

**UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE
MESQUITA FILHO" - UNESP**

Faculdade de Engenharia de Ilha Solteira - campus de Ilha Solteira

PEDRO HENRIQUE FUGITA BÓIS

**Monitoramento de segurança em ambiente industrial por meio de visão computacional
baseada em redes neurais**

Ilha Solteira

2026

Pedro Henrique Fugita Bóis

**Monitoramento de segurança em ambiente industrial por meio de visão computacional
baseada em redes neurais**

Trabalho de Graduação apresentado à Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), Faculdade de Engenharia de Ilha Solteira, Ilha Solteira, para obtenção do título de Bacharel em Engenharia Mecânica.

Orientador: Prof.^a Dr.^a Christiane Marie Schweitzer

Ilha Solteira
2026

FICHA CATALOGRÁFICA

Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

B682m Bóis, Pedro Henrique Fugita.
Monitoramento de segurança em ambiente industrial por meio de visão computacional baseada em redes neurais / Pedro Henrique Fugita Bóis. -- Ilha Solteira: [s.n.], 2026
48 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Mecânica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2026

Orientador: Christiane Marie Schweitzer

Inclui bibliografia

1. Visão computacional. 2. Aprendizado profundo. 3. Detecção de objetos. 4. YOLO. 5. Segurança do trabalho.

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA – CÂMPUS DE ILHA SOLTEIRA

CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

ATA DA DEFESA – TRABALHO DE GRADUAÇÃO

TÍTULO: *Monitoramento de segurança em ambiente industrial por meio de visão computacional baseada em redes neurais*

ALUNO: Pedro Henrique Fugita Bóis **RA:** 172055211

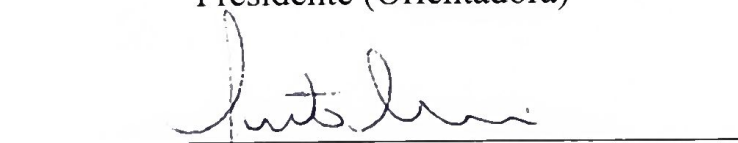
Orientador: Profa. Dra. Christiane Marie Schweitzer

Aprovado (x) – Reprovado () pela Comissão Examinadora

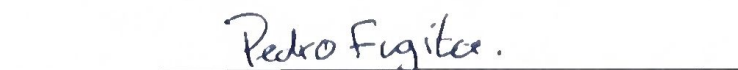
Nota obtida: 10 (Dez)

Comissão Examinadora:


Profa. Dra. Christiane Marie Schweitzer
Presidente (Orientadora)


Prof. Dr. Antonio Marcos Cossi
Membro da banca


Profa. Dra. Camila Gianini Gonsalez Bueno
Membro da banca


Pedro Henrique Fugita Bóis
Discente

Ilha Solteira (SP) 08 de junho de 2026.

AGRADECIMENTOS

Meus agradecimentos a todos os familiares, amigos, professores e funcionários da FEIS-UNESP, que direta ou indiretamente contribuíram para a realização deste trabalho. Em especial, dedico meus agradecimentos:

A Deus, por ter me dado saúde e fé para chegar até aqui.

Aos meus pais, Márcia Rosana e José Roberto, e à minha irmã, Mariana, pelo amor, apoio e incentivo.

À minha namorada, Ana Flávia, pelo amor, apoio, conforto, confiança e incentivo em todos os momentos.

A cada integrante de minha república pelo acolhimento e companheirismo nos momentos difíceis e por ter sido minha segunda casa ao longo dessa jornada.

À Prof.^a Dr.^a Christiane Marie Schweitzer, pela oportunidade e por todo ensinamento, confiança e orientação.

*“A estrada do sucesso é uma estrada longa.
É uma escada íngreme, às vezes com degraus
muito altos, e a gente vai ter que se esforçar
muito para subir esses degraus. A desistência
é um atributo dos fracos. Os fortes não desistem,
eles vão até o fim para buscar o que desejam.”*
– Ozires Silva

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema automatizado para o monitoramento do uso de Equipamentos de Proteção Individual (EPI) em ambientes industriais, empregando técnicas de visão computacional baseadas em redes neurais convolucionais. O objetivo consiste em aumentar a eficácia da fiscalização de segurança e garantir a conformidade com as normas regulamentadoras por meio de um sistema de detecção em tempo real de baixo custo computacional. A metodologia adota a arquitetura YOLOv8n, treinada a partir do conjunto de dados SH17, que engloba diversas classes de equipamentos e contextos operacionais de manufatura. O treinamento foi realizado em ambiente de nuvem com auxílio de processamento gráfico de alto desempenho, aplicando técnicas de aprendizado por transferência e aumento de dados para otimizar a generalização do modelo. O modelo treinado atingiu um mAP@0.5 de 32,9%, revocação de 48,0% e precisão de 57,2%, indicando desempenho aceitável para detecção de EPIs em tempo real em cenários controlados, quando comparado com resultados da literatura. A implementação final resultou em um software com interface gráfica capaz de processar fontes de vídeo diversificadas, emitindo alertas visuais imediatos em caso de não conformidades com latência média de 1800 ms. Conclui-se que a automação do monitoramento reduz a dependência de inspeção humana, mitiga riscos de acidentes e fornece evidências digitais para a gestão de segurança do trabalho, apresentando-se como uma solução escalável e de baixo custo para o cenário industrial moderno.

Palavras-chave: visão computacional; aprendizado profundo; detecção de objetos; YOLO; segurança do trabalho.

ABSTRACT

This work presents the development of an automated system for monitoring the use of Personal Protective Equipment (PPE) in industrial environments, employing computer vision techniques based on convolutional neural networks. The objective is to enhance the effectiveness of safety inspection and ensure compliance with regulatory standards through a low computational cost real-time detection system. The methodology adopts the YOLOv8n architecture, trained using the SH17 dataset, which includes several equipment classes and operational manufacturing contexts. The training process was carried out in a cloud environment with the support of high-performance graphical processing, applying transfer learning and data augmentation techniques to optimize the model's generalization capability. The trained model achieved a mAP@0.5 of 32.9%, recall of 48.0%, and precision of 57.2%, indicating acceptable performance for real-time PPE detection in controlled scenarios, when compared to the literature. The final implementation resulted in software with a graphical interface capable of processing diverse video sources, issuing immediate visual alerts in cases of non-compliance, with an average latency of 1800 ms. It is concluded that monitoring automation reduces dependence on human inspection, mitigates accident risks, and provides digital evidence for occupational safety management, presenting itself as a scalable and low-cost solution for the modern industrial sector.

Keywords: computer vision; deep learning; object detection; YOLO; occupational safety.

LISTA DE FIGURAS

1	Fluxograma simplificado de uma rede neural do tipo R-CNN.	18
2	Fluxograma simplificado de uma rede neural do tipo YOLO.	19
3	Fluxograma simplificado de treinamento do modelo YOLO.	21
4	Distribuição de todas as instâncias de classes.	26
5	Exemplos de imagens do <i>dataset</i>	27
6	Exemplo de arquivo de anotação do <i>dataset</i>	27
7	Organização dos diretórios do <i>dataset</i>	28
8	Código para integração de ambientes e descompactação do <i>dataset</i>	29
9	Código para criação do arquivo no formato .yaml.	30
10	Código para carregamento do modelo YOLO.	32
11	Código para validação do treinamento realizado.	33
12	Fluxograma do software de aplicação do modelo treinado.	34
13	Métricas de desempenho do modelo treinado.	36
14	Curvas Precisão-Revocação do modelo treinado.	37
15	Matriz de confusão do modelo treinado.	38
16	Curva F1Score x Confiança do modelo treinado.	39
17	Amostras dos resultados obtidos pelo modelo treinado.	40
18	Interface do aplicativo desenvolvido em funcionamento (Cenário 1). . . .	41
19	Interface do aplicativo desenvolvido em funcionamento (Cenário 2). . . .	41

LISTA DE TABELAS

Tabela 1	–	Classes presentes no <i>dataset</i>	25
Tabela 2	–	Parâmetros de configuração de treinamento do modelo YOLOv8n.	31
Tabela 3	–	Parâmetros de configuração de validação do modelo treinado.	33

LISTA DE ABREVIATURAS E SIGLAS

CNN Convolutional Neural Network.

EPI Equipamento de Proteção Individual.

ESG Environmental, Social and Governance.

FPS Frames Per Second.

Geons Íons Geométricos.

GPU Graphics Processing Unit.

IoU Intersection over Union.

NR-06 Norma Regulamentadora nº 06.

R-CNN Region-based Convolutional Neural Network.

RGB Red, Green, Blue.

YOLO You Only Look Once.

SUMÁRIO

1	INTRODUÇÃO	11
1.1	Contexto	11
1.2	Justificativas	11
1.2.1	Justificativas Técnicas	11
1.2.2	Justificativas Sociais	12
1.3	Objetivos	12
1.4	Estrutura do Trabalho	12
2	REVISÃO DA LITERATURA	14
2.1	Visão Humana	14
2.2	Visão Computacional	14
2.2.1	Tensores	15
2.2.2	Neurônios Artificiais	15
2.2.3	Redes Neurais Convolucionais	17
2.2.4	Redes Neurais Convolucionais por Região	17
2.2.5	Você Só Olha Uma Vez	18
2.3	Treinamento de Redes Neurais	19
2.4	Validação de Modelo de Rede Neural	21
2.5	Inferência de Modelo de Rede Neural	23
2.5.1	Vídeos Gravados	23
2.5.2	Câmeras de Vídeo	23
2.5.3	Captura de Tela	23
3	DESENVOLVIMENTO	25
3.1	Base de Dados (Dataset)	25
3.2	Ambiente Computacional	28
3.3	Configuração de Treinamento	29
3.4	Configuração de Validação do Modelo Treinado	32
3.5	Inferência de Modelo Treinado	33
4	RESULTADOS E DISCUSSÕES	35
4.1	Análise da Base de Dados	35
4.2	Análise das Curvas de Treinamento e Validação	35
4.3	Inferência por meio de Desenvolvimento de Software	41
5	CONSIDERAÇÕES FINAIS	43

REFERÊNCIAS 45

1 INTRODUÇÃO

1.1 Contexto

Dados da ORGANIZAÇÃO INTERNACIONAL DO TRABALHO (2019) comprovam que ambientes industriais estão entre os mais suscetíveis a acidentes de trabalho devido à presença de máquinas pesadas, rotativas, operações em altura, agentes químicos, ruídos e demais riscos ocupacionais. O uso correto de Equipamento de Proteção Individual (EPI) é uma exigência legal e essencial para a prevenção desses eventos, definido pela Norma Regulamentadora nº 06 (NR-06), que estabelece responsabilidades tanto para empregadores quanto para trabalhadores em questões referentes ao fornecimento, uso e fiscalização dos EPIs (BRASIL. Ministério do Trabalho, 2018).

Mesmo com a obrigatoriedade legal, o controle sobre o uso efetivo dos EPIs ainda apresenta limitações associadas à fiscalização, como em situações passíveis de falhas humanas, cobertura limitada e falta de evidência documental, de acordo com Stumm (2020). Nesse cenário, o uso de tecnologias emergentes como visão computacional e inteligência artificial representa uma oportunidade concreta para automatizar o monitoramento do uso de EPIs, garantindo maior precisão, escalabilidade e suporte à segurança ocupacional (GUNEY et al., 2024).

O atual mercado de visão computacional para segurança industrial já superou a fase de “prova de conceito” acadêmica e entrou em um estágio de produtização agressiva para consolidação comercial. A prova disso são as grandes corporações de hardware que estão embutindo redes neurais direto na infraestrutura de seus produtos (LogicLens Solutions, 2026).

1.2 Justificativas

O projeto possui razões reais para implementação devido às suas características técnicas de construção e à sua importância social dentro da indústria moderna, que busca o constante melhoramento dos processos e da segurança do trabalho.

1.2.1 Justificativas Técnicas

Na grande maioria das indústrias, o controle do uso de EPIs é realizado de maneira visual por agentes de segurança que estão constantemente presentes nas linhas de produção. Também estão presentes no desenvolvimento de uma cultura de segurança dentro das empresas, por meio do incentivo ao uso de ferramentas colaborativas que tornam as soluções dos problemas de segurança do trabalho mais rápidas (ORGANIZAÇÃO INTERNACIONAL DO TRABALHO, 2019). Ainda existe o auxílio de câmeras que captam vídeos dos colaboradores em movimento e os reúnem em uma central supervisionada por pessoas, porém, mesmo com o melhor treinamento existente, as pessoas que visualizam os vídeos nesta central possuem uma limitação cognitiva

natural humana e, portanto, não conseguem verificar um grande número de imagens ao mesmo tempo.

A aplicação de algoritmos de detecção de objetos, como os da família *You Only Look Once* (YOLO), permite identificar e classificar EPIs em tempo real a partir de imagens ou vídeos. Esta característica é consequência da arquitetura deste modelo de inteligência artificial ser simples se comparada com outros modelos que cumprem a mesma função, como *Region-based Convolutional Neural Network* (R-CNN), atendendo a requisitos de baixa latência para sistemas de alertas automáticos em tempo real, de acordo com Redmon et al. (2016).

Apesar da eficácia demonstrada por modelos de detecção em ambientes controlados, trabalhos existentes possuem limitações relacionadas a oclusão parcial de EPIs por maquinários, variações de iluminação, diferentes ângulos de visão, diversidade de contextos industriais e não apresentam validações em cenários reais.

1.2.2 Justificativas Sociais

Neste quesito, o projeto contribui diretamente para a preservação da vida e da integridade física dos trabalhadores, promovendo ambientes mais seguros. A automatização do monitoramento de EPIs reforça a cultura de prevenção e segurança, além de reduzir acidentes, afastamentos, processos trabalhistas judiciais. Estudos recentes demonstram que a aplicação de modelos YOLO em ambientes industriais apresenta resultados promissores no reconhecimento de EPIs como capacetes, óculos e coletes (ISAILOVIC et al., 2022).

Essa abordagem demonstra o compromisso das organizações com a responsabilidade social corporativa, alinhando-se também às diretrizes de *Environmental, Social and Governance* (ESG) (UNITED NATIONS, 2014).

1.3 Objetivos

O presente trabalho tem como objetivo técnico desenvolver e avaliar um modelo de rede neural profunda baseada na arquitetura YOLO para detecção de múltiplas classes de EPIs em ambientes industriais por meio de vídeo em tempo real.

O objetivo aplicado consiste em implementar a rede neural treinada em um sistema funcional de monitoramento em tempo real de ambientes industriais.

Hipótese: modelos YOLO podem atingir desempenho suficiente para monitoramento em tempo real utilizando hardware de custo reduzido.

1.4 Estrutura do Trabalho

Além da Seção 1 de introdução, a Seção 2 apresenta a fundamentação teórica feita a partir da revisão da literatura, abordando conceitos de redes neurais e comparando com o funcionamento biológico humano. A Seção 3 detalha a metodologia, descrevendo a preparação,

o treinamento e implementação do sistema. A Seção 4 expõe os resultados obtidos e a análise estatística do desempenho do modelo. Por fim, a Seção 5 apresenta as considerações finais, limitações do estudo e propostas para trabalhos futuros.

2 REVISÃO DA LITERATURA

Diversos estudos apontam as dificuldades associadas ao monitoramento manual do uso de EPIs em ambientes industriais, especialmente no que se refere à fiscalização contínua e à limitação de recursos humanos, o que pode comprometer o atendimento às normas regulamentadoras. Diante desse cenário, ferramentas baseadas em técnicas de visão computacional e inteligência artificial têm sido amplamente investigadas como alternativas para a automação do monitoramento do uso de EPIs.

Nesta Seção, abordam-se inicialmente os mecanismos biológicos e cognitivos que fundamentam o entendimento de imagens, estabelecendo um paralelo entre a percepção humana e o processamento digital. Em seguida, detalham-se os princípios das redes neurais e a evolução dos algoritmos de detecção de objetos, fornecendo a base teórica necessária para a compreensão das etapas de treinamento, validação e teste que serão descritas nos capítulos subsequentes.

2.1 Visão Humana

O sistema visual humano é frequentemente utilizado como inspiração para o desenvolvimento de modelos de visão computacional, uma vez que os estímulos visuais captados pela retina ocular são interpretados no córtex visual de forma hierárquica, que identifica bordas básicas, formas complexas (Íons Geométricos (Geons)), texturas e oclusões para categorizar e localizar objetos em um ambiente (GRILL-SPECTOR; MALACH, 2004; BIEDERMAN, 1987).

A percepção visual é o resultado da atividade biológica neural para decodificar estímulos visuais atuando juntamente com o raciocínio humano que organiza essas informações com base na experiência, memória e atenção.

No contexto industrial, embora a visão humana seja o padrão de referência, ela é limitada pela fadiga e pela impossibilidade de monitoramento onipresente, justificando a transposição desses mecanismos biológicos para algoritmos de visão computacional que buscam simular essa percepção de forma automatizada e contínua.

2.2 Visão Computacional

A visão computacional é um campo de estudo de inteligência artificial dedicado ao desenvolvimento de técnicas que permitem computadores interpretar automaticamente o conteúdo de imagens e vídeos, simulando o funcionamento da percepção visual humana por meio de algoritmos (CYGANER, 2013).

Com aplicações espaciais e militares, os primeiros estudos sobre o tema surgiram na década de 1960, quando foram inventados os primeiros computadores científicos capazes de

realizar cálculos complexos envolvendo grandes matrizes por meio de relações de álgebra linear e geometria analítica já consolidadas há séculos.

O processamento de dados cada vez mais ágil gerou avanços nos sistemas de digitalização de imagens, como sensores de câmeras e equipamentos de envio de informação digital, sendo possível posteriormente desenvolver ferramentas matemáticas e estatísticas para extrair informações relevantes a partir dessas imagens digitais (GONZALEZ; WOODS, 2010).

De acordo com Cyganek (2013), esse período marcou a transição da simples manipulação de imagens digitais para o desenvolvimento de modelos matemáticos capazes de representar objetos e cenas, estabelecendo os fundamentos teóricos que deram origem às modernas técnicas de processamento de imagens e aprendizado de máquina.

No ambiente industrial, a visão computacional tem sido amplamente aplicada na automação da fiscalização de segurança, especialmente na detecção do uso de EPIs. Estudos recentes demonstram que modelos baseados em aprendizado profundo são capazes de identificar automaticamente itens de segurança em imagens e vídeos, reduzindo a dependência da inspeção humana e aumentando a eficiência do monitoramento (GUNEY et al., 2024; ISAILOVIC et al., 2022).

Apesar dos avanços, esses estudos apresentam limitações importantes como utilização de conjuntos de dados limitados e genéricos, onde há variações significativas de iluminação, oclusões e posicionamento dos trabalhadores. Além disso, não avaliam o impacto de diferentes configurações de treinamento e aplicabilidade para cenários industriais reais, o que limita a compreensão do desempenho relativo dos modelos.

2.2.1 Tensores

No contexto da visão computacional, a imagem digital é representada matematicamente por meio de tensores, que consistem em matrizes multidimensionais que podem ser manipuladas por meio de operações matemáticas (GUNEY et al., 2024).

Existem diversos tipos de representações por meio de tensores, como a de uma imagem digital colorida no padrão *Red, Green, Blue* (RGB) de 8 bits, muito comuns hoje em dia. A sua representação é um tensor tridimensional de dimensões $H \times W \times 3$, em que H e W correspondem, respectivamente, à altura e largura da imagem, enquanto o valor 3 refere-se aos canais de cor vermelho, verde e azul, que variam de 0 (preto absoluto) a 255 (branco absoluto).

Isso torna possível representar diversas características de uma imagem em uma função matemática e usar esta função em redes neurais que realizam novos cálculos para extrair informações relevantes.

2.2.2 Neurônios Artificiais

A unidade elementar de processamento das redes neurais artificiais é o neurônio artificial, que executa uma operação matemática simples baseada na combinação linear de suas entradas,

conforme a Equação 1. Esta formulação é a base estrutural das Redes Neurais Artificiais (CYGANNEK, 2013).

$$z = \sum_{i=1}^n w_i x_i + b \quad (1)$$

Para processar uma imagem digital, os valores de intensidade armazenados no tensor precisam ser mapeados para as variáveis de entrada x_i por meio da normalização e linearização, onde a matriz multidimensional é convertida em um vetor unidimensional adequado para um neurônio artificial funcionar sem instabilidades causadas por números elevados, como 255 (GOODFELLOW; BENGIO; COURVILLE, 2016).

Exemplificando este processo, considere apenas dois pixels no padrão RGB dispostos horizontalmente, sendo o da esquerda vermelho (255, 0, 0) e o da direita azul (0, 0, 255). O tensor resultante T desta imagem possui o formato (1, 2, 3) e a representação matricial deste tensor é dada por:

$$T = \left[\begin{array}{c} \underbrace{\begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix}}_{\text{vermelho}} \quad \underbrace{\begin{bmatrix} 0 \\ 0 \\ 255 \end{bmatrix}}_{\text{azul}} \end{array} \right] \quad (2)$$

Para alimentar o neurônio, essa estrutura é normalizada para uma escala que varia entre 0 e 1 e depois linearizada em um vetor de entrada x com $n = 6$ elementos:

$$x = \left[\underbrace{1; 0; 0}_{\text{vermelho}}; \underbrace{0; 0; 1}_{\text{azul}} \right] \quad (3)$$

Como sua função é detectar exatamente estas cores, são utilizados pesos (w_i) que representam a especialização de cor. Portanto valores mais altos de peso no canal RGB ligado a cor vermelha indicam que o neurônio está mais sensível a esta cor, e o mesmo vale para o azul. O viés (b) é um valor constante que permite amplificar alguns canais para refinar e remover as instabilidades durante a identificação da cor desejada. (b) e (w_i) são parâmetros aprendidos durante o processo de treinamento da rede neural previamente realizado.

Considerando um vetor de pesos w e um viés b configurados hipoteticamente para refletir o objetivo dos neurônios exemplificados, tem-se:

$$w = [0, 5; -0, 1; -0, 1; -0, 2; 0, 1; 0, 3], \quad b = 0, 1 \quad (4)$$

O cálculo da Equação 1 de acordo com os parâmetros dados é detalhado na Equação 5 a seguir:

$$\begin{aligned}
z &= (1 \cdot 0,5) + (0 \cdot -(0,1)) + (0 \cdot -(0,1)) + (0 \cdot -(0,2)) + (0 \cdot 0,1) + (1 \cdot 0,3) + 0,1 \\
z &= 0,5 + 0 + 0 + 0 + 0 + 0,3 + 0,1 \\
z &= 0,9
\end{aligned} \tag{5}$$

O resultado $z = 0,9$ representa um nível alto de ativação do neurônio que supre o limiar necessário para a detecção da cor desejada. Isso ocorre porque os pesos positivos multiplicaram os sinais ativos da imagem para melhorar a identificação da cor. Se os sinais presentes multiplicassem os pesos negativos ou os pesos menores, resultaria em um z muito menor, indicando que o padrão não foi encontrado e que os pixels estão na ordem invertida (azul na esquerda e vermelho na direita).

2.2.3 Redes Neurais Convolucionais

Um conjunto destes neurônios artificiais forma uma camada neural artificial, que por sua vez, ao se unirem, formam uma Rede Neural Convolucional, do inglês, *Convolutional Neural Network* (CNN).

Estas estruturas totalmente integradas respeitam uma hierarquia de complexidade com funções matemáticas específicas, chamadas de convoluções. Esses níveis convolucionais são filtros de informação agrupados para identificar, por exemplo, bordas, contrastes, texturas, agrupamentos de objetos, formas e cores (CYGANKEK, 2013).

Para Redmon et al. (2016), a arquitetura, ou seja, a forma de organização destas camadas divide-se em em três partes, cada uma responsável por um nível de abstração da imagem:

- *backbone*: é a base da rede que recebe a imagem bruta na forma de tensores e extrai mapas de características. Quanto mais profunda a rede, mais complexos são os atributos que consegue identificar.
- *neck*: tem o papel de combinar os mapas de características vindos de diferentes camadas do *backbone*. Ele detecta objetos em múltiplas escalas, pois funde informações de baixa resolução (visão global) com informações de alta resolução (detalhes finos).
- *head*: é a camada final que é responsável por definir "o que" é o objeto (classificação) e "onde" ele está (localização). A resposta, ou seja, o tensor de saída desta camada e como se comunica e interage com outras camadas define se a arquitetura da rede opera em um ou dois estágios.

2.2.4 Redes Neurais Convolucionais por Região

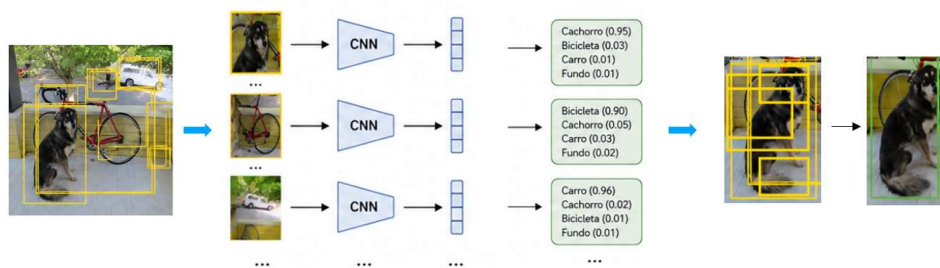
O desenvolvimento tecnológico contínuo em métodos de processamento computacional permitiu o aparecimento da primeira abordagem para tornar o processo mais ágil, que foi a

Region-based Convolutional Neural Network (R-CNN), de acordo com Redmon et al. (2016). Sua característica principal é a de operar em dois estágios:

- **divisão e classificação:** antes de as informações entrarem na rede neural, a imagem passa por um algoritmo que a divide em aproximadamente 2.000 regiões candidatas e as redimensiona para um formato fixo maior, o que aumenta a precisão. Cada região passa individualmente pelo *backbone* para a extração de características. O resultado é um vetor de probabilidades que classifica a classe do objeto, mas carece de contexto espacial global.
- **regressão da bounding box:** é o estágio final da *head*, que ocorre depois da classificação e utiliza outro algoritmo para ajustar as coordenadas da caixa delimitadora por meio da regressão linear tendo como base a junção de todas as informações de localização do objeto na imagem original.

A Figura 1 apresenta um fluxograma simplificado do funcionamento de uma rede *R-CNN*.

Figura 1 – Fluxograma simplificado de uma rede neural do tipo R-CNN.



Fonte: elaborado pelo autor (2026).

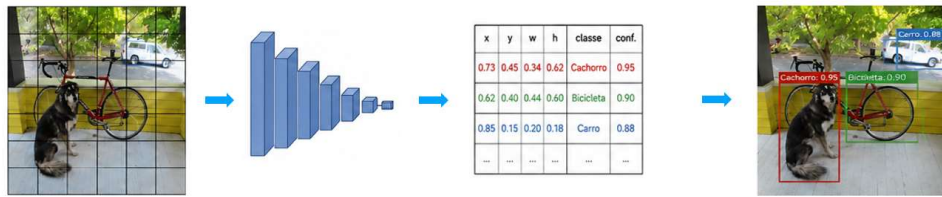
O fato de o processamento ser fragmentado e repetitivo para cada um dos muitos recortes gera um custo computacional elevado, impedindo a operação em tempo real, porém, se o custo operacional for atendido, este método ganha uma análise minuciosa e precisa da imagem.

2.2.5 Você Só Olha Uma Vez

A evolução das arquiteturas de detecção de objetos culminou no desenvolvimento do algoritmo *You Only Look Once* (YOLO), cujo nome em inglês, *You Only Look Once*, segundo Redmon et al. (2016), resume sua capacidade de processar a imagem inteira de uma única vez, superando a análise fragmentada e sequencial dos métodos anteriores e integrando simultaneamente a percepção global e os detalhes locais.

Essa eficiência é possível graças à integração completa das camadas convolucionais que geram tensores híbridos por meio da combinação das informações das camadas iniciais com as das camadas finais. Isso permite que a rede entenda o contexto global da imagem durante o treinamento, diferenciando-se de arquiteturas antigas que analisavam recortes isolados sem saber o que havia ao redor.

Figura 2 – Fluxograma simplificado de uma rede neural do tipo YOLO.



Fonte: elaborado pelo autor (2026).

Como pode ser visualizado na Figura 2, para realizar a detecção, a imagem é dividida virtualmente em uma grade e caso o centro geométrico de um EPI esteja presente dentro de uma célula específica desse grid, ela torna-se a responsável por processar os tensores híbridos e gerar a previsão final, contendo as coordenadas exatas, a classe do objeto e o nível de confiança de classe.

Dessa forma as arquiteturas da família YOLO realizam a extração de características e a predição em um único fluxo contínuo de dados (*backbone*, *neck* e *head*) de forma mais ágil, tornando-o eficientemente suficiente para aplicações em tempo real atualmente.

A família de algoritmos YOLO possui modelos mais robustos e profundos, como as versões *medium* ou *large*, que promovem ganhos marginais na precisão métrica, mas aumentam drasticamente o custo computacional e a latência. Trabalhos como de Guney et al. (2024) e Gupta et al. (2025) ressaltam que para que um sistema de monitoramento de segurança industrial seja viável, ele deve processar imagens em tempo real com baixa latência, conciliando o *trade-off* entre velocidade de inferência e precisão.

2.3 Treinamento de Redes Neurais

Assim como nos humanos, é preciso ensinar um cérebro artificial a enxergar e tomar decisões corretas, apresentando exemplos e corrigindo os erros. Esses exemplos constituem o banco de dados de treinamento, que possui informações precisas sobre a localização e a classe dos EPIs presentes em uma imagem.

Uma vez definida a arquitetura da rede neural e preparados os dados no banco de imagens com classes e coordenadas, inicia-se o processo iterativo. De acordo com Goodfellow, Bengio e Courville (2016), o algoritmo recebe uma imagem de treinamento, aplica pesos iniciais com valores aleatórios e, em seguida, compara o tensor de saída gerado com os valores exatos contidos no gabarito por meio da Função de Perda (*Loss Function*). Esta é a métrica matemática que quantifica o quão distante a previsão inicial feita pela rede está do resultado real, baseando-se nos erros de classificação, localização e confiança.

Com os valores do erro calculados, o algoritmo ajusta seus pesos internos (w) por meio da Retropropagação (*Backpropagation*), que calcula o gradiente do erro em relação aos pesos utilizados e realiza os ajustes necessários para minimizar a perda na próxima previsão.

A cada época (*epoch*), isto é, cada vez que o algoritmo processa todo o banco de imagens dividido em subconjuntos menores denominados lotes (*batches*) que permitem a rede treinar de forma mais eficiente, esses mecanismos fornecem dados para a rede neural realizar microajustes em direção a valores de erro cada vez menores.

O treinamento consiste em executar dezenas ou centenas de épocas até que o valor da função de perda se estabilize em um patamar mínimo aceitável, indicando que o modelo não consegue mais reduzir o erro significativamente da Taxa de Aprendizado (*Learning Rate*) configurada.

Ainda segundo Goodfellow, Bengio e Courville (2016), o objetivo final deste processo é a Generalização, ou seja, a capacidade da rede detectar corretamente os objetos em imagens novas e cenários reais, sem atingir o estado de Sobreajuste (*Overfitting*). Este fenômeno ocorre quando a rede se torna excessivamente complexa ou treina por tempo demasiadamente longo e começa a apenas decorar as imagens de treinamento, atingindo erros próximos a zero, mas falha drasticamente ao tentar detectar EPIs em imagens inéditas, pois perdeu sua capacidade de abstração.

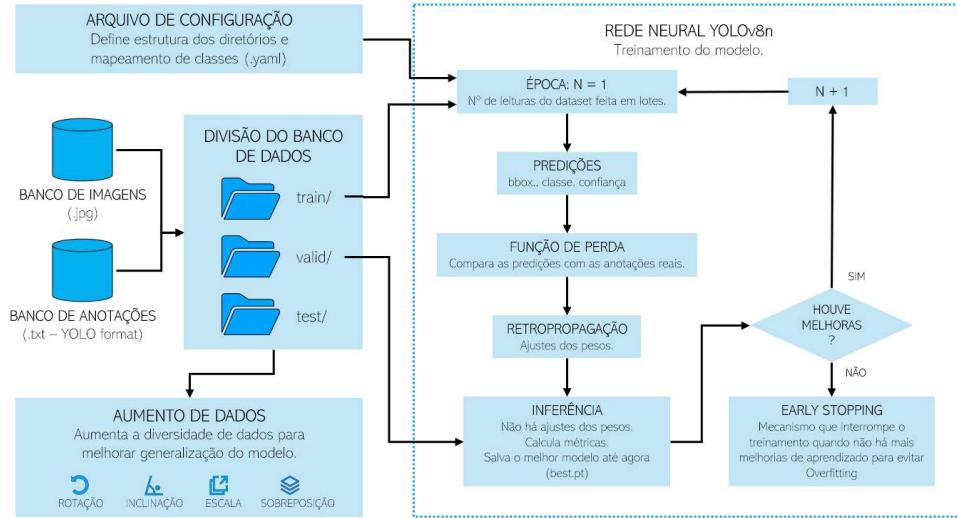
Para evitar isso, é comum usar técnicas que interrompem automaticamente o treinamento quando não há mais melhorias de aprendizado, como a Parada Antecipada (*Early Stopping*). Também são utilizadas técnicas de aumento de dados de treinamento (*Data Augmentation*) para simular variações nas imagens, como rotação, translação, escala, perspectiva e combinação de múltiplas camadas.

Treinar uma rede neural convolucional profunda do zero exige um poder computacional massivo e milhões de imagens. Para contornar essa limitação, utiliza-se a técnica de Aprendizado por Transferência (*Transfer Learning*), que inicia o treinamento com uma rede previamente treinada em um conjunto de dados genérico e que já possui a capacidade de extrair características visuais dos objetos.

A eficiência do treinamento também pode ser influenciada pela forma como os dados são carregados e disponibilizados ao modelo. A utilização de múltiplos processos de leitura (*Workers*) e o armazenamento temporário dos dados em memória de cachê são estratégias comuns para reduzir o tempo de acesso a disco e otimizar o desempenho computacional.

Esses conceitos formam a base teórica necessária para a compreensão dos métodos utilizados no treinamento de modelos de detecção de objetos baseados em redes neurais convolucionais. Um esquema simplificado do processo de treinamento é apresentado na Figura 3.

Figura 3 – Fluxograma simplificado de treinamento do modelo YOLO.



Fonte: elaborado pelo autor (2026).

2.4 Validação de Modelo de Rede Neural

Com o processo anterior concluído, é preciso avaliar se o modelo foi treinado corretamente e pode ser aplicado de forma segura. Isso é feito seguindo métricas geradas durante o treinamento, que são consolidadas na literatura de aprendizado profundo e detecção de objetos (REDMON et al., 2016).

As avaliações são feitas comparando as detecções geradas pela inteligência artificial com a localização real dos objetos nas imagens de teste presentes no gabarito, técnica conhecida como *Intersection over Union* (IoU). Nesta etapa é gerado o erro de posição da caixa delimitadora (*box loss*), dado pela Equação 6.

$$box_loss = 1 - IoU = 1 - \frac{\text{Área da Interseção}}{\text{Área da União}} \quad (6)$$

Em uma classificação, o erro ao dizer qual classe é (*cls loss*) influencia diretamente na precisão (*precision*), que mede a quantidade de acertos que o modelo teve ao dizer que um objeto estava realmente presente na imagem. Essa métrica precisa ser a menor possível, pois não é seguro errar a classificação de um EPI, como dizer que um capacete é uma sombra, por exemplo. Valores elevados de *precision* evitam alarmes falsos excessivos.

Considerando as ocorrências de verdadeiro positivo (TP) e falso positivo (FP), tem-se a Equação 7 para o cálculo da precisão.

$$precision = \frac{TP}{TP + FP} \quad (7)$$

A revocação (*recall*) mede quantos objetos realmente existentes na imagem o modelo conseguiu

detectar corretamente, ou seja, indica a sensibilidade do modelo. Por exemplo, de 100 pessoas, o modelo detectou 95, ou seja, valores baixos de *recall* podem fazer o modelo deixar de detectar funcionários sem EPI, o que é inaceitável em termos de segurança.

Considerando as ocorrências de verdadeiro positivo (TP) e falso negativo (FN), tem-se a Equação 8 para o cálculo da revocação.

$$recall = \frac{TP}{TP + FN} \quad (8)$$

Os parâmetros de mAP e mAP50-95 resumem todas as informações de precisão e confiança em uma única métrica. O primeiro indica uma avaliação mais realista, que é mais relevante para indústria, enquanto o segundo é mais rigoroso, pois exige que a caixa desenhada pelo modelo esteja muito próxima da caixa real do objeto.

Para diminuir os alarmes falsos e as não detecções de funcionários sem EPI, deve-se adotar um nível de confiança alto, assim é possível identificar as classes críticas na identificação.

A matriz de confusão (*confusion matrix*) detalha os erros de classificação, mostrando quais classes o modelo está confundindo entre si, como por exemplo, confundir luvas com ferramentas. Altos valores fora da diagonal principal indicam que o modelo precisa de melhorias de dataset.

Diversos trabalhos têm explorado esta capacidade, por exemplo, Gupta et al. (2025), que aplicaram o modelo YOLOv8 para detecção de itens de segurança e reportaram precisão de 92,5% e mAP@0.5 de 91,2%.

De forma semelhante, Isailovic et al. (2022) investigaram a detecção de EPIs de cabeça utilizando YOLOv4, alcançando mAP de 94,5% na identificação de capacetes, com desempenho robusto mesmo em cenários industriais.

Já Guney et al. (2024) desenvolveram um sistema baseado em YOLO para monitoramento contínuo, obtendo uma precisão média de 89% e revocação de 87%, destacando que as maiores dificuldades de detecção residem em classes de menores dimensões, como protetores auriculares e óculos, devido à baixa resolução espacial em imagens capturadas a longas distâncias.

Ambos os autores ressaltam que para que um sistema de monitoramento de segurança industrial seja viável, ele deve processar imagens em tempo real com baixa latência, conciliando o *trade-off* entre velocidade de inferência e precisão.

Em sua publicação, Ahmad e Rahimi (2024) realizaram o treinamento do modelo com sua base de dados utilizando a arquitetura YOLOv9-e, que é uma versão da família YOLO de maior complexidade e, portanto, exige um maior poder computacional, alcançando um mAP@0.5 de 64,8%, precisão de 73,4% e revocação de 59,7%. Em seu trabalho, também existem comparações com outras versões, destacando-se a YOLOv8n, que obteve um mAP@0.5 de 58,0%, precisão de 67,5% e revocação de 53,6%.

Apesar da ampla utilização dessas métricas, muitos trabalhos se limitam à análise quantitativa, sem investigar qualitativamente os erros do modelo. A ausência de testes em

ambientes reais, que possuem grande variação de condições ambientais, e a falta de métricas relacionadas ao desempenho computacional, como latência, ainda são lacunas relevantes na literatura (SOARES; OLIVEIRA; ARAÚJO, 2023; SANTOS, 2023).

2.5 Inferência de Modelo de Rede Neural

Após a etapa de treinamento, o modelo passa a ser utilizado em sua fase operacional, que consiste na aplicação da rede neural previamente treinada para a detecção automática de EPIs em imagens e vídeos reais, demonstrando sua viabilidade prática como ferramenta de apoio à segurança do trabalho industrial.

O modelo desenvolvido pode operar em três modalidades distintas, permitindo sua utilização tanto em análises posteriores quanto em aplicações em tempo real.

2.5.1 Vídeos Gravados

Nesta modalidade, o sistema realiza a análise de arquivos de vídeo previamente gravados e armazenados em um diretório local ou em serviço de armazenamento em nuvem, como o Google Drive.

O processo consiste em carregar o arquivo de vídeo e aplicar o modelo treinado para obter como saída um novo arquivo de vídeo contendo as caixas delimitadoras, classes detectadas e níveis de confiança sobrepostos à imagem original. Essa abordagem é especialmente útil para auditorias de segurança, análises retrospectivas de incidentes e validações técnicas do desempenho do modelo em diferentes cenários industriais.

2.5.2 Câmeras de Vídeo

A segunda modalidade permite a utilização do sistema em tempo real por meio da captura direta de imagens provenientes de uma câmera de vídeo conectada ao computador, como webcams USB, câmeras IP ou sistemas de vigilância industrial.

Neste modo de operação, cada quadro capturado pela câmera é imediatamente processado pela rede neural, permitindo a detecção instantânea do uso de EPIs. Caso seja identificado algum colaborador em situação de não conformidade, o sistema pode ser integrado a mecanismos de alerta visual ou sonoro, fornecendo resposta imediata para mitigação de riscos.

Essa configuração viabiliza a implantação do sistema em postos de trabalho, portarias industriais ou áreas críticas de produção juntamente com um sistema automatizado de avisos.

2.5.3 Captura de Tela

A terceira modalidade consiste na execução do modelo a partir da captura direta da tela do computador, permitindo que o sistema processe qualquer vídeo exibido no monitor, independentemente de sua origem.

Dessa forma, é possível realizar a inferência em transmissões ao vivo de sistemas de câmeras industriais, softwares de monitoramento, gravações reproduzidas localmente ou transmissões remotas, sem a necessidade de conexão física com o dispositivo de captura de imagem.

A possibilidade de testar o sistema diretamente na tela do computador facilita o ciclo de depuração em casos que o desenvolvedor não se encontra fisicamente em uma planta industrial. O uso de vídeos e simulações em tela substitui a exposição a riscos ocupacionais dos ambientes industriais e contorna as limitações de hardware, como falta de câmera de segurança, o que não impede a execução de um processo contínuo de melhoria e re-treinamento da rede neural, garantindo maior agilidade no refinamento dos algoritmos.

3 DESENVOLVIMENTO

Esta seção detalha a constituição da base de dados, os recursos computacionais utilizados, a arquitetura da rede neural selecionada e a estratégia adotada para o treinamento e validação do modelo de detecção de EPIs.

3.1 Base de Dados (Dataset)

Para que a rede neural aprenda a identificar os equipamentos de segurança em um cenário real, utilizou-se o conjunto de dados denominado SH17 (AHMAD; RAHIMI, 2024), voltado especificamente para a detecção de equipamentos de proteção individual e segurança humana na indústria de manufatura moderna. Este conjunto de dados possui 8.099 imagens de resolução variando de 1920x1002 a 8192x5462 pixels, coletadas em ambientes industriais diversos com foco na indústria de manufatura.

O conjunto contém imagens já anotadas com as coordenadas de onde cada objeto identificado se encontra na imagem e como aquele objeto foi classificado. O banco de dados possui no total 75.994 instâncias distribuídas em 17 classes representadas por um número, dadas na Tabela 1.

Tabela 1 – Classes presentes no *dataset*.

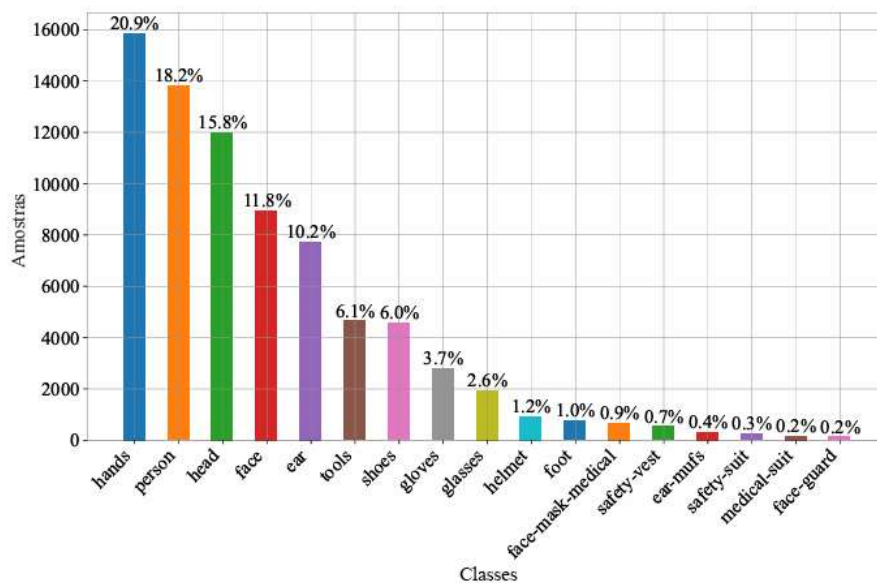
Classe	Nº
Pessoa (<i>Person</i>)	0
Orelha (<i>Ear</i>)	1
Protetor auricular (<i>Earmuffs</i>)	2
Rosto (<i>Face</i>)	3
Protetor facial (<i>Face-guard</i>)	4
Máscara facial médica (<i>Face-mask-medical</i>)	5
Pé (<i>Foot</i>)	6
Ferramentas (<i>Tools</i>)	7
Óculos (<i>Glasses</i>)	8
Luvas (<i>Gloves</i>)	9
Capacete (<i>Helmet</i>)	10
Mãos (<i>Hands</i>)	11
Cabeça (<i>Head</i>)	12
Avental médico (<i>Medical-suit</i>)	13
Sapatos (<i>Shoes</i>)	14
Roupa de segurança (<i>Safety-suit</i>)	15
Traje de segurança (<i>Safety-vest</i>)	16

Fonte: elaborado pelo autor (2026), a partir de Ahmad e Rahimi (2024).

Embora o foco principal deste trabalho esteja na detecção de equipamentos de proteção

individual, a presença de classes que compõem a estrutura anatômica básica, como pessoa, cabeça, rosto, pé e mãos, é relevante para fornecer contexto espacial ao modelo e auxiliar na correta associação entre o trabalhador e os EPIs utilizados. A distribuição e balanceamento das classes presentes no banco de imagens é ilustrado na Figura 4 .

Figura 4 – Distribuição de todas as instâncias de classes.



Fonte: Ahmad e Rahimi (2024).

As imagens são divididas aleatoriamente em três subconjuntos para garantir a integridade do processo de aprendizado. Essa estratégia é amplamente utilizada em problemas de aprendizado de máquina por proporcionar quantidade suficiente de dados para o treinamento, ao mesmo tempo em que reserva amostras independentes para validação e teste.

As proporções adotadas não constituem uma regra rígida, podendo variar de acordo com o tamanho e as características do conjunto de dados. Em geral, distribuições próximas ao que são mostradas a seguir apresentam bom equilíbrio entre aprendizado, ajuste de hiperparâmetros e avaliação da capacidade de generalização do modelo. Pequenas variações nessas porcentagens normalmente não afetam significativamente os resultados, desde que cada subconjunto contenha quantidade representativa de amostras.

- Treino (70%): Subconjunto majoritário utilizado para o ajuste dos pesos da rede neural.
- Validação (20%): Utilizado para monitorar o desempenho durante o treinamento e prevenir o Sobreajuste.
- Teste (10%): Imagens inéditas reservadas para a avaliação final da capacidade de generalização do modelo.

Em cada subconjunto encontra-se dois tipos de arquivos:

- Imagens (.jpg): Arquivos de imagem no formato JPEG que contêm as cenas industriais com trabalhadores utilizando ou não os EPIs. Alguns exemplos são apresentados na Figura 5.

Figura 5 – Exemplos de imagens do *dataset*.



Fonte: Ahmad e Rahimi (2024).

- Anotações (.txt): Arquivos de texto com mesmo nome dos arquivos de imagem, contendo uma linha para cada objeto identificado. Cada linha contém um número que identifica a classe do objeto, as coordenadas normalizadas do centro da caixa delimitadora nos eixos vertical e horizontal, bem como sua largura e altura, como pode ser visualizado na Figura 6.

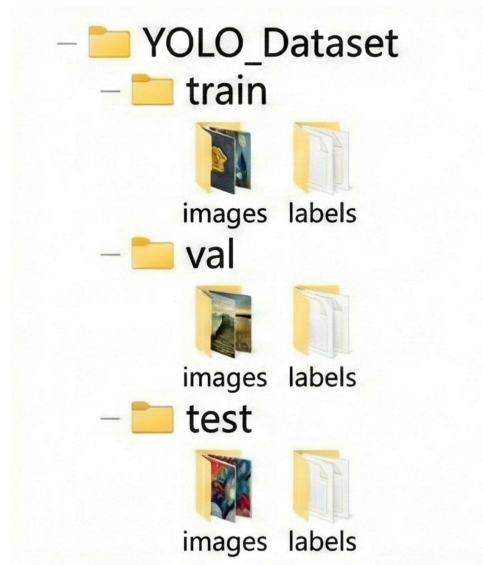
Figura 6 – Exemplo de arquivo de anotação do *dataset*.

```
12 0.431781 0.255571 0.018265 0.042000
0 0.426210 0.290286 0.028311 0.114857
```

Fonte: (AHMAD; RAHIMI, 2024).

Os autores fazem considerações sobre as qualidades das anotações e dimensões dos objetos presentes, em que, das 75.994 instâncias, 59.025 (78%) delas ocupam menos de 5% da área total da imagem e 39.764 (52%) ocupam menos de 1% (AHMAD; RAHIMI, 2024).

O banco de imagens foi armazenado em um serviço de armazenamento em nuvem, o Google Drive, que permite o acesso remoto e compartilhamento fácil dos dados durante o desenvolvimento do projeto. Na Figura 7 pode-se ver a divisão das pastas.

Figura 7 – Organização dos diretórios do *dataset*.

Fonte: elaborado pelo autor (2026).

3.2 Ambiente Computacional

O desenvolvimento, treinamento e validação do modelo foram realizados em nuvem, utilizando a plataforma Google Colab que permite o acesso gratuito a uma Unidade de Processamento Gráfico, do inglês, *Graphics Processing Unit* (GPU), de alto desempenho, como a Tesla T4, essencial para acelerar os cálculos matriciais complexos exigidos pelas redes neurais, dispensando a necessidade de uma estação de trabalho física de alto custo.

O ambiente de desenvolvimento utilizou a interface Jupyter Notebook para escrever o código na linguagem Python, que possui bibliotecas fundamentais na construção de redes neurais, tais como:

- PyTorch: Responsável por fornecer a infraestrutura matemática para o aprendizado profundo, gerenciar os tensores matemáticos de dados e realizar a comunicação direta com a GPU para executar o algoritmo de retropropagação e o cálculo de gradientes de forma eficiente (PYTORCH, 2016).
- Ultralytics: É a biblioteca que fornece os algoritmos pré-configurados da arquitetura YOLO, incluindo funções para carregamento de dados e métricas de avaliação, eliminando a necessidade de reescrever o código de cada camada convolucional manualmente (ULTRALYTICS, 2023).
- OpenCV: Biblioteca que permite a manipulação e processamento de imagens e vídeos, sendo essencial para a visualização dos resultados (OPENCV, 2000).

É importante ressaltar que o Google Colab não oferece suporte para acesso direto a dispositivos de captura de vídeo, como webcams e câmeras de vigilância. Nesses casos, o

modelo treinado foi exportado e integrado em um ambiente Python local, utilizando as mesmas bibliotecas mencionadas acima para garantir a consistência na execução do algoritmo.

3.3 Configuração de Treinamento

Como o banco de imagens se encontra em nuvem no formato de um arquivo compactado e os resultados dos treinamentos precisam ser armazenados, é preciso vincular o Google Colab ao Google Drive e realizar o processo de descompactação dos arquivos. Isso pode ser realizado por meio do código em Python visto na Figura 8 a seguir.

Figura 8 – Código para integração de ambientes e descompactação do *dataset*.

```
# Realiza a conexão entre o Google Colab e o Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Importa as bibliotecas utilizadas para manipulação de arquivos e diretórios
import os
import shutil

# Define o caminho do arquivo compactado contendo o conjunto de dados
zip_path = '/content/drive/MyDrive/YOLO/dataset.zip'

# Define o diretório onde o conjunto de dados será armazenado
dataset_dir = '/content/dataset'

# Executa o comando de descompactação do arquivo ZIP para o diretório definido
!unzip -q "$zip_path" -d "$dataset_dir"
```

Fonte: elaborado pelo autor (2026).

A estrutura dos diretórios e o mapeamento das classes foram definidos em um arquivo de configuração com extensão *.yaml*, utilizado como entrada para o algoritmo gerenciar o carregamento de dados, como pode ser visto na Figura 9.

Figura 9 – Código para criação do arquivo no formato .yaml.

```

# Definir as classes conforme o dataset SH17
classes = [
    'Person',
    'Ear',
    'Earmuffs',          # 2
    'Face',              # 3
    'Face-guard',        # 4
    'Face-mask-medical', # 5
    'Foot',              # 6
    'Tools',             # 7
    'Glasses',           # 8
    'Gloves',            # 9
    'Helmet',            # 10
    'Hands',             # 11
    'Head',              # 12
    'Medical-suit',      # 13
    'Shoes',             # 14
    'Safety-suit',       # 15
    'Safety-vest'        # 16
]

# Define o dicionário de configuração do dataset
config = {
    'path': '/content/dataset/EPI', # Caminho raiz do dataset
    'train': 'train/images', # Caminho relativo para imagens de treino
    'val': 'valid/images', # Caminho relativo para imagens de validação
    'test': 'test/images', # Caminho relativo para imagens de teste
    'nc': 17, # Número de classes
    'names': classes # Nomes das classes
}

# Define o caminho para salvar o arquivo YAML
yaml_path = '/content/custom_data.yaml'
with open(yaml_path, 'w') as f:
    yaml.dump(config, f)

```

Fonte: elaborado pelo autor (2026).

Optou-se pela utilização da arquitetura YOLOv8n (versão *nano*), pois exige um menor número de parâmetros treináveis e possui uma boa relação entre velocidade de inferência e precisão. Embora arquiteturas como esta entregam menor precisão, elas diminuem drasticamente o custo computacional e a latência. Como o objetivo final deste trabalho é a aplicação em tempo real, a versão YOLOv8n sacrifica incrementos de precisão para garantir uma alta taxa de quadros por segundo, do inglês, *Frames Per Second* (FPS), permitindo que o sistema seja embarcado em hardwares industriais de baixo custo de processamento.

A execução do treinamento foi controlada por parâmetros não arbitrários, sendo definidos empiricamente com base nas limitações de hardware e nas características do dataset SH17. O número de *epochs* de 150 representa um teto empírico seguro para garantir a convergência da função de perda em conjuntos de dados do porte do utilizado neste trabalho. Para que o modelo não sofra de sobreajuste, utilizou-se o valor de 30 *epochs* para o parâmetro de *patience*. O valor de *batch* foi delimitado para 28, pois a capacidade de memória de vídeo da GPU utilizada

(16 GB) não permite lotes de imagens maiores, o que causaria o erro de estouro de memória, enquanto que lotes menores tornariam o cálculo ruidoso e prolongariam o tempo de treinamento. A resolução das imagens de 640 pixels foi mantida no padrão de arquiteturas YOLO, pois oferece um equilíbrio ideal entre a preservação de detalhes espaciais dos EPIs e a velocidade de processamento. A otimização de fluxo com 2 *workers* e armazenamento em *cache* na memória RAM são configurações essenciais em ambientes de nuvem para evitar gargalos entre a CPU e a GPU, acelerando a leitura dos lotes subsequentes e reduzindo o tempo ocioso da placa de vídeo. Além disso, argumentos de aumento de dados foram configurados para simular variações nas imagens, como rotação, translação, escala, perspectiva e combinação de múltiplas camadas, o que é crucial para melhorar a capacidade de generalização do modelo, especialmente considerando a diversidade de cenários industriais do dataset escolhido.

Os valores de configuração adotados estão resumidos na Tabela 2 e o carregamento do modelo com seus parâmetros, podem ser visualizados na Figura 10.

Tabela 2 – Parâmetros de configuração de treinamento do modelo YOLOv8n.

Parâmetro	Configuração adotada
data	custom_data.yaml
epochs	150
batch	28
patience	30
imgsz	640
device	0
workers	2
cache	ram
degrees	10
translate	0.1
scale	0.6
perspective	0.0005
mixup	0.15
mosaic	1.0
copy_paste	0.1

Fonte: elaborado pelo autor (2026).

Figura 10 – Código para carregamento do modelo YOLO.

```

from ultralytics import YOLO # Importa a biblioteca

model = YOLO('yolov8n.pt') # Carrega o modelo Nano

results = model.train(      # Inicia loop de treinamento
    data='/content/custom_data.yaml', # Arquivo de configuração do dataset
    epochs=150,                # Total de iterações no dataset
    patience=30,                # Para o treino se não houver melhora

    # ——— HARDWARE & RESOLUÇÃO ———
    imgsz=640,                  # Resolução das imagens
    batch=28,                   # Tamanho do lote
    device=0,                   # Usa a GPU
    workers=2,                  # Threads para carregamento de dados
    cache='disk',               # Salva dados em disco para acelerar

    # ——— DATA AUGMENTATIONS ———
    degrees=10,                 # Rotação de ±10°
    translate=0.1,              # Translação de 10%
    scale=0.6,                  # Zoom in/out de até 60%
    perspective=0.0005,         # Distorção 3D leve
    mosaic=1.0,                 # Combinação de camadas por sobreposição
    mixup=0.15,                 # Combinação de camadas por sobreposição
    copy_paste=0.1,             # Combinação de camadas por colagem

    # ——— SALVAR ———
    project='/content/drive/MyDrive/YOLO/treinamentos', # Pasta raiz
    name='treino_1',            # Nome da pasta
    pretrained=True,            # Usa pesos pré-treinados
    single_cls=False,           # Multiclasse
    plots=True,                 # Gera gráficos de métricas
    exist_ok=True               # Sobrescreve pasta se existir
)

```

Fonte: elaborado pelo autor (2026).

3.4 Configuração de Validação do Modelo Treinado

Para validar se o sistema é seguro e eficiente para uso industrial, o desempenho do modelo foi avaliado utilizando o código elaborado que pode ser visto na Figura 11, que utiliza um conjunto de imagens que não foram utilizadas durante o treinamento.

Figura 11 – Código para validação do treinamento realizado.

```
# Carrega o melhor modelo treinado
model = YOLO('/content/drive/MyDrive/YOLO/treinamentos/treino_1/weights/best.pt')

metrics = model.val(
    data='/content/custom_data.yaml', # Arquivo de configuração do dataset
    split='test',                     # Conjunto de imagens inéditas
    imgsz=640,                        # Resolução das imagens
    batch=28,                         # Tamanho do lote
    device=0,                         # Usa a GPU

    # — SALVAR —
    project='/content/drive/MyDrive/YOLO/validacao', # Pasta raiz
    name='validacao_1',                             # Nome da pasta
    exist_ok=True,                                   # Sobrescreve se existir
    plots=True,                                     # Salva Matriz de Confusão e Curvas F1
)
```

Fonte: elaborado pelo autor (2026).

Assim como na etapa anterior, alguns parâmetros essenciais foram utilizados novamente para otimizar e manter a compatibilidade de dados, como pode ser visto na Tabela 3.

Tabela 3 – Parâmetros de configuração de validação do modelo treinado.

Parâmetro	Configuração adotada
data	custom_data.yaml
split	test
imgsz	640
batch	28
device	0

Fonte: elaborado pelo autor (2026).

Os melhores pesos são utilizados nesta etapa, sendo preciso definir o diretório onde está localizado juntamente com o conjunto de imagens que será avaliado. Diferente da etapa anterior, aqui não ocorrem épocas, o processo é realizado de uma única vez, eliminando a necessidade de alocar os arquivos temporários na memória da GPU.

3.5 Inferência de Modelo Treinado

As possibilidades de aplicação do modelo podem ser concatenadas em apenas um aplicativo computacional que possui uma interface gráfica para a visualização e interação com os resultados.

Esta integração a um software para uso real foi desenvolvida em linguagem Python e permite a seleção dinâmica da fonte de vídeo por meio da interface, podendo utilizar tanto uma câmera conectada quanto a captura da tela do computador em que o aplicativo está sendo executado.

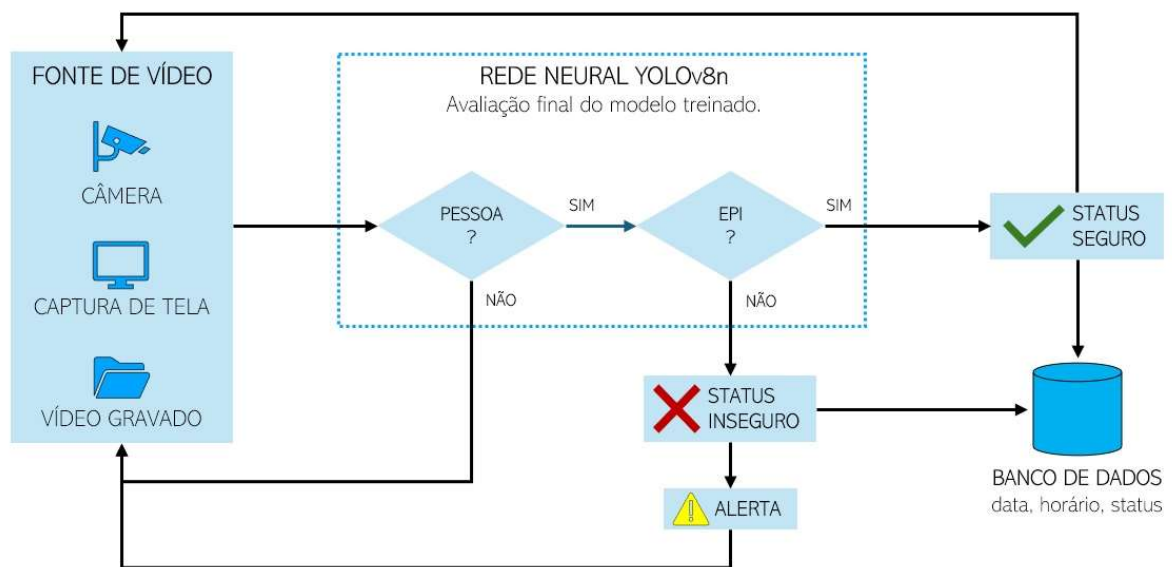
O sistema realiza a detecção simultânea de pessoas e EPIs previamente configurados para o local onde se situa a câmera e contabiliza o número de indivíduos presentes na cena e quais EPIs estão utilizando. Sempre que uma pessoa é identificada sem o equipamento obrigatório, o sistema emite um alerta visual, que pode ainda ser configurado para emitir outros avisos, como sonoro ou em forma de notificação automática, conforme as necessidades do usuário.

Além disso, o software permite o ajuste dinâmico dos níveis de confiança para cada classe. Também possui um sistema de tolerância de oclusões temporárias quando um frame não possui as informações visuais suficientes, possibilitando a personalização do sistema de acordo com requisitos específicos de segurança do ambiente monitorado.

Para análises futuras e fins de registro de informações importantes, quando o sistema identifica uma pessoa, registra informações como data, hora e status de segurança em um banco de dados. Também é possível configurar quais dados devem ser armazenados, podendo até mesmo salvar a imagem destacada do trabalhador identificado e monitorado.

O código-fonte completo do software desenvolvido está disponível para consulta no repositório oficial do projeto (BÓIS, 2026) e o fluxograma básico da arquitetura é apresentado na Figura 12.

Figura 12 – Fluxograma do software de aplicação do modelo treinado.



Fonte: elaborado pelo autor (2026).

A inferência do modelo foi realizada em uma máquina local equipada com processador Intel Core i7 de sétima geração e 8 GB de memória RAM, sem aceleração de GPU dedicada.

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta e discute os resultados obtidos ao longo do desenvolvimento do sistema de monitoramento de segurança industrial baseado em visão computacional e redes neurais convolucionais do tipo YOLO. São abordadas as avaliações de desempenho, os testes de inferência em tempo real por meio de um aplicativo desenvolvido pelo autor, as limitações identificadas e as possibilidades de aperfeiçoamento para trabalhos futuros.

4.1 Análise da Base de Dados

Embora o conjunto de dados SH17 forneça a base para o treinamento inicial de modelos de detecção de EPIs, uma análise da distribuição de instâncias de classes revela características inerentes que impactam diretamente o desempenho e a capacidade de generalização da rede neural.

Classes como "Mãos"(15.850) e "Pessoa"(13.802) possuem representatividade massiva, enquanto "Máscara facial"(669) e "Protetor facial"(134) são sub-representadas. Este desequilíbrio extremo faz com que o modelo priorize o aprendizado das classes majoritárias, resultando em baixa precisão para as classes minoritárias. Esse viés de amostragem faz com que a função de perda priorize o aprendizado das classes com maior representatividade para reduzir o erro global, o que justifica a dificuldade do modelo em identificar as classes menos representadas.

Além disso, a predominância de objetos em escala diminuta, como protetores auriculares e óculos, eleva a complexidade da extração de características feita pela rede neural, considerando que muitas dessas instâncias ocupam menos de 1% da área total da imagem.

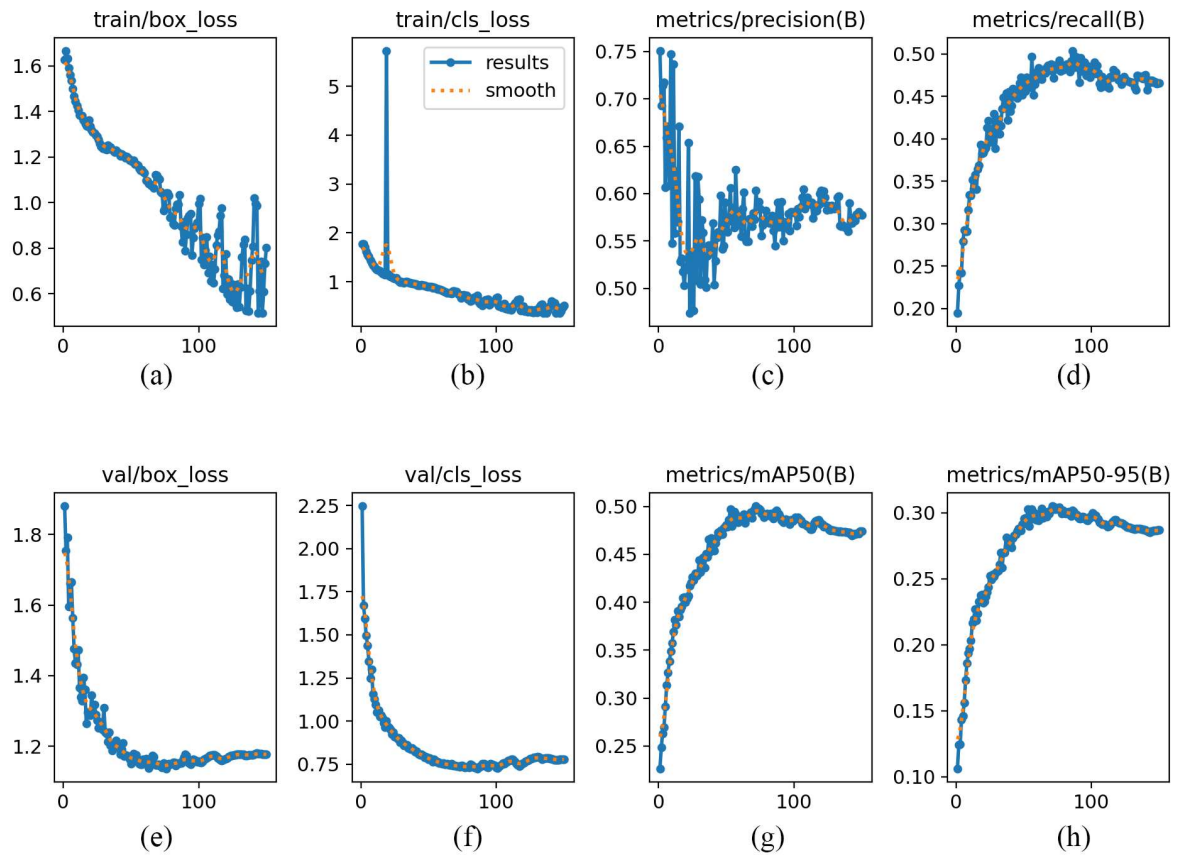
O modelo treinado herda representações atreladas a padrões de iluminação de galpões, ângulos de observação específicos de câmeras e EPIs com cores padronizadas. Isso restringe a sua aplicação em cenários diferentes, o que ocasiona queda de desempenho.

A mitigação desses desafios requer a adoção de técnicas de balanceamento de classes, como o aumento de dados (*data augmentation*), utilização de algoritmos que atribuem maior importância às classes sub-representadas durante o processo de treinamento e adição de amostras mais detalhadas destas classes inseridas no ambiente alvo ao dataset para readaptar a rede neural.

4.2 Análise das Curvas de Treinamento e Validação

A Figura 13 apresenta a evolução das métricas de desempenho e das funções de perda ao longo das 150 épocas de treinamento.

Figura 13 – Métricas de desempenho do modelo treinado.



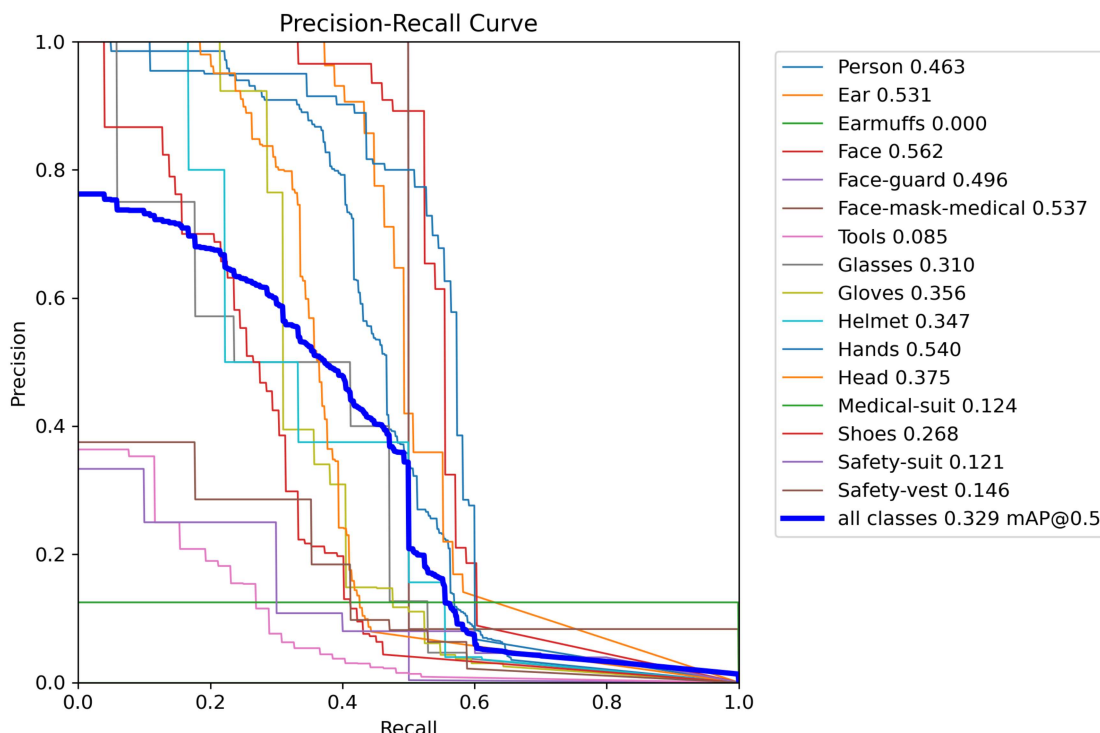
Fonte: elaborado pelo autor (2026).

Observando as curvas de perda de treinamento (a) e (b), nota-se um decaimento consistente até o final do processo, indicando que a rede foi capaz de aprender as características do conjunto de dados de treino e posicionar as caixas delimitadoras de forma eficaz. Entretanto, ao analisar as perdas de validação (e) e (f), percebe-se um comportamento de estabilização e leve ascensão com ruídos a partir da época 100. Esse fenômeno sugere um início de sobreajuste (*overfitting*), onde o modelo passou a se especializar excessivamente nos dados de treino, reduzindo ligeiramente sua capacidade de generalização. Como a sobreposição entre caixas reais e previstas pelo modelo não precisa ser perfeita, a segurança em ambiente industrial não é comprometida com esses resultados.

As curvas de precisão média (g) e (h) atingiram seus valores máximos no intervalo entre as épocas 80 e 100, apresentando uma leve oscilação descendente nas épocas finais. O sistema de *early stopping* e a seleção automática do melhor peso garantiram que o modelo final utilizado fosse aquele correspondente ao ponto de máximo desempenho global, mitigando os efeitos da estagnação observada no final do ciclo.

A Figura 14 apresenta as curvas de Precisão-Revocação (PR Curve) obtidas especificamente no conjunto de 100 imagens inéditas, o que revela o comportamento do modelo diante de desafios reais de generalização.

Figura 14 – Curvas Precisão-Revocação do modelo treinado.



Fonte: elaborado pelo autor (2026).

A validação do modelo realizada revelou um desempenho global de mAP@0.5 de 32,9%, precisão de 57,2% e revocação de 48,0%. Embora o mAP@0.5 seja baixo, ele se aproxima dos resultados obtidos por Ahmad e Rahimi (2024) para a arquitetura utilizada.

Isso reflete a alta complexidade do dataset utilizado, que possui objetos que ocupam pouco espaço nas imagens e escassez de exemplos positivos, como na classe de protetores auriculares, o que resultou em uma métrica nula.

É possível destacar que as classes com melhor desempenho foram de rosto, mãos e máscara facial médica. A curva da classe de mãos demonstra uma resiliência notável, mantendo a precisão acima de 40% mesmo com níveis de revocação superiores a 50%.

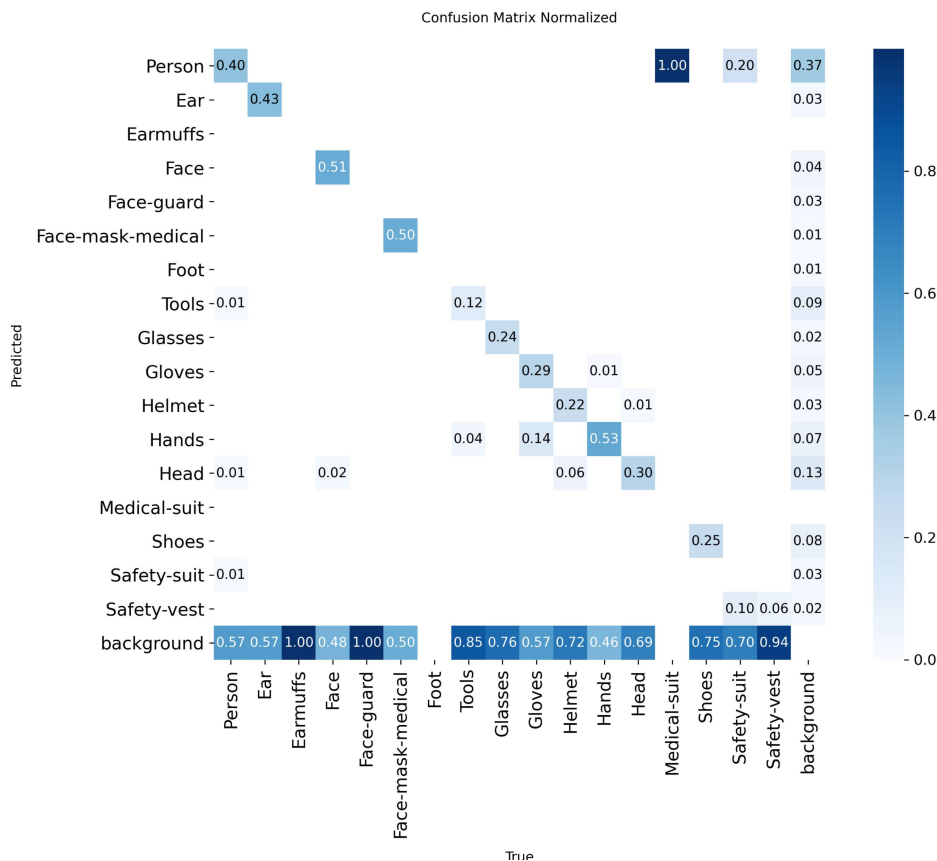
Os equipamentos de proteção principais, como luvas e capacete apresentaram um desempenho intermediário. Suas curvas exibem um declínio mais acentuado no início, indicando que o modelo detecta com alta confiança os EPs que estão em destaque e com boa iluminação, mas perde precisão rapidamente em objetos menores ou com baixa iluminação.

A classe pessoa, que obteve resultados quase perfeitos na validação, sofreu uma redução de desempenho no teste. Isso sugere que o modelo aprendeu a identificar pessoas muito bem no contexto específico do treino, mas encontrou dificuldades em variações de pose ou vestimenta no conjunto de imagens inédito.

O aspecto de degraus observado nas linhas é característico de avaliações em conjuntos de dados menores, onde uma classe com menos instâncias possui um peso estatístico maior na construção da curva. Para compreender detalhadamente a natureza dos erros cometidos pelo

modelo no conjunto de teste, foi gerada a Matriz de Confusão Normalizada, apresentada na Figura 15.

Figura 15 – Matriz de confusão do modelo treinado.



Fonte: elaborado pelo autor (2026).

A análise da matriz permite identificar padrões principais de comportamento do classificador, como a última linha da matriz, que apresenta valores expressivos para diversas classes, como veste de segurança, ferramentas e pessoas. Isso indica que, na maioria dos erros, o modelo não confundiu o objeto com outra classe, mas simplesmente falhou em detectá-lo, considerando-o parte do cenário. Essas confusões são críticas para um sistema de segurança, pois podem levar à não identificação de trabalhadores sem EPIs.

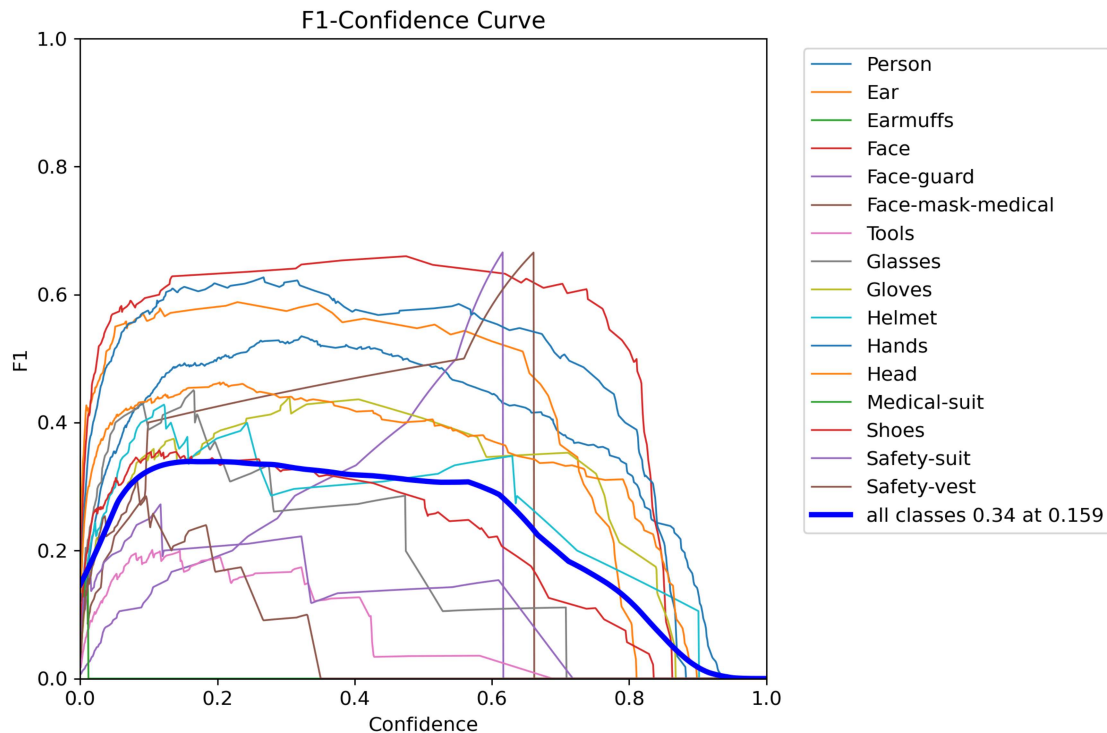
A presença de falsos positivos também é notável na coluna da direita, onde 37% dos recortes de fundo foram classificados erroneamente como classe pessoa, sugerindo que o modelo é sensível a formas que se assemelham à silhueta humana em ambientes complexos.

Os EPIs de pequenas dimensões, como óculos e protetores auriculares, apresentaram altas taxas de falso negativo, informando que o modelo ignora o objeto. Isso ocorre porque, em uma imagem de 640px, esses objetos tornam-se aglomerados de poucos pixels, perdendo texturas distintivas. Já a confusão entre 'luvas' e 'mãos' é esperada devido à similaridade morfológica, sugerindo que o modelo foca na forma e não necessariamente na cor ou textura do material.

A Curva F1-Score em função da Confiança apresentada na Figura 16 foi utilizada para

determinar o ponto de operação ideal do sistema e representa a média harmônica entre a precisão e a revocação, sendo a métrica mais adequada para equilibrar a necessidade de evitar falsos positivos, como alertas errados, e falsos negativos, como a falha na detecção de risco.

Figura 16 – Curva F1Score x Confiança do modelo treinado.



Fonte: elaborado pelo autor (2026).

A análise da curva média global indica que o desempenho máximo do modelo no conjunto de teste é atingido com um limiar de confiança de 0.159, resultando em um F1-Score de 0.34. Este resultado é considerado baixo em comparação a modelos treinados em cenários controlados, que geralmente variam entre 0.4 e 0.5. Isso indica que, para o modelo detectar corretamente os EPIs, o software de aplicação não deve ser configurado com filtros de confiança muito rigorosos para todas as classes, garantindo que o sistema maximize a detecção de não conformidades sem suprimir alertas válidos por meio de uma precisão ligeiramente menor.

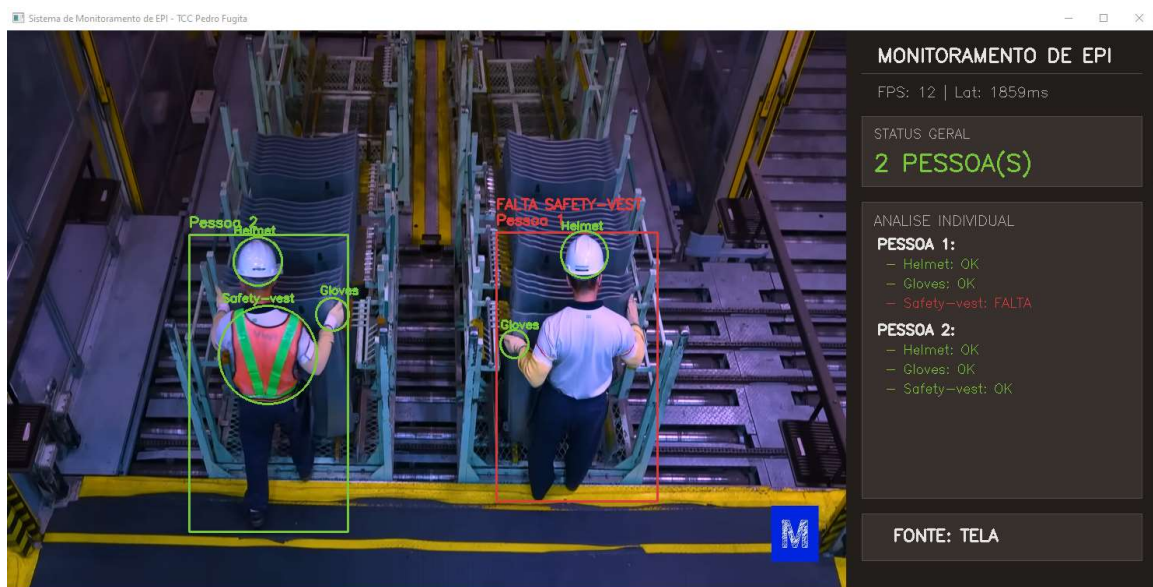
As evidências quantitativas obtidas por meio das métricas de validação são reforçadas pela análise visual apresentada na Figura 17.

4.3 Inferência por meio de Desenvolvimento de Software

Para fins de demonstração, o protótipo de software criado foi ajustado para realizar a análise diretamente da tela do computador, simulando a análise ao vivo em ambientes industriais reais por meio de vídeos disponíveis online.

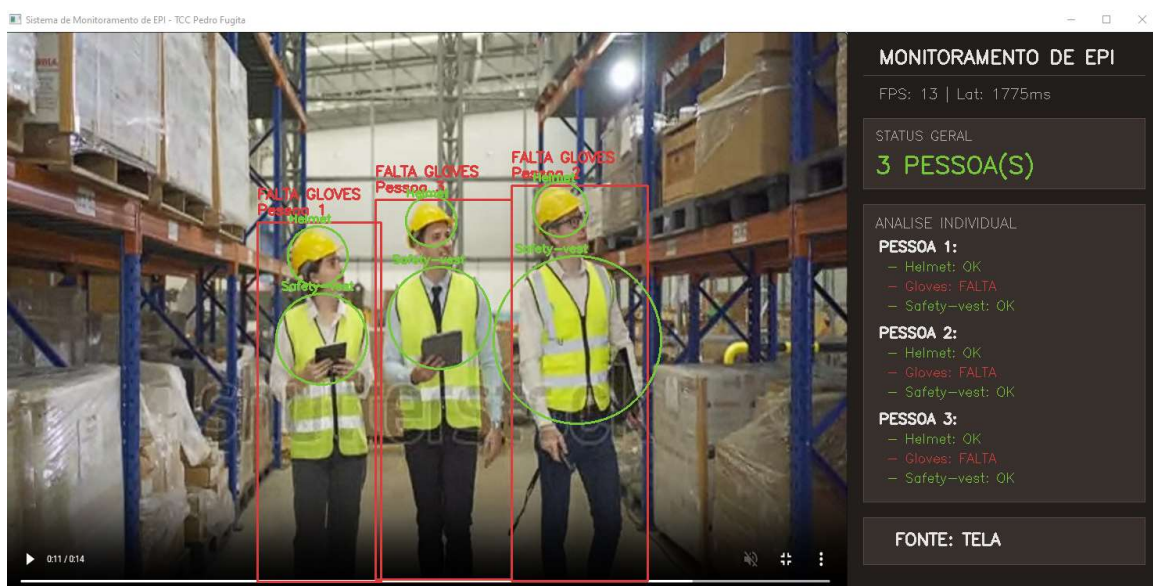
A Figura 18 e a Figura 19 ilustram a interface do aplicativo durante sua operação, utilizando a fonte de vídeo e o valor de precisão citados anteriormente e configurado para identificar capacetes, luvas e vestes de segurança.

Figura 18 – Interface do aplicativo desenvolvido em funcionamento (Cenário 1).



Fonte: elaborado pelo autor (2026)

Figura 19 – Interface do aplicativo desenvolvido em funcionamento (Cenário 2).



Fonte: elaborado pelo autor (2026)

Percebe-se que o sistema foi capaz de identificar corretamente o número de pessoas presentes na cena, bem como os EPIs utilizados ou não por cada indivíduo, e tudo é informado na lateral direita da interface, juntamente com um resumo do status de segurança e a fonte atual da imagem. Na primeira imagem, o trabalhador da esquerda está adequado às normas de uso de EPI em relação àquele local de trabalho e o da direita apresenta riscos, pois não foi identificado o uso da veste de segurança. Na segunda imagem, o software identificou a falta de luvas por parte dos colaboradores e que os outros equipamentos de segurança previstos para o local estão sendo utilizados. A presença de caixas delimitadoras coloridas ao redor dos objetos detectados somada às etiquetas de classe, confirma a eficácia do modelo treinado.

O modelo apresentou uma latência média de 1800 ms, decorrente do tempo de processamento que o modelo de visão computacional leva para identificar os EPIs quando aparecem na cena até a exibição dos resultados, gerando um vídeo de aproximadamente 12 FPS. Embora abaixo dos 30 FPS ideais, essa taxa é suficiente para o monitoramento industrial, onde a dinâmica de movimento de trabalhadores permite uma amostragem de segurança a cada 0,08 segundos sem perda de criticidade, validando a escolha da versão *nano* para hardware de baixo custo computacional.

5 CONSIDERAÇÕES FINAIS

Os resultados obtidos demonstraram que o modelo treinado é capaz de identificar classes relacionadas à segurança com desempenho compatível com aplicações em tempo real, evidenciando o potencial da abordagem para auxiliar na fiscalização do cumprimento das normas regulamentadoras. A interface desenvolvida contribuiu para a usabilidade da ferramenta, oferecendo recursos para monitoramento, emissão de alertas e registro de evidências digitais, o que reforça a viabilidade tecnológica da solução.

Entretanto, a análise dos resultados revelou limitações relevantes, como o desempenho do sistema mostrar-se sensível a fatores como variações de iluminação, oclusões parciais e diversidade de cenários operacionais. Adicionalmente, observou-se maior dificuldade na detecção de EPIs de pequenas dimensões, resultando em taxas mais elevadas de falsos negativos.

No contexto de aplicação real, tais limitações representam um risco operacional significativo, uma vez que o sistema pode não identificar situações de exposição a perigo, gerando uma falsa percepção de segurança. Dessa forma, destaca-se que a tecnologia desenvolvida deve ser utilizada como ferramenta complementar de apoio à gestão de segurança, não substituindo a supervisão humana nem as práticas consolidadas de prevenção de acidentes.

Com base nas lacunas identificadas, propõem-se como trabalhos futuros as seguintes diretrizes para aprimoramento da solução:

- Validação em ambiente industrial real: Implantação do sistema em condições reais de operação, considerando variáveis como cores reais dos EPIs usados em determinados locais e mudanças de iluminação, a fim de avaliar a robustez do modelo ao longo do tempo.
- Análise estatística dos resultados: Aplicação de métodos estatísticos para comparar o desempenho do modelo em diferentes cenários operacionais, turnos de trabalho e setores industriais, garantindo maior rigor científico na validação.
- Avaliação de arquiteturas alternativas: Investigação do desempenho de modelos mais recentes e robustos da família YOLO, com a construção de índices que relacionem precisão, tempo de inferência e consumo computacional, visando a otimização para aplicações em tempo real.
- Fusão de sensores: Exploração do uso combinado de câmeras RGB e sensores térmicos de infravermelho, com o objetivo de aumentar a confiabilidade da detecção em ambientes com baixa luminosidade ou presença de interferentes visuais.
- Aprimoramento do conjunto de dados: Expansão e diversificação do banco de imagens, com foco em amostras de EPIs de pequenas dimensões e em dados coletados diretamente nos ambientes industriais reais onde a tecnologia será aplicada, para melhorar a capacidade de generalização do modelo e reduzir as taxas de falsos negativos.

Por fim, conclui-se que a integração entre engenharia de segurança do trabalho e inteligência artificial apresenta elevado potencial para o desenvolvimento de soluções inovadoras voltadas à prevenção de acidentes. Contudo, para que a ferramenta proposta atinja níveis adequados de confiabilidade e autonomia, torna-se indispensável a continuidade do desenvolvimento, com ciclos iterativos de validação, ajustes e monitoramento em condições reais de uso.

REFERÊNCIAS

- AHMAD, H. M.; RAHIMI, A. **SH17: a dataset for human safety and personal protective equipment detection in manufacturing industry**. Windsor, ON, Canada: University of Windsor, 2024. Dataset. Disponível em: <https://arxiv.org/abs/2407.04590>. Acesso em: 10 jan. 2026.
- BIEDERMAN, I. Recognition-by-components: A theory of human image understanding. **Psychological Review**, v. 94, n. 2, p. 115–147, 1987. Disponível em: <https://doi.org/10.1037/0033-295X.94.2.115>. Acesso em: 10 jan. 2026.
- BRASIL. Ministério do Trabalho. **Norma Regulamentadora nº 6 (NR-6): Equipamento de Proteção Individual – EPI**. Brasília, DF, 2018.
- BÓIS, P. H. F. **Sistema automatizado de monitoramento de segurança com visão computacional**. 2026. Disponível em: <https://github.com/pedro-fugita/epi-monitoramento-epi>. Acesso em: 10 jan. 2026.
- CYGANEK, B. **Object detection and recognition in digital images**. Chichester, UK: John Wiley & Sons, Ltd., 2013.
- GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 3. ed. Upper Saddle River: Prentice Hall, 2010.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA: MIT Press, 2016.
- GRILL-SPECTOR, K.; MALACH, R. The human visual cortex. **Annual Review of Neuroscience**, v. 27, p. 649–677, 2004. Disponível em: <https://doi.org/10.1146/annurev.neuro.27.070203.144220>. Acesso em: 10 jan. 2026.
- GUNEY, E. et al. Yolo-based personal protective equipment monitoring system for workplace safety. **JITSI: Jurnal Ilmiah Teknologi Sistem Informasi**, v. 5, n. 2, p. 77–85, 2024. Disponível em: <https://doi.org/10.62527/jitsi.5.2.238>. Acesso em: 10 jan. 2026.
- GUPTA, V. K. et al. Automated ppe detection using yolov8 for real-time workplace safety monitoring. **International Journal of Innovative Science and Research Technology**, v. 10, n. 4, p. 2228–2233, 2025. Disponível em: <https://doi.org/10.38124/ijisrt%2F25apr1699>. Acesso em: 10 jan. 2026.
- ISAILOVIC, V. et al. The compliance of head-mounted industrial ppe by using deep learning object detectors. **Scientific Reports**, v. 12, n. 16347, 2022. Disponível em: <https://www.nature.com/articles/s41598-022-20282-9>. Acesso em: 10 jan. 2026.
- LogicLens Solutions. **LogicLens Solutions vs Legacy CCTV: Why Software-First AI Beats Hardware-Locked Analytics**. [S.l.], 2026.
- OPENCV. **OpenCV Documentation**. 2000. Disponível em: <https://docs.opencv.org>. Acesso em: 10 jan. 2026.
- ORGANIZAÇÃO INTERNACIONAL DO TRABALHO. **Safety and health at the heart of the future of work**. Geneva, 2019.

PYTORCH. **PyTorch Documentation**. 2016. Disponível em: <https://pytorch.org/docs/stable/index.html>. Acesso em: 10 jan. 2026.

REDMON, J. et al. You only look once: unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. Las Vegas, NV: IEEE, 2016. p. 779–788. Disponível em: <https://ieeexplore.ieee.org/document/7780460>. Acesso em: 10 jan. 2026.

SANTOS, G. G. F. **Desenvolvimento de um sistema de monitoramento em tempo real de utilização de EPI a partir de aprendizado de máquina**. 2023. Dissertação (Mestrado em Engenharia de Controle e Automação) — Universidade Federal do Pará, 2023.

SOARES, B. S.; OLIVEIRA, L. O. F.; ARAÚJO, E. D. **Avaliação da aplicação da inteligência artificial na detecção do uso de Equipamentos de Proteção Individual em ambientes de trabalho**. 2023. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) — Universidade Federal de Uberlândia, 2023.

STUMM, S. B. **Segurança do trabalho e ergonomia**. 1. ed. São Paulo: Contentus, 2020. E-book. Disponível em: <https://plataforma.bvirtual.com.br>. Acesso em: 10 jan. 2026.

ULTRALYTICS. **Ultralytics YOLO Documentation**. 2023. Disponível em: <https://docs.ultralytics.com>. Acesso em: 10 jan. 2026.

UNITED NATIONS. **The Ten Principles of the UN Global Compact**. New York, 2014.