente no terminal usando o comando "yolo" como se pode observar no comando de inicio de treinamento da rede abaixo:

```
1 !yolo task=detect mode=train model=yolov8m.pt data=ppe.yaml
1 epochs=150 batch=8 imgsz=640 plots=True project=/yolo/
1 name=Train640x150x8 close_mosaic=0 degrees=20.0 shear=0.2
1 scale=0.2 fliplr=0.5 mosaic=0.7 mixup=0.1
```

No comando acima, que é responsável por iniciar o treinamento, as configurações de treinamento para modelos YOLO abrangem diversos hiperparâmetros. Cada argumento listado no código desempenha uma função específica, e as configurações deles podem afetar o desempenho, a velocidade e a precisão do modelo. No caso deste projeto, os hiperparâmetros incluem:

- *task=detect*: Define que a rede YOLOv8 é configurada para identificar e localizar objetos ou regiões de interesse em uma imagem ou vídeo;
- *mode=train*: Indica que a rede está no modo de treinamento;
- *model=yolov8m.pt*: Especifica o caminho no diretório para o modelo de base convolucional usado no treinamento, que neste projeto é baseado em um modelo pré-treinado para a identificação de objetos na base de dados COCO;
- *data=ppe.yaml*: Indica o caminho no diretório do arquivo de configuração *model.yaml*, ou no caso do projeto o arquivo de configuração personalizado denominado *ppe.yaml*, que contém a nomenclatura das classes específicas a serem identificadas no projeto;
- *epochs=150*: Define o número de épocas de treinamento;

- *batch=8*: Especifica o número de imagens por lote de treinamento;
- *imgsz=640*: Define o tamanho de entrada das imagens;
- *plots=True*: Habilita a visualização da evolução do treinamento ao longo das épocas;
- *project=yolo/* e *name=Train640x150x8*: Define onde os arquivos resultantes do treinamento serão salvos.

Como complemento às configurações, como mencionado anteriormente, as técnicas de aumento de dados (Data Augmentation) podem ser editadas diretamente no comando de início de treinamento. No caso deste projeto, as técnicas de aumento de dados escolhidas foram:

- *close\_mosaic=0*: Nas épocas finais, desabilita a augmentação de mosaico;
- *degrees=20.0*: Realiza rotações aleatórias na imagem com ângulos entre +20 e -20 graus;
- *shear=0.2*: Aplica distorção de cisalhamento na imagem com uma probabilidade de 20%;
- *scale=0.2*: Aplica zoom de 20% na imagem aleatoriamente; probabilidade de 20%;
- *flip\_lr=0.5*: Realiza espelhamento horizontal na imagem com uma probabilidade de 50%;
- *mosaic=0.7*: Aplica a técnica de mosaico na imagem com uma probabilidade de 70%;
- *mixup=0.1*: Realiza mistura de imagens, inserindo uma imagem dentro de outra, com uma probabilidade de 10%.

Conforme mencionado anteriormente, o YOLOv8 oferece a capacidade de visualizar o treinamento em tempo real, o que permite acompanhar as principais métricas à medida

que o treinamento evolui. Como ilustrado na Figura 27, essa visualização representa uma parte dos resultados em tempo real do treinamento.

Figura 27 – Recorte da visualização da interação do treinamento e suas respectivas métricas.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/150	3.62G	1.649	1.625	1.532	24	640: 100% 654/654 [02:50<00:00, 3.83it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 146/146 [00:27<00:00, 5.24it/s]
	all	2328	12011	0.567	0.617	0.565 0.275

Fonte: Elaborado pelo próprio Autor (2023)

Ao final de cada época, é realizado o cálculo da precisão média utilizando o conjunto de validação definido. Após a conclusão da validação, o modelo de base convolucional é salvo na pasta do projeto. Se o Valor Médio de Precisão Média (mAP) do modelo for maior do que os anteriores, ele é salvo como "best.pt".

### 3.4.6 Configurações da rede YOLOv8 para obter modelos com maior acurácia

Para determinar o melhor modelo usando o YOLOv8, foi conduzido uma série de testes nos quais foram ajustados vários hiperparâmetros para cada treinamento. Essas alterações incluíram modificações no número de épocas, o tamanho dos lotes de imagens por época, as dimensões de entrada das imagens e a aplicação de zoom. Bem como mudança no tipo da rede YOLOv8 utilizada.

Observe que não foram feitas alterações nos hiperparâmetros relacionados à saturação, exposição e outras modificações na cor das imagens; os valores padrão da rede YOLOv8 foram mantidos. Essa decisão foi baseada em testes realizados na YOLOv4, como será destacado na próxima seção, onde a diferença no mAP ao incluir ou não essas modificações foi insignificante, variando apenas alguns pontos percentuais.

Foram realizadas as seguintes modificações nos parâmetros de entrada da rede durante os vários testes realizado da seguinte forma:

- Redimensionamento da Imagem de Entrada: As modificações na rede incluíram o redimensionamento das imagens de entrada. Isso foi feito buscando um aumento de precisão, uma vez que imagens maiores contêm mais detalhes dos objetos a serem identificados.

- Variação no Número de Lotes por Época: Outra modificação realizada envolveu a variação no número de lotes por época, com objetivo encontrar um equilíbrio entre precisão e tempo de processamento. A intenção foi verificar se a relação da redução do tempo de processamento atrelado a perda de precisão seria aceitável.
- Modificação na Escala das Imagens: Foi realizada modificação na escala das imagens visando aprimorar a identificação de objetos pequenos na rede, visto que ajustar na escala das imagens pode ajudar a melhorar a capacidade da rede de detectar objetos menores, tornando-a mais versátil e precisa em diferentes situações.
- Utilização de Modelo de Rede YOLOv8 mais Complexo: Por fim, foi realizada a utilização de um modelo de rede YOLOv8 mais complexo. Implicando em utilizar uma arquitetura de rede neural mais sofisticada e avançada. A complexidade do modelo pode contribuir para melhorar a precisão da detecção de objetos, desde que os recursos de hardware e o tempo de treinamento sejam adequados.

Essas modificações visaram aprimorar o desempenho da rede, equilibrando precisão, eficiência e capacidade de identificação de objetos de diferentes tamanhos. Cada modificação contribuiu para a busca de um modelo de detecção de objetos mais eficaz e adaptado às necessidades específicas do projeto.

A Tabela 2 mostra as modificações realizadas durante a fase de treinamento.

Tabela 2 – Valores utilizados nos treinamentos da YOLOv8

Teste	imgsz	epoch	batch	scale	Tipo da Yolo
1º	416	150	4	0.2	YOLOv8n
2º	416	150	8	0.2	YOLOv8n
3º	608	150	4	0.2	YOLOv8n
4º	608	150	8	0.2	YOLOv8n
5º	640	150	8	0.5	YOLOv8n
6º	640	134	8	0.5	YOLOv8m

Fonte: Elaborado pelo próprio Autor (2023)

Na próxima seção, serão apresentados os resultados dos treinamentos e explicado as razões que conduziram às sequências de modificações nos parâmetros com base nos

dados obtidos em cada treinamento. Essas modificações foram implementadas com o objetivo específico de aprimorar as pontuações do mAP (Mean Average Precision).

### 3.5 IMPLEMENTAÇÃO DO ALGORITMO DE IDENTIFICAÇÃO DO EPI QUADRO A QUADRO EM UM VÍDEO

Após a seleção do modelo mais eficaz, conforme discutido na seção 4, que resultou na escolha de um modelo da YOLOv8, procedeu-se à implementação de algoritmos para avaliar a conformidade relacionadas ao uso de Equipamentos de Proteção Individual (EPI) com base em dados de entrada de imagens provenientes de webcams e câmeras IP (Câmeras de Protocolo de Internet). O objetivo era testar a capacidade da rede em identificar a conformidade em tempo real.

Para isso, o YOLOv8 oferece um modo de detecção intuitivo e versátil que simplifica a execução de inferências usando modelos treinados. Ele suporta uma ampla gama de formatos de arquivos de imagens e vídeos como entrada de dados.

No entanto, ao realizar inferências com dados provenientes de uma webcam, foi necessário utilizar uma máquina local, uma vez que o Google Colab não suporta o uso de webcam em conjunto com a rede YOLOv8. Para contornar essa limitação, os arquivos de configuração e o melhor modelo treinado foram baixados e configurados para serem usados no ambiente do Jupyter Notebook em uma máquina local. Isso permitiu a realização das inferências com sucesso.

Para utilizar o modelo no modo de predição a partir de um vídeo capturado pela webcam, foi empregado o código a seguir:

```
1 from ultralytics import YOLO
2 model = YOLO("C:/Users/Gregorio/Documents/Documentos - IFES (Mestrado)
3 /YOLOv8/best.pt") results = model.predict(source=0, save=True,
4 show=True)
```

Este código carrega o modelo treinado mais eficaz, denominado "best.pt", e especifica que a previsão deve ser feita utilizando o feed da webcam (onde *source=0* representa a webcam). Ao término da inferência, o arquivo resultante é salvo na pasta raiz do

diretório onde se encontra o melhor modelo. Além disso, um arquivo em formato de texto é criado, contendo as detecções de cada quadro de imagem.

Para realizar a previsão usando uma câmera IP, o seguinte código foi empregado, basta fornecer o endereço IP da câmera desejada como argumento para o parâmetro *source*.

```
1 from ultralytics import YOLO  
2 model = YOLO("C:/Users/Gregorio/Documents/Documentos - IFES (Mestrado)  
3 /YOLOv8/bestn.pt") source = "http://192.168.18.33:6677/index"  
4 results = model(source, stream=False, save=True, show=True)
```

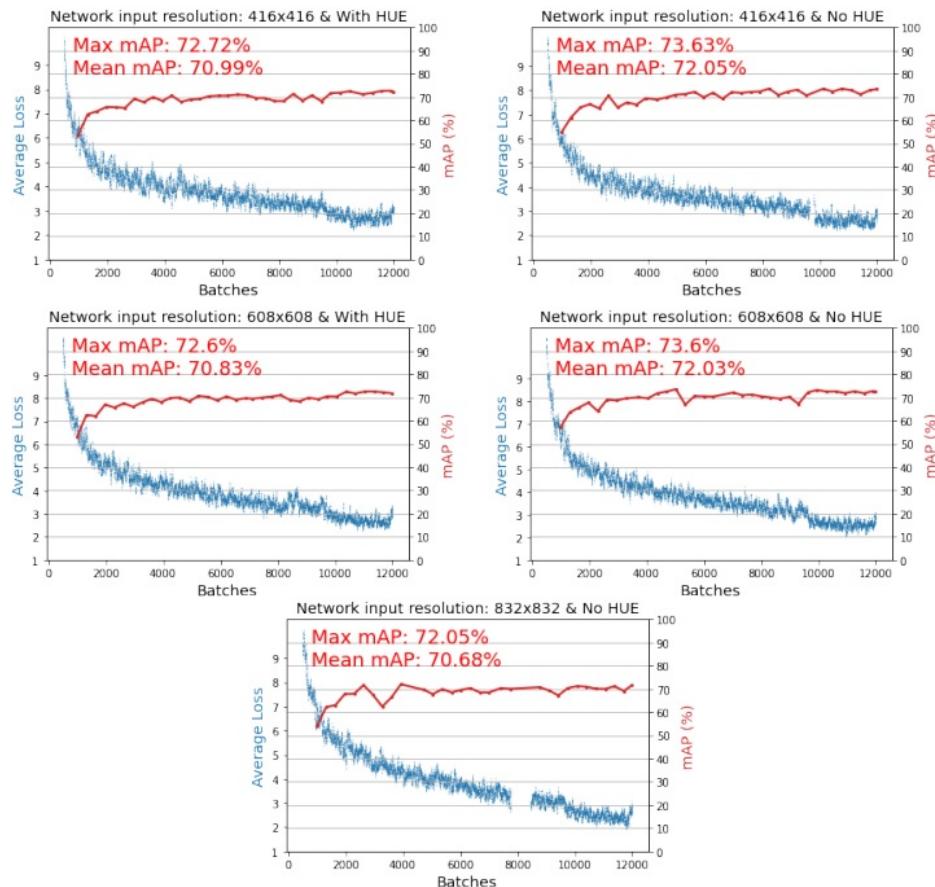
Foi Utilizado uma webcam com a configuração para gravar em 8MP, 1080p a 30fps. Foi testado também a capacidade do modelo identificar a utilização ou não dos EPIs em diferentes distancias pré definidas afim de verificar a acuracia de identificação da conformidade com a segurança de acordo com a posição que os colaboradores se encontram em relação a câmera.

## 4 RESULTADOS E DISCUSSÕES

### 4.1 RESULTADOS DA YOLOV4

Como mencionado na metodologia, para a rede YOLOv4 foram feitos diversos treinamentos diversificando o dimensionamento da entrada da imagem na rede bem como habilitando e desabilitando a aplicação de HUE. Cada treino demorou aproximadamente 40 horas para completar as 12000 interações pré-definidas no arquivo de configuração, não existindo diferença significativa no tempo de treinamento mesmo alterando a dimensão da entrada da imagem. Em todos os treinamentos feitos foi observando *overfitting* a partir de um determinado ponto, foi decidido aguardar o término do treinamento para uma visualização mais precisa do comportamento da rede nos diversos experimentos realizados para posteriores comparações. A figura 28 apresenta os resultados dos melhores modelos treinados.

Figura 28 – Melhores modelos apresentados em gráfico de mAP@50% e Average Loss versus interações.



Fonte: Elaborado pelo próprio Autor (2023)

Como se pode observar nos gráficos da figura 28, apesar das mudanças de parâmetros em todos os treinamentos foram obtidos precisões médias muito próximas, se estabilizando em aproximadamente 70% após 8000 interações. E a perda média seguindo praticamente a mesma tendência, se estabilizando no final do treinamento em aproximadamente 2,8.

Apesar dos resultados muitos próximos, foi possível definir um melhor modelo, calculando a média de todos os mAP em cada modelo. O modelo com a maior pontuação média foi o modelo com resolução de entrada de 416x416 sem ampliação de HUE.

A Tabela 3 mostra como este modelo classificou cada classe durante as interações avaliadas pela métrica de precisão.

Tabela 3 – Precisão de cada classe nas interações correspondentes.

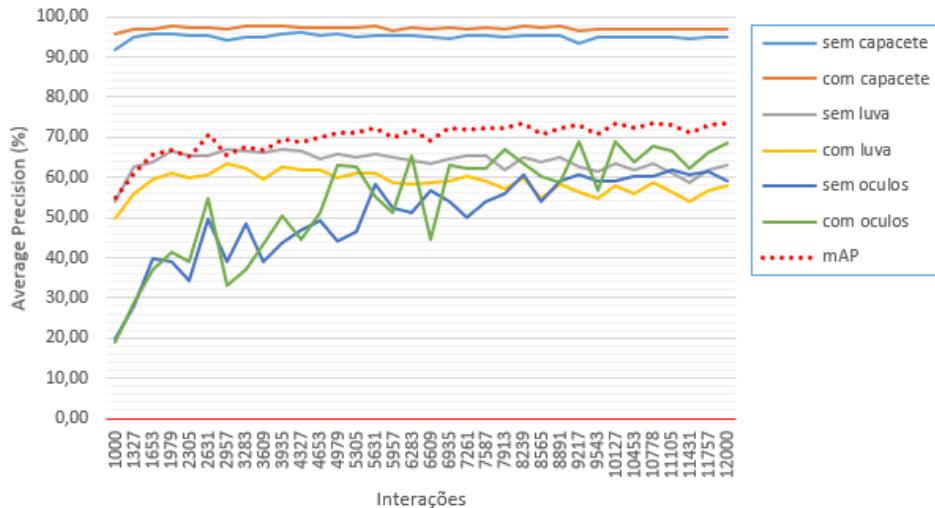
Anotação	1000	4000	8000	12000
sem capacete	91.65%	95.55%	95.24%	94.86%
com capacete	95.58%	97.79%	97.66%	96.91%
sem luva	54.17%	66.82%	65.05%	63.03%
com luva	49.66%	62.76%	59.75%	57.97%
sem oculos	19.60%	43.64%	60.62%	59.20%
com oculos	19.02%	50.48%	63.48%	68.52%
mAP	54.95%	69.51%	73.63%	73.41%

Fonte: Elaborado pelo próprio Autor (2023)

O figura 29 demonstra graficamente a evolução do aprendizado da rede cada classe. Evidenciando que a rede conseguiu aprender e reter informações no decorrer das interações de forma correta.

Como pode ser observado na Tabela 3 e no gráfico 29, a melhor base convolucional da YOLOv4 foi obtida quando o treinamento completou entorno de 8000 interações atingindo um mAP de 73,63% com cerca de 26,66 horas de tempo de treinamento da rede, superando os outros modelos ligeiramente.

Figura 29 – Gráfico da evolução da rede discretizado por classes.



Fonte: Elaborado pelo próprio Autor (2023)

## 4.2 RESULTADOS DA YOLOV8

Como mencionado na metodologia, foram realizados um total de 6 testes principais para avaliar o melhor modelo gerado pela YOLOv8. Cinco dos testes utilizaram a versão YOLOv8 nano e o sexto utilizou a YOLOv8 medium. A tabela abaixo apresenta o tempo de processamento necessário para concluir todas as épocas em cada treinamento.

Tabela 4 – Tempo de processamento da rede YOLOv8 em cada teste.

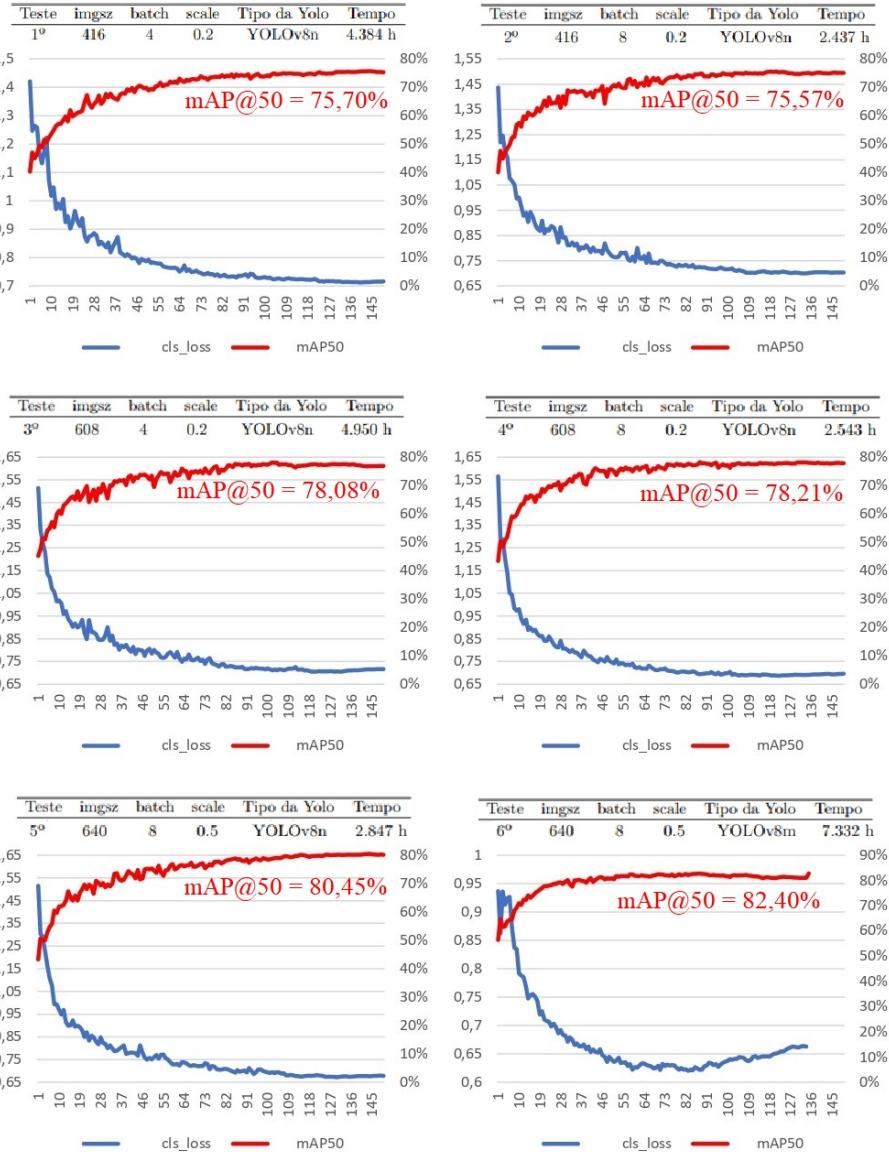
Teste	imgsz	batch	scale	Tipo da Yolo	Tempo
1º	416	4	0.2	YOLOv8n	4.384 h
2º	416	8	0.2	YOLOv8n	2.437 h
3º	608	4	0.2	YOLOv8n	4.950 h
4º	608	8	0.2	YOLOv8n	2.543 h
5º	640	8	0.5	YOLOv8n	2.847 h
6º	640	8	0.5	YOLOv8m	7.332 h

Fonte: Elaborado pelo próprio Autor (2023)

Os graficos na figura 30 abaixo demonstram o desempenho do treinamento da rede em cada cada teste.

Como podemos observar nos testes 1 e 2, e conforme detalhado na Tabela 4, ao manter os valores de entrada da imagem constantes e variando apenas o número de lotes (batch), conseguimos reduzir o tempo de processamento em aproximadamente 2

Figura 30 – Treinamentos realizados durante os testes com a rede YOLOv8.



Fonte: Elaborado pelo próprio Autor (2023)

horas, com uma pequena perda na precisão média, como pode ser visto na tabela 5.

Nos testes 3 e 4, observamos o mesmo padrão identificado nos testes anteriores com a redução do tempo de processamento em aproximadamente 2,4 horas. No entanto, a alteração no tamanho da imagem de entrada resultou em ganhos na precisão média de 2,64% quando comparado aos testes anteriores, o que era esperado devido ao aumento na dimensão de entrada da imagem e entre os dois foi contraria uma pequena diferença de mAP.

No teste 5, foi decidido alterar a argumentação *scale*, aumentando a escala de 20% para

50% para as imagens geradas na técnica de aumento de dados (*Data Augmentation*). Essa decisão foi motivada pelo fato de que, na estratificação da precisão de cada anotação nos testes de 1 a 4, as anotações "sem luva" e "com luva" apresentaram uma proximidade de 60%. Foi idealizado que um aumento na escala da imagem poderia levar a uma melhor identificação do uso ou não de Equipamento de Proteção Individual (EPI) que são menos evidentes. Com a modificação foi possível conseguir aproximadamente 2,2% a mais de mAP. Atingindo um valor de 80%.

Para o teste 6, foi decidido utilizar os mesmos padrões do teste 5, visto ao ótimo desempenho, porém alterando o tipo da rede YOLOv8 de *nano* para *medium*, com essa alteração, o tempo de processamento aumentou, mas em contra partida foi obtido ganhos no mAP. Durante o teste, a rede finalizou o treinamento em 131 épocas desprendendo 7,332 horas, pois a mesma identificou que depois de 50 épocas não existiu ganho significativo na precisão média.

Na tabela 5 é demonstrado os valores máximos de mAP de cada treinamento:

Tabela 5 – Valores máximos de mAP obtidos em cada teste

Teste	mAP	Diferença
1º	75,70%	-
2º	75,57%	-0,27%
3º	78,08%	+2,64%
4º	78,21%	+0,19%
5º	80,45%	+2,21%
6º	82,40%	+2,05%

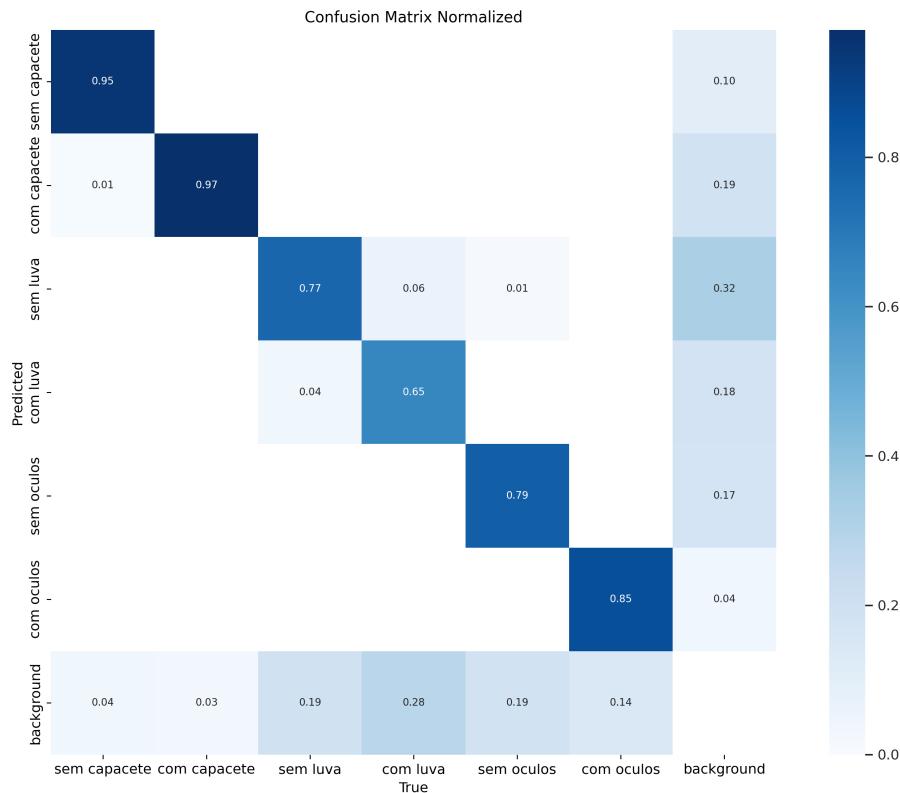
Fonte: Elaborado pelo próprio Autor (2023)

Com base nos dados coletados e após a conclusão de todos os testes mencionados, foi possível determinar o melhor modelo treinado utilizando a YOLOv8. Esse modelo foi obtido no sexto treinamento, e alcançou um mAP de 82,40% com um tempo de processamento de 7.332 horas.

A figura 31 representa a matriz de confusão do melhor modelo da rede YOLOv8. Fica evidente que em sua maioria dos falsos negativos foram e falsos positivos das anotações "sem luva", "com luva", "sem óculos" e "com óculos" foram identificados como

elementos de fundo (*background*) da imagem.

Figura 31 – Matriz de confusão do melhor modelo da rede YOLOv8



Fonte: Elaborado pelo próprio Autor (2023)

Na Tabela 6 na seção seguinte podem ser verificados as precisões de cada classe no conjunto de validação utilizando como base convulacional o melhor modelo treinado.

#### 4.3 COMPARAÇÃO DOS RESULTADOS DAS REDES YOLOV4 E YOLOV8

Ao analisar os resultados dos treinamentos em ambas as redes, torna-se claro que a rede YOLOv8 alcançou um desempenho significativamente superior em comparação com a YOLOv4. O pior modelo na rede YOLOv8 apresentou uma precisão maior do que o melhor modelo na rede YOLOv4. Além disso, é notável que o tempo de processamento na YOLOv8 foi consideravelmente menor em comparação com a YOLOv4, caindo de 40 horas para 7.332 horas, considerando o melhor modelo da rede YOLOv8.

Essa comparação ressalta a eficácia da YOLOv8 em termos de precisão e eficiência de processamento, tornando-a uma escolha clara em aplicações de detecção de objetos.

A tabela 6 demonstra validação dos dados com comparação das precisões de cada classe alcançadas utilizando os melhores modelos treinados de cada rede.

Tabela 6 – Precisão de cada classe nos melhores modelos de ambas redes

Anotação	YOLOv4	YOLOv8
Tempo de processamento	40h	7.332h
sem capacete	95.24%	96.50%
com capacete	97.66%	98.10%
sem luva	65.05%	70.60%
com luva	59.75%	64.10%
sem óculos	60.62%	79.40%
com óculos	63.48%	85.50%
mAP	73.63%	82.4%

Fonte: Elaborado pelo próprio Autor (2023)

Com a rede YOLOv8, observamos melhorias na precisão de todas as classes, resultando em um aumento no mAP em comparação com a YOLOv4, com um incremento notável de 8,77%.

Destaca-se que a rede YOLOv8 demonstrou uma eficiência significativamente maior na identificação de anotações relacionadas à presença ou ausência de óculos, superando sua predecessora em 18,78% para anotações sem óculos e em 22,02% para anotações com óculos. Esse resultado evidencia a capacidade aprimorada da YOLOv8 na detecção de objetos menores em comparação com a rede YOLOv4, devido às modificações em sua arquitetura.

A YOLOv8 emprega duas funções de perda fundamentais: CloU (Interseção Completa sobre União) e DFL (Focalização de Distribuição). A CloU opera de maneira semelhante ao tradicional Índice de Interseção sobre União (IoU), porém leva em conta não apenas a sobreposição das caixas delimitadoras, mas também a precisão da localização entre elas. Por outro lado, a DFL utiliza a técnica de focalização para atribuir mais peso às observações difíceis durante o treinamento do modelo, ajudando a lidar com o desbalanceamento de classes e aprimorando o desempenho em tarefas de detecção de objetos, especialmente quando se trata de classes raras ou difíceis de detectar, como é o caso das anotações "sem óculos" e "com óculos" (TERVEN; CÓRDOVA-ESPARZA;

ROMERO-GONZÁLEZ, 2023).

Dessa forma, ambas as funções de perda penalizam mais fortemente a detecção incorreta de objetos pequenos, incentivando a YOLOv8 a ser mais precisa na identificação de objetos, especialmente em situações que envolvem objetos de tamanho reduzido.

Esses ganhos de precisão refletem a eficácia do modelo YOLOv8 em lidar com a tarefa de detecção da conformidade do uso ou não uso do EPI, tornando-o uma escolha superior em relação ao desempenho geral quando comparado à YOLOv4. Portanto o modelo foi escolhido para ser utilizado nos testes em imagens avulsas, vídeos e vídeos de câmera em tempo real, visto que ela traria resultados mais precisos devido ao seu elevado mAP.

#### 4.4 RESULTADOS DO MELHOR MODELO APLICADO EM IMAGENS

As figuras 32, 33 e 34 a seguir mostram exemplos de como o melhor modelo de rede neural conseguiu classificar e segmentar corretamente as pessoas que estavam conformes ou não em relação ao uso do EPI.

Figura 32 – Resultado do melhor modelo em imagens avulças.



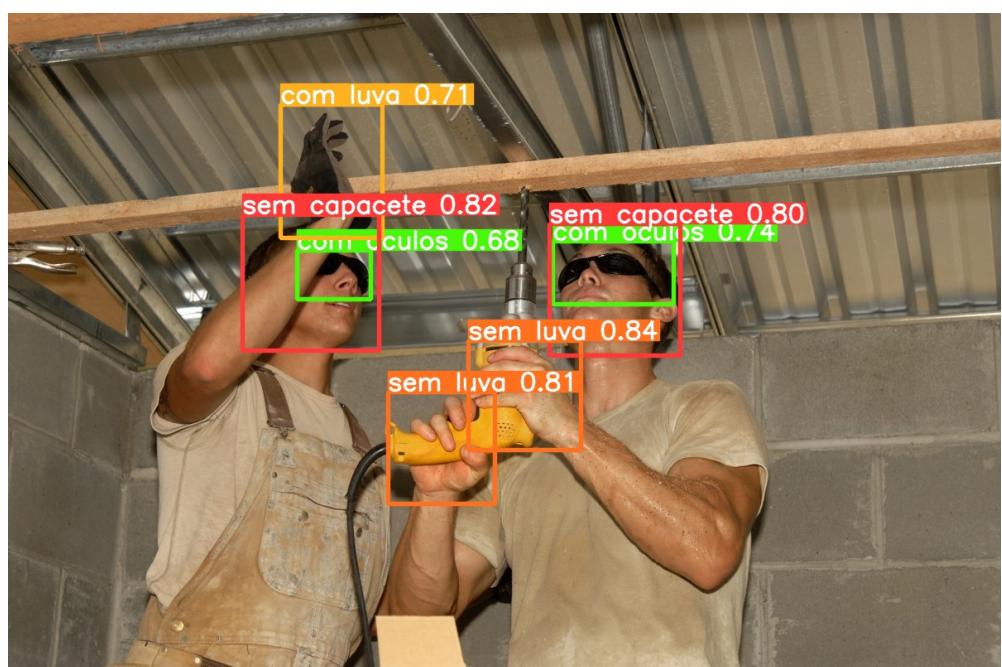
Fonte: Elaborado pelo próprio Autor (2023)

Figura 33 – Resultado do melhor modelo em imagens avulças.



Fonte: Elaborado pelo próprio Autor (2023)

Figura 34 – Resultado do melhor modelo em imagens avulças.

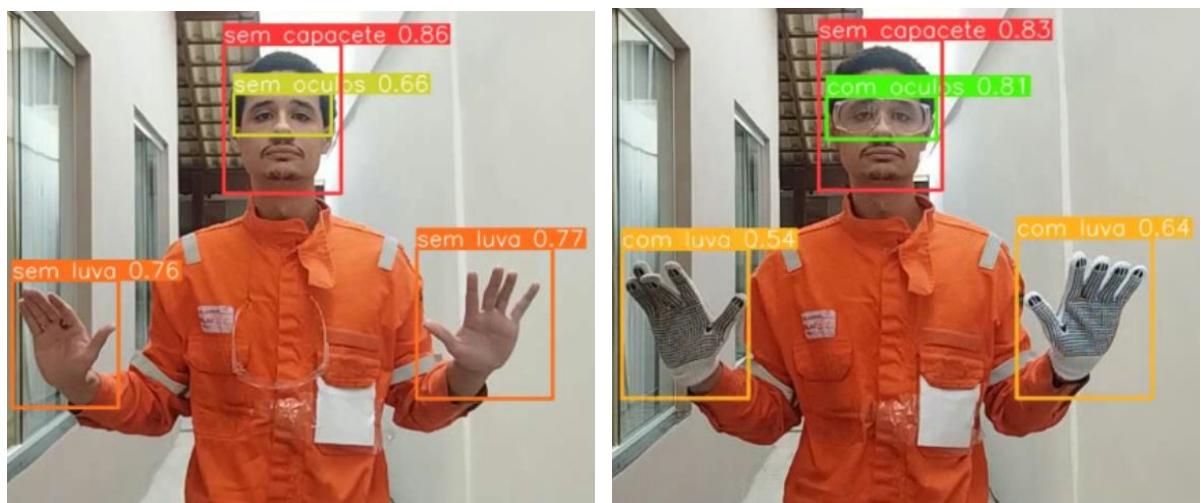


Fonte: Elaborado pelo próprio Autor (2023)

#### 4.5 RESULTADOS DO MELHOR MODELO APLICADO EM VIDEOS (WEBCAM E CAMERAS IP)

A figura a seguir apresenta exemplos de quadros retirados da rede rodando em uma gravação em tempo real com webcam, demonstrando o processamento preciso e em tempo real pela rede YOLOv8 a uma taxa de 30 quadros por segundo (fps). A rede YOLOv8 conseguiu identificar e segmentar corretamente as imagens capturadas pela webcam com alta precisão com uma distância da camera de 1 a 4 metros.

Figura 35 – Resultado da inferencia da rede rodando tempo real de 1 a 2 metros.



Fonte: Elaborado pelo próprio Autor (2023)

Figura 36 – Resultado da inferencia da rede rodando tempo real de 2 a 4 metros.



Fonte: Elaborado pelo próprio Autor (2023)

Embora a detecção tenha sido geralmente precisa durante a identificação em tempo

real, durante os testes com a melhor rede ao rodar vídeos gerados pela webcam, observou-se a ocorrência de falhas na detecção, incluindo detecções ausentes e detecções falsas. Muitas vezes, essas falhas estavam relacionadas à distância entre o objeto-alvo e a câmera quando afastados a mais de 4 metros, como ilustrado na Figura 37.

Figura 37 – Falsos negativos e não detecção durante predição com webcam em tempo real quando afastado acima de 4 metros.



Fonte: Elaborado pelo próprio Autor (2023)

## 5 CONSIDERAÇÕES FINAIS

Concluiu-se que ambas as redes neurais produziram respostas conforme o esperado, conseguindo classificar e segmentar corretamente a conformidade ou não conformidade do uso de EPI. Sendo que o melhor modelo da rede YOLOv8 superou todas as outras obtendo uma alta precisão média de 82,40%.

Neste trabalho, percebeu-se que a YOLOv8 é uma rede realmente poderosa, como pode ser analisado pela tabela 6 a precisão das classes de anotação "sem capacete" e "com capacete" foi superior a 95%, o que significa que a rede pode aprender corretamente com um grande volume de dados, mas também em contra partida a rede conseguiu obter resultados de forma satisfatória ao identificar as classes "sem óculos" e "com óculos", que receberam muito menos dados do que a anotação de capacete, além de serem mais dificeis de identificar devido ao tamanho diminuto, chegando superar 79%.

Como mostrado na seção anterior, devido à sua alta velocidade de processamento, é totalmente viável usar a rede neural baseada em YOLOv8 para a detecção de EPI em tempo real, vinculando a rede a uma câmera. Para as próximas etapas, será criado um sistema no qual a rede neural YOLOv8 treinada nesse projeto analisará vídeos de um circuito de TV de vigilância instalado na área de trabalho com um grande fluxo de pessoas indo e vindo dos locais de trabalho, que, ao identificar o não cumprimento do funcionário quanto ao uso de EPI, o sistema gerará um alerta para a equipe de SMS tomar as medidas apropriadas, além de gerar também logs do número de pessoas que estão em conformidade ou não com a segurança, gerando assim estatísticas que podem ser usadas como indicadores de confiabilidade que auxiliarão na tomada de decisão e na gestão de segurança dentro da empresa.

Também serão coletadas mais imagens, para serem anotados pessoas não utilizando e utilizando luvas, utilizando e não utilizando óculos de proteção, a fim de melhorar a precisão da identificação da conformidade de segurança do uso ou não uso desses EPIs tão importantes para a integridade do trabalhador.

Parte do trabalho com o título *Development Of A Real-Time Monitoring System For Detect The Use Of Personal Protective Equipment (PPE) From Machine Learning*

foi publicado no XLIII congresso *Ibero-Latin-American Congress on Computational Methods in Engineering* em Foz do Iguaçu no Paraná, Brasil em 25 de novembro de 2022.

## REFERÊNCIAS

- AHMED, M. I. B. *et al.* Personal protective equipment detection: A deep-learning-based sustainable approach. **Sustainability**, v. 15, n. 18, 2023. ISSN 2071-1050. Disponível em: <<https://www.mdpi.com/2071-1050/15/18/13990>>.
- ALBUMENTATIONS. **Bounding boxes augmentation for object detection**. 2019. [Online; july 29; 2022]. Disponível em: <[https://albumentations.ai/docs/getting\\_started/bounding\\_boxes\\_augmentation/](https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/)>.
- ALMEIDA, I. D. Importância da utilização dos Óculos de proteção na construção civil. **Universidade de Taubaté. Taubaté.**, p. 32, 2019.
- BOCHKOVSKIY, A.; WANG, C.; LIAO, H. M. Yolov4: Optimal speed and accuracy of object detection. **CoRR**, abs/2004.10934, 2020. Disponível em: <<https://arxiv.org/abs/2004.10934>>.
- BRASIL. Ministério do Trabalho. NR 06 - EQUIPAMENTO DE PROTEÇÃO INDIVIDUAL - EPI, Brasília: Ministério do Trabalho.** 2018. [Online; Acessado Março 28, 2022]. Disponível em: <<https://www.gov.br/trabalho-e-previdencia/pt-br/composicao/orgaos-especificos/secretaria-de-trabalho/inspecao/seguranca-e-saude-no-trabalho/normas-regulamentadoras/nr-06.pdf>>.
- CABREJOS, J. A. L.; ROMAN-GONZALEZ, A. Artificial intelligence system for detecting the use of personal protective equipment. **International Journal of Advanced Computer Science and Applications**, v. 14, n. 5, 2023.
- CHEN, S.; DEMACHI, K. Vision-based approach for ensuring proper use of personal protective equipment (ppe) in decommissioning of fukushima daiichi nuclear power station. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 10, n. 15, p. 5129, 2020.
- EBERMAM, R. A. K. E. Uma introdução comprehensiva às redes neurais convolucionais: Um estudo de caso para reconhecimento de caracteres alfabeticos. **Revista de Sistemas de Informação da FSMA**, Universidade Federal do Espírito Santo, Vitória - ES, n. 22, p. 49–59, 2018.
- FANG, W. *et al.* A deep learning-based approach for mitigating falls from height with computer vision: Convolutional neural network. **Advanced Engineering Informatics**, Elsevier, v. 39, p. 170–177, 2019.
- FIORAVANTE, E.; BARONI, F. Causas da resistência ao uso do equipamento de proteção individual (epi). p. 9, 2012.
- GONZAGA, L. M.; ALMEIDA, G. M. de. Artificial intelligence usage for identifying automotive products. p. 7, 2020.
- GOUVÉA, S. A. A. Importância do uso correto do capacete de segurança na construção civil. **Universidade de Taubaté. Taubaté.**, p. 32, 2018.
- HASSAN YASSER KHALIL, I. A. E. Learning feature fusion in deep learning-based object detector. **Journal of Engineering**, Hindawi, p. 11, 2020.

- HUSSAIN, M. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. **Machines**, v. 11, n. 7, 2023. ISSN 2075-1702. Disponível em: <<https://www.mdpi.com/2075-1702/11/7/677>>.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.I.: s.n.], 2012. p. 1097–1105.
- LECUN, Y. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Taipei, Taiwan, v. 86, n. 11, p. 2278–2324, 1998.
- LEE, Y.-R. *et al.* Deep learning-based framework for monitoring wearing personal protective equipment on construction sites. **Journal of Computational Design and Engineering**, v. 10, n. 2, p. 905–917, 03 2023. ISSN 2288-5048. Disponível em: <<https://doi.org/10.1093/jcde/qwad019>>.
- LI, Y. *et al.* Deep learning-based safety helmet detection in engineering management based on convolutional neural networks. **Advances in Civil Engineering**, Hindawi, v. 2020, n. 1-10, 2020.
- LIMA, M. V. V. A. Importância do uso correto de luvas de proteção em viveiros florestais. **Universidade de Taubaté. Taubaté.**, p. 25, 2019.
- LO, J.-H.; LIN, L.-K.; HUNG, C.-C. Real-time personal protective equipment compliance detection based on deep learning algorithm. **Sustainability**, v. 15, n. 1, 2023. ISSN 2071-1050. Disponível em: <<https://www.mdpi.com/2071-1050/15/1/391>>.
- NASCIMENTO, I. G. do *et al.* Segurança no trabalho: Motivos que levam o trabalhador da construção civil a deixar de utilizar os epis. **XI Congresso Nacional De Excelência Em Gestão.**, p. 21, 2015.
- NATHA, N. D.; BEHZADANB, A. H.; PAALA, S. G. Deep learning for site safety: Real-time detection of personal protective equipment. **Automation in Construction**, Elsevier, v. 112, p. 103085, 2020.
- OCUPACIONAL. **Medicina e engenharia de segurança do trabalho. Consequências do descumprimento das Normas Regulamentadoras**. 2018. [Online; Acessado Maio 28, 2022]. Disponível em: <<https://www.ocupacional.com.br/ocupacional/consequencias-do-descumprimento-dasnormas-regulamentadoras/>>.
- PADILLA, R.; NETTO, S. L.; SILVA, E. A. B. da. A survey on performance metrics for object-detection algorithms. In: **2020 International Conference on Systems, Signals and Image Processing (IWSSIP)**. [S.I.: s.n.], 2020. p. 237–242.
- PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: **International conference on machine learning**. [S.I.: s.n.], 2013. p. 1310–1318.
- REDMON, J. e. a. **You Only Look Once: Unified, Real-Time Object Detection**. 2016. [Online; accessed july 03, 2022]. Disponível em: <<http://arxiv.org/abs/1506.02640>>.
- REIS, D. *et al.* **Real-Time Flying Object Detection with YOLOv8**. 2023.

ROBOFLOW. **Public Dataset - Hard Hat Workers.** 2018. [Online; Acessado Maio 25, 2021]. Disponível em: <<https://public.roboflow.com/object-detection/hard-hat-workers/2/images/90d4f039570e8ddf467cd6ab24aa4883>>.

SAHA, S. **A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.** 2018. [Online; accessed September 12, 2022]. Disponível em: <<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>>.

SINHA, R. K.; PANDEY, R.; PATTNAIK, R. Deep learning for computer vision tasks: a review. **arXiv preprint arXiv:1804.03928**, 2018.

TERVEN, J.; CORDOVA-ESPARZA, D. **A Comprehensive Review of YOLO: From YOLOv1 and Beyond.** 2023.

TERVEN, J.; CÓRDOVA-ESPARZA, D.-M.; ROMERO-GONZÁLEZ, J.-A. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. **Machine Learning and Knowledge Extraction**, MDPI AG, v. 5, n. 4, p. 1680–1716, nov. 2023. ISSN 2504-4990. Disponível em: <<http://dx.doi.org/10.3390/make5040083>>.

TOSMANN, J. M. **REVISTA CIPA, Cipa e Incêndio. Importância da fiscalização do uso de EPIs e EPCs.** 2019. [Online; Acessado Março 28, 2022]. Disponível em: <[shorturl.at/sxCl6](http://shorturl.at/sxCl6)>.

TRENTIN, M. L. A importância do epi e do certificado de aprovação (ca) para o servidor público estatutário. **Universidade do Vale do Rio dos Sinos. São Leopoldo**, p. 16, 2016.

TZUTALIN. **LabelImg.** 2017. [Online; july 29; 2022]. Disponível em: <<https://github.com/heartexlabs/labelImg>>.

VARGAS, A. C. G.; CARVALHO, A. M. P.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. **Universidade Federal Fluminense. Niterói.**, p. 4, 2019.

VIANA, M. R. Estatística de acidentes de trabalho em uma empresa de coleta de resíduos domiciliares e industriais – estudo de caso. **Universidade Tecnológica Federal Do Paraná**, p. 65, 2014.

VO, A. **Deep Learning – Computer Vision and Convolutional Neural Networks.** 2018. [Online; accessed September 12, 2022]. Disponível em: <<https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/>>.