UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
ENGENHARIA DE COMPUTAÇÃO

MATHEUS DA SILVA OLIVEIRA

DETECÇÃO DE EPI'S COM FERRAMENTA DE VISÃO COMPUTACIONAL

Tucuruí-PA
Novembro de 2024

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
ENGENHARIA DE COMPUTAÇÃO

MATHEUS DA SILVA OLIVEIRA

DETECÇÃO DE EPI'S COM FERRAMENTA DE VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso, apresentado à Faculdade de Engenharia de Computação, do Campus Universitário de Tucuruí, da Universidade Federal do Pará, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Daniel da Conceição Pinheiro

Tucuruí-PA
Novembro de 2024

Oliveira, Matheus
Detecção de EPI's com Ferramenta de Visão Computacional / Matheus da Silva Oliveira . – Tucuruí-PA, Novembro de 2024.
56 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel da Conceição Pinheiro

Monografia – UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
, Novembro de 2024.
1. EPI's 2. YOLO. 3. Python. 4. Machine Learning I. Título.

'Serum scientia est ea quae nos ad bonum appropinquat.'
" Conhecimento verdadeiro é aquele que nos aproxima do bem."
(Platão)

	UNIVERSIDADE FEDERAL DO PARÁ CAMPUS UNIVERSITÁRIO DE TUCURUÍ FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

TÍTULO: DETECÇÃO DE EPI'S COM FERRAMENTA DE VISÃO COMPUTACIONAL

DISCENTE: MATHEUS DA SILVA OLIVEIRA
MATRÍCULA: 201733840006

#	BANCA EXAMINADORA	CONDIÇÃO
1	*Prof. Dr. Daniel da Conceição Pinheiro*	*Orientador*
2	*Prof. Eng. Vigner Vieira dos Santos*	*Membro interno*
3	*Eng. Fellipe Augusto Santana Queiroz*	*Membro externo*

Data da Defesa: 27/11/2025	**Hora Início:** 10:04	**Hora Término:** 10:37

Trabalho Escrito (0 a 10 pontos por critério)	**Examinador 1**	**Examinador 2**	**Examinador 3**
Formatação	7,00	9,00	9
Linguagem (gramática e semântica)	7,00	9,00	9,00
Conteúdo técnico	7,00	8,00	8

Defesa Oral (0 a 10 pontos por critério)	**Examinador 1**	**Examinador 2**	**Examinador 3**
Sequência lógica de apresentação	9,00	10,00	10
Administração do tempo	10,00	10,00	10
Expressão oral	9,00	10,00	10
Domínio do tema	9,00	10,00	9,00

Média por examinador	8,29	9,43	9,29
Média Final	**9,00**		
Conceito Final	EXC		

Tucuruí-Pa, 28 de novembro de 2024.

gov.br Documento assinado digitalmente
DANIEL DA CONCEICAO PINHEIRO
Data: 28/11/2024 16:50:10-0300
Verifique em https://validar.iti.gov.br

Orientador

gov.br Documento assinado digitalmente
VIGNER VIEIRA DOS SANTOS
Data: 28/11/2024 16:38:07-0300
Verifique em https://validar.iti.gov.br

Membro interno

gov.br Documento assinado digitalmente
FELLIPE AUGUSTO SANTANA QUEIROZ
Data: 28/11/2024 16:44:33-0300
Verifique em https://validar.iti.gov.br

Membro externo

RESUMO

Este trabalho de conclusão de curso aborda o desenvolvimento de um sistema de reconhecimento de Equipamentos de Proteção Individual (EPIs) utilizando a técnica YOLO (*You Only Look Once*). A detecção automática de EPIs em ambientes de trabalho é fundamental para garantir a segurança dos trabalhadores e cumprir com as normas regulamentares de segurança. O uso de EPIs, como capacetes, luvas, coletes reflexivos e óculos de proteção, é essencial em diversos setores, especialmente na construção civil e na indústria. No entanto, a fiscalização manual do uso desses equipamentos pode ser ineficiente e suscetível a erros humanos.

O YOLO é um dos algoritmos mais avançados para detecção de objetos em tempo real, conhecido por sua alta velocidade e precisão. Este projeto envolveu a coleta e anotação de um conjunto de dados de imagens de trabalhadores equipados com diversos EPIs. As imagens foram cuidadosamente selecionadas para representar uma ampla gama de cenários e condições de iluminação, garantindo a robustez do modelo. O algoritmo YOLO foi então treinado com esses dados, utilizando técnicas de aprendizado profundo para ajustar seus parâmetros e otimizar seu desempenho. Durante o processo de treinamento, várias estratégias foram implementadas para melhorar a precisão do modelo. Após o treinamento, o modelo foi testado em um conjunto de dados de validação para avaliar sua capacidade de reconhecer corretamente os EPIs nas imagens. Os resultados foram analisados com base em métricas como precisão, *recall* e *F1-score*, demonstrando a eficácia do modelo desenvolvido.

Palavras chave:: EPI's, YOLO, Python, Machine Learning.

ABSTRACT

This end-of-course work deals with the development of a Personal Protective Equipment (PPE) recognition system using the YOLO (You Only Look Once) technique. The automatic detection of PPE in work environments is fundamental to guaranteeing worker safety and complying with regulatory safety standards. The use of PPE, such as helmets, gloves, reflective vests and goggles, is essential in many sectors, especially construction and industry. However, manually checking the use of this equipment can be inefficient and susceptible to human error.

YOLO is one of the most advanced algorithms for real-time object detection, known for its high speed and accuracy. This project involved collecting and annotating a dataset of images of workers wearing various PPE. The images were carefully selected to represent a wide range of scenarios and lighting conditions, ensuring the robustness of the model. The YOLO algorithm was then trained on this data, using deep learning techniques to adjust its parameters and optimize its performance. During the training process, various strategies were implemented to improve the model's accuracy. After training, the model was tested on a validation dataset to assess its ability to correctly recognize PPE in images. The results were analyzed based on metrics such as precision, recall and F1-score, demonstrating the effectiveness of the model developed.

Keywords: EPI's, YOLO, Python, Machine Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

YOLO	You Only Look Once, ou Olhe Apenas Uma Vez
ML	Machine Learning, ou aprendizado de máquina
CPU	Central Processing Unit, ou Unidade Central de Processamento
RAM	Random Access Memory, ou memória de acesso randômico
GPU	Graphics Processing Units, ou Unidade de processamento gráfico
IA	Inteligência Artificial

SUMÁRIO

1 INTRODUÇÃO

O uso de Equipamentos de Proteção Individual (EPIs) é fundamental para garantir a segurança dos trabalhadores em diversos ambientes industriais e de construção. A correta utilização desses equipamentos pode prevenir acidentes graves e proteger a saúde dos colaboradores. No entanto, a fiscalização manual e a constante verificação do uso adequado de EPIs são tarefas desafiadoras e propensas a erros humanos. Nesse contexto, a aplicação de técnicas de visão computacional para a detecção automática de EPIs surge como uma solução promissora, capaz de aumentar a eficiência e a precisão desse processo de fiscalização. (KUMAR; SAHOO, 2019) (PRADHAN; SAMANTARAY, 2021)

No Brasil, as Normas Regulamentadoras (NRs) do Ministério do Trabalho e Emprego estabelecem os requisitos mínimos para a implementação de medidas de controle e sistemas preventivos de segurança e saúde nos ambientes de trabalho. A NR 6, em específico, trata dos Equipamentos de Proteção Individual. Esta norma define EPI como "todo dispositivo ou produto, de uso individual utilizado pelo trabalhador, destinado à proteção de riscos suscetíveis de ameaçar a segurança e a saúde no trabalho". (EMPREGO, 1978)

De acordo com a NR 6, os EPIs devem ser fornecidos gratuitamente pelo empregador, em perfeito estado de conservação e funcionamento, e a sua utilização deve ser rigorosamente supervisionada. Além disso, outras normas, como a NR 18 (Condições e Meio Ambiente de Trabalho na Indústria da Construção) e a NR 35 (Trabalho em Altura), também fazem menção específica ao uso de EPIs, detalhando as condições e exigências para cada tipo de equipamento. (EMPREGO, 1995) (EMPREGO, 2012)

1.1 Equipamentos de Proteção Individual

Os principais EPIs utilizados em ambientes industriais e de construção incluem:

- **Capacete:** Destinado à proteção da cabeça contra impactos e perfurações provenientes de queda de objetos. Também pode proteger contra choques elétricos e queimaduras em casos específicos.
- **Colete:** Geralmente utilizado para aumentar a visibilidade do trabalhador, especialmente em locais com tráfego de veículos ou máquinas. Pode ser equipado com faixas refletivas.
- **Máscara:** Utilizada para proteger as vias respiratórias contra poeiras, fumos, névoas, gases e vapores tóxicos. Existem diversos tipos de máscaras, incluindo descartáveis, semifaciais e respiradores de ar abastecido.
- **Óculos:** Protetores oculares são essenciais para proteger os olhos contra impactos, partículas volantes, poeiras e radiações nocivas, como luz ultravioleta ou infravermelha.

- **Abafador:** Também conhecido como protetor auricular, é utilizado para proteger a audição em ambientes com níveis elevados de ruído, prevenindo a perda auditiva induzida por ruído (PAIR).
- **Luvas:** As luvas de proteção são utilizadas para proteger as mãos contra agentes abrasivos, cortantes, perfurantes, químicos e biológicos. Existem diferentes tipos de luvas para atender a riscos específicos.
- **Botas:** Calçados de segurança são essenciais para proteger os pés contra impactos, perfurações, produtos químicos, escorregões e outros riscos. Podem incluir biqueira de aço e solado antiderrapante.

Figura 1 – Exemplo de EPI's.

Fonte: (Paromed, 2024)

Nos últimos anos, a área de visão computacional tem avançado significativamente, impulsionada por redes neurais convolucionais (CNNs) e técnicas de *deep learning*. Entre as diversas abordagens, os modelos YOLO (*You Only Look Once*) se destacam por sua capacidade

de realizar detecção de objetos em tempo real com alta precisão. O YOLOv8, a versão mais recente dessa família de modelos, traz melhorias em termos de arquitetura e desempenho, oferecendo diferentes distribuições (*nano, small, medium, large e extra large*) que variam em complexidade e eficiência. (REDMON; FARHADI, 2018)

A literatura existente já demonstra o potencial de modelos de deep learning para a detecção de EPIs. Estudos como o de Redmon e Farhadi (2016) introduziram o YOLO, destacando sua eficiência em tarefas de detecção de objetos em tempo real. (REDMON; FARHADI, 2016) Versões subsequentes, como o YOLOv4 (Bochkovskiy et al., 2020), trouxeram melhorias significativas, sendo amplamente adotadas em diversos cenários de segurança industrial. (BOCHKOVSKIY; WANG; LIAO, 2020) No entanto, a aplicação específica das distribuições do YOLOv8 para a detecção de EPIs ainda é um campo a ser explorado, justificando a relevância e a inovação deste trabalho. Para a realização deste estudo, serão utilizados datasets específicos contendo imagens de trabalhadores em diferentes ambientes industriais, com e sem o uso de EPIs. A metodologia envolverá o treinamento dos modelos YOLOv8n, YOLOv8s, YOLOv8m e YOLOv8l seguido de uma avaliação comparativa dos resultados obtidos. As métricas de desempenho incluirão a precisão (precision), sensibilidade (recall), F1-score, mean Average Precision (mAP) e o tempo de processamento para cada modelo. Espera-se que os resultados deste trabalho contribuam para o avanço das tecnologias de segurança no trabalho, oferecendo uma ferramenta robusta e eficiente para a detecção automática de EPIs. Além disso, a comparação detalhada entre as distribuições do YOLOv8 fornecerá insights valiosos sobre a adequação de cada modelo para diferentes aplicações, auxiliando na escolha do modelo mais apropriado para implementações práticas.

1.2 Objetivo Geral

Desenvolver e avaliar um modelo de detecção de Equipamentos de Proteção Individual (EPIs) utilizando quatro distribuições do YOLOv8 (nano, small, medium e large), comparando seu desempenho em termos de precisão, taxa de falsos positivos, tempo de processamento e capacidade de generalização, para aplicações em tempo real em ambientes industriais.

1.3 Objetivos Específicos:

- **1.** Treinar e avaliar o desempenho das distribuições nano, small, medium e large do YOLOv8 na detecção de EPIs, considerando métricas como precisão, taxa de falsos positivos e tempo de processamento.
- **2.** Desenvolver um código em Python para implementar e testar os modelos treinados em tempo real, validando sua eficiência e verificando a adequação para aplicações práticas

em ambientes industriais.

- **3.** Identificar as vantagens e limitações de cada distribuição do YOLOv8 em diferentes cenários, destacando a adequação de cada modelo em função de restrições computacionais e requisitos de aplicação.

2 REFERENCIAL TEÓRICO

O referencial teórico é um componente essencial de qualquer pesquisa acadêmica, fornecendo a base conceitual e contextual para o estudo. Este capítulo detalha as principais teorias e conceitos relacionados às tarefas de processamento de imagens baseadas em visão computacional, incluindo classificação de imagens, localização de objetos, redes neurais convolucionais (CNNs) e a série de modelos YOLO (*You Only Look Once*). Também serão abordadas as aplicações práticas destas técnicas na detecção de Equipamentos de Proteção Individual (EPIs) em ambientes industriais.

2.0.1 Inteligência Artificial:

A Inteligência Artificial (IA) é um campo da ciência da computação dedicado ao estudo e desenvolvimento de sistemas capazes de simular comportamentos inteligentes. Seu objetivo principal é promover o entendimento e a aplicação eficaz de técnicas que resolvam problemas complexos, incluindo planejamento e comunicação em diversos contextos práticos (LUGER, 2013). Embora aborde uma ampla gama de problemas, o autor destaca oito características fundamentais que são comuns às diferentes vertentes da área:

A utilização de computadores para realizar processos como raciocínio, reconhecimento de padrões, aprendizado e outras formas de inferência.

A abordagem de problemas que não podem ser resolvidos exclusivamente por algoritmos tradicionais, demandando, assim, a aplicação de buscas heurísticas como estratégia para encontrar soluções.

A capacidade de lidar com informações imprecisas, incompletas ou mal definidas, utilizando representações formais que permitam mitigar essas limitações.

A habilidade de raciocinar com base em aspectos qualitativos importantes de uma situação, sem depender exclusivamente de dados quantitativos.

O enfoque em questões que consideram tanto o significado semântico quanto a estrutura sintática de informações.

A produção de respostas que, embora não sejam exatas ou ótimas, atendem aos requisitos de forma "suficiente", especialmente em cenários onde soluções exatas seriam inviáveis ou muito custosas.

A utilização de grandes volumes de conhecimento específico de um domínio para resolver problemas, característica essencial dos sistemas especialistas.

A aplicação de meta-conhecimento, ou seja, o conhecimento sobre o próprio processo de resolução de problemas, para aprimorar o controle das estratégias utilizadas. Embora esse

tópico ainda seja desafiador e explorado por poucos sistemas, ele está emergindo como uma área central de pesquisa.

2.0.2 Machine Learning

O aprendizado de máquina (Machine Learning) busca aprimorar um critério de desempenho utilizando dados de exemplo ou experiências anteriores. Esse processo envolve um modelo com parâmetros definidos que são ajustados por meio de um software durante a fase de treinamento, utilizando os dados disponíveis. O modelo pode ser utilizado de forma preditiva, para realizar projeções futuras, de maneira descritiva, para extrair insights dos dados, ou até mesmo para ambos os propósitos.

A construção de modelos no aprendizado de máquina é fortemente influenciada pela teoria estatística, pois o objetivo central é realizar inferências a partir de amostras de dados. A ciência da computação desempenha um papel crucial em dois momentos principais: durante o treinamento, ao desenvolver algoritmos eficientes para resolver problemas de otimização e lidar com grandes volumes de dados, e na etapa de inferência, ao garantir que a representação e a solução algorítmica do modelo sejam eficazes.

Em muitas aplicações, a eficiência dos algoritmos utilizados, tanto no treinamento quanto na inferência, é um aspecto fundamental. Isso inclui a complexidade em termos de espaço de armazenamento e tempo de processamento, que pode ser tão relevante quanto a precisão do modelo na realização de previsões. A combinação de eficiência computacional e precisão é, portanto, essencial para o sucesso do aprendizado de máquina em contextos práticos. (ALPAYDIN, 2020)

2.0.3 Deep Learning

O Deep Learning, ou aprendizado profundo, é uma área da Inteligência Artificial focada no desenvolvimento de redes neurais complexas, capazes de realizar análises detalhadas e tomar decisões baseadas em dados estruturados. Ele se destaca por sua eficácia em lidar com informações complexas e por sua aplicação em situações onde grandes volumes de dados estão disponíveis. Essa abordagem é amplamente utilizada em tecnologias modernas, sendo integrada em plataformas e dispositivos que fazem parte do cotidiano.

Empresas de tecnologia têm implementado o aprendizado profundo em diversas funções. Por exemplo, plataformas de redes sociais analisam mensagens de texto para identificar padrões e compreender contextos. Já ferramentas de busca e tradução automática, como as oferecidas pelo Google e Bing, empregam essas redes para otimizar resultados e aprimorar serviços de tradução. No campo dos dispositivos móveis, o aprendizado profundo é amplamente utilizado no reconhecimento de voz e facial, bem como em outras funções inteligentes.

Além do setor de consumo, o aprendizado profundo tem se mostrado indispensável em áreas como a saúde. Ele é utilizado para a análise de imagens médicas, como radiografias e ressonâncias magnéticas, ajudando no diagnóstico de condições de saúde. No setor automotivo, a tecnologia é fundamental no desenvolvimento de veículos autônomos, sendo aplicada em funções como mapeamento, planejamento de trajetos, percepção do ambiente ao redor e monitoramento do estado do motorista.

A evolução do aprendizado profundo deriva de avanços na inteligência artificial e no aprendizado de máquina. Este último é um campo que se concentra em criar sistemas capazes de aprender a partir de exemplos. Por meio de algoritmos especializados, as máquinas podem identificar padrões e deduzir funções de conjuntos de dados previamente fornecidos, gerando soluções para problemas variados.

Os algoritmos de aprendizado de máquina são projetados para analisar informações contidas em conjuntos de dados, que são tabelas organizadas onde as linhas representam exemplos específicos e as colunas detalham características ou atributos desses exemplos. Esses algoritmos processam os dados, identificando padrões e criando modelos preditivos que podem ser aplicados em novos contextos.

Outro conceito-chave é o de funções, que no aprendizado de máquina representam mapeamentos determinísticos entre entradas e saídas. Isso significa que, dado um mesmo conjunto de dados de entrada, uma função sempre produzirá os mesmos resultados de saída, conferindo previsibilidade e consistência aos modelos desenvolvidos.

No cotidiano, exemplos práticos de aprendizado profundo incluem a análise automática de imagens, identificação de voz em assistentes virtuais e rastreamento de movimentos em sistemas de segurança. No entanto, a complexidade dessa tecnologia exige não apenas dados em grande escala, mas também recursos computacionais robustos, especialmente em aplicações mais avançadas.

A capacidade do aprendizado profundo de transformar grandes volumes de dados em insights práticos tem impulsionado sua adoção em áreas como negócios, segurança e educação. Essa tecnologia permite a automação de tarefas complexas, proporcionando maior eficiência e reduzindo custos em diversas indústrias.

Por fim, o aprendizado profundo continua a evoluir, possibilitando avanços ainda mais surpreendentes no futuro. Ele se posiciona como uma das ferramentas mais promissoras para a resolução de problemas complexos, com potencial de revolucionar setores inteiros e melhorar significativamente a qualidade de vida em diferentes contextos.

2.0.4 Redes Neurais

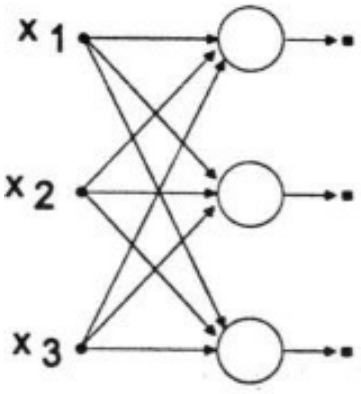
De acordo com Braga, Carvalho e Ludermir (2000), Redes Neurais Artificiais (RNAs) são compostas por unidades de processamento simples, conhecidas como nodos, projetadas para realizar cálculos matemáticos. Essas unidades estão organizadas em uma ou mais camadas interconectadas por meio de diversas conexões. Cada uma dessas conexões possui um peso associado, responsável por armazenar o conhecimento do modelo e encaminhar as entradas para os neurônios da rede. Cada neurônio, por sua vez, utiliza uma função de ativação que determina se o sinal será propagado ou inibido.

Durante o treinamento das RNAs, os pesos das conexões são ajustados para decidir quais neurônios serão ativados nas camadas seguintes, simulando o funcionamento das sinapses no sistema nervoso animal. A função de ativação tem a responsabilidade de regular a intensidade da ativação e pode assumir diferentes formas, dependendo do objetivo do modelo. A escolha adequada dessa função impacta diretamente no comportamento das saídas de cada neurônio.

Para resolver problemas utilizando RNAs, a rede passa por um processo inicial de aprendizado, no qual é exposta a uma série de exemplos. Com base nesses exemplos, a rede é capaz de identificar e extrair automaticamente as características necessárias para representar as informações fornecidas. Essas características extraídas são posteriormente empregadas na geração de soluções para os problemas apresentados.

Haykin (2007) reforça que as RNAs possuem uma capacidade natural de armazenar conhecimento adquirido por meio de experiências e torná-lo acessível para uso posterior. Assim como o cérebro humano, as redes neurais aprendem por meio de um processo adaptativo, e as forças de conexão entre seus neurônios atuam como uma forma de armazenar o conhecimento acumulado ao longo do treinamento.

Figura 2 – Exemplo de Classificação.

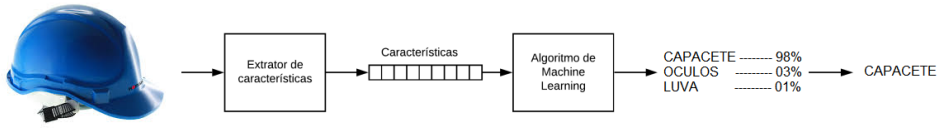


Fonte: Adaptado de Braga, Carvalho e Ludermir (2000)

2.1 Classificação de Imagem

A classificação de imagens é o processo de atribuir uma etiqueta ou categoria a uma imagem com base no conteúdo visual. Essas categorias podem ser pré-definidas ou aprendidas a partir dos próprios dados com as redes neurais. Este processo envolve a extração de características visuais relevantes das imagens e a aplicação de algoritmos de aprendizado de máquina para classificá-las em categorias predefinidas. A classificação de imagens é utilizada em uma ampla gama de aplicações, desde a identificação de produtos em sistemas de comércio eletrônico até a detecção de doenças em imagens médicas. (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) A figura 3 ilustra esse processo.

Figura 3 – Exemplo de Classificação.



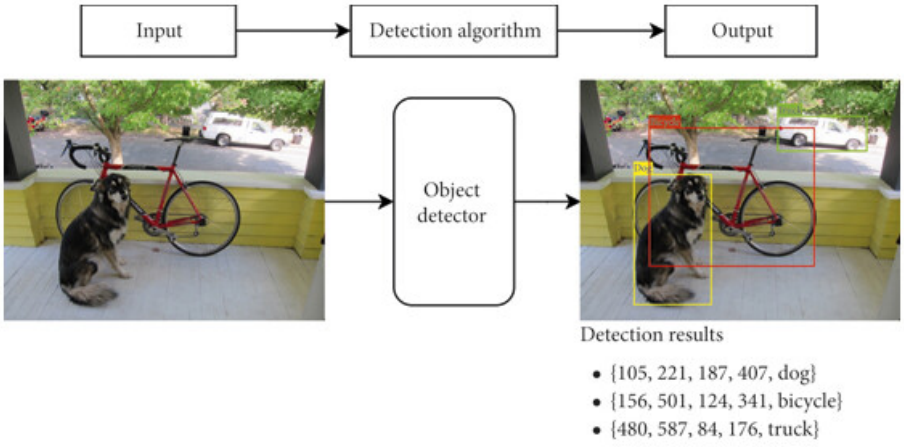
Fonte: Próprio Autor.

2.2 Localização de Objetos

O objetivo desta tarefa é segmentar o objeto desejado dentro da imagem. Para isso, é necessário dispor de imagens contendo os objetos e suas respectivas máscaras, onde apenas os objetos desejados estão destacados, para treinar o algoritmo de classificação. Assim como na seção anterior, um extrator de características é utilizado para converter a imagem em um vetor de características. Por fim, o algoritmo de aprendizado de máquina classifica todos os pixels da imagem na classe correspondente, gerando uma máscara da imagem original. (LONG; SHELHAMER; DARRELL, 2015) Como podemos ver na figura 4, são geradas caixas delimitadoras, essas caixas contam com os números de suas coordenadas e, em seguida, a classe a qual essa caixa delimitadora pertence.

Vários algoritmos têm sido desenvolvidos para a localização de objetos. Entre eles, os mais proeminentes são as Redes Neurais Convolucionais baseadas em regiões (R-CNN), que dividem a imagem em várias regiões e aplicam CNNs em cada região para detectar objetos (GIRSHICK et al., 2014), e os detectores de uma única passada, como o SSD (*Single Shot MultiBox Detector*) (LIU et al., 2016) e o YOLO (*You Only Look Once*) (REDMON; FARHADI, 2016), que realizam a detecção de objetos em uma única etapa de inferência, tornando o processo extremamente rápido e eficiente.

Figura 4 – Exemplo de Localização de Objetos.



Fonte: Hassan Yasser Khalil (2020).

2.3 Redes Neurais Convolucionais (CNNs)

As Redes Neurais Convolucionais (CNNs) constituem a espinha dorsal das modernas técnicas de visão computacional. Projetadas especificamente para lidar com dados de imagem, as CNNs são altamente eficazes na extração de características visuais relevantes e na realização de tarefas como a classificação e a detecção de objetos. (LECUN et al., 1998)

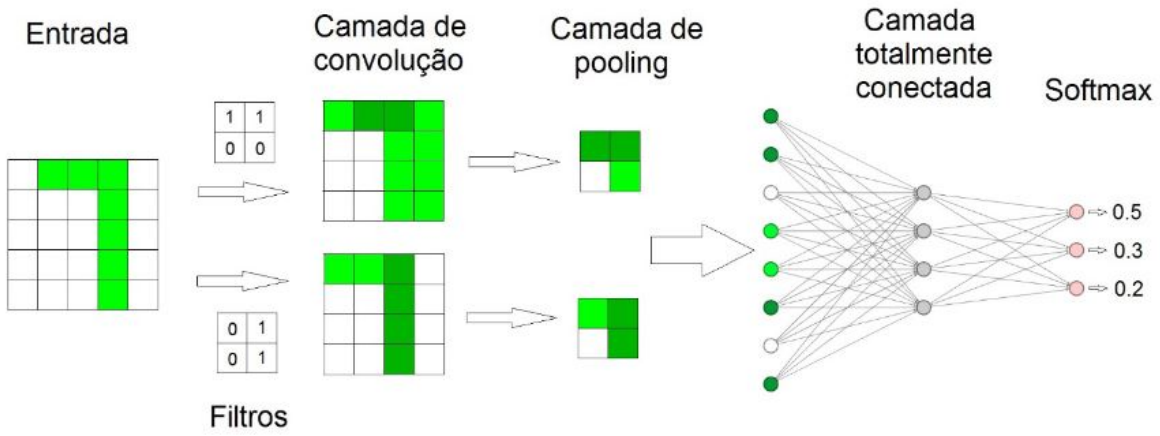
2.3.1 Estrutura das CNNs

A estrutura de uma CNN é composta por várias camadas, cada uma desempenhando uma função específica no processamento da imagem. As camadas convolucionais são responsáveis pela aplicação de filtros (*kernels*) à imagem de entrada para extrair características importantes. (HE et al., 2016) Estes filtros atuam como detectores de padrões visuais, como bordas e texturas. As camadas de *pooling* (agrupar), como *max-pooling e average-pooling*, reduzem a dimensionalidade das características extraídas, preservando as informações mais importantes e reduzindo o custo computacional. (ZEILER; FERGUS, 2014) A etapa de *flattening* transforma as características bidimensionais em um vetor unidimensional, que pode ser processado por camadas densamente conectadas (*fully connected*). (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

O funcionamento de uma CNN pode ser descrito através das seguintes etapas principais:

- **Entrada:** A entrada de uma CNN é tipicamente uma imagem. Cada imagem pode ser representada como uma matriz de pixels, onde cada pixel possui valores de intensidade de cor.
- **Convolução:** A camada de convolução é a primeira camada de uma CNN. Ela aplica

Figura 5 – Exemplo de arquitetura CNN.



Fonte: (EBERMAM; KROHLING, 2018)

filtros (kernels) à imagem de entrada para produzir mapas de características. Cada filtro é uma pequena matriz que desliza sobre a imagem de entrada e realiza uma operação de convolução. (SIMONYAN; ZISSERMAN, 2014)

- **Operação de Convolução:** A operação de convolução envolve a multiplicação dos valores dos pixels na imagem pela matriz do filtro e a soma dos resultados. Esse processo é repetido para cada posição do filtro na imagem, gerando um mapa de características (feature map). (GLOROT; BORDES; BENGIO, 2011)

- **Agregação (Pooling):** A agregação, ou pooling, é uma operação de redução de dimensionalidade que combina as características extraídas em regiões locais. A técnica mais comum é o max-pooling, que substitui uma região do mapa de características pelo valor máximo naquela região. Isso reduz a resolução espacial das características, diminuindo a quantidade de parâmetros e o custo computacional, além de tornar a rede mais resistente a variações de posição e escala. (LECUN; BENGIO; HINTON, 2015)

- **Camadas Totalmente Conectadas (Fully Connected)** As camadas totalmente conectadas são similares às camadas de redes neurais tradicionais, onde cada neurônio está conectado a todos os neurônios da camada anterior. Estas camadas combinam as características extraídas para tomar decisões finais sobre a classificação da imagem. (GOODFELLOW; BENGIO; COURVILLE, 2016)

- **Função de Ativação Softmax** A última camada da CNN geralmente utiliza a função de ativação softmax, que produz uma distribuição de probabilidade sobre as classes possíveis. Cada valor de saída da função softmax representa a probabilidade da imagem pertencer a uma determinada classe. A imagem pode ser classificada como as categorias predefinidas juntamente com as probabilidades associadas a cada uma. (BISHOP, 2006)

A estrutura de uma CNN pode ser resumida nas seguintes etapas principais:

- **Entrada:** Imagem de entrada.
- **Convolução:** Aplicação de filtros para gerar mapas de características.
- ***Pooling*:** Redução de dimensionalidade e resistência a variações.
- **Camadas Totalmente Conectadas** Combinação das características extraídas.
- ***Softmax*** Geração de probabilidades de classe.

2.4 YOLO

You Only Look Once, ou YOLO, que também é uma CNN, é uma família de modelos de detecção de objetos que revolucionou o campo da visão computacional pela sua capacidade de realizar detecções em tempo real com alta precisão. Desenvolvida por Joseph Redmon e Ali Farhadi, a primeira versão foi apresentada em 2015. Desde então, a YOLO passou por várias atualizações, cada uma trazendo melhorias significativas em termos de precisão, velocidade e eficiência. Vamos explorar brevemente a evolução das versões YOLO, da YOLOv5 até a utilizada neste projeto, a YOLOv8.

2.4.1 YOLOv5 (2020)

Embora não oficialmente parte da série original desenvolvida por Redmon, a YOLOv5 foi lançada pela *Ultralytics* e rapidamente ganhou popularidade devido à sua facilidade de uso e implementações em *frameworks* populares como *PyTorch*. A YOLOv5 introduziu melhorias práticas em termos de velocidade de treinamento e inferência, além de maior precisão. (ULTRALYTICS, 2020)

2.4.2 YOLOv6 (2022)

A YOLOv6, também desenvolvida pela *Ultralytics*, continuou a trajetória de melhorias incrementais, focando em reduzir a latência e aumentar a precisão. Esta versão apresentou melhorias na arquitetura e técnicas de treinamento que resultaram em melhor desempenho em *benchmarks* padrão. (ULTRALYTICS, 2022)

2.4.3 YOLOv7 (2022)

Lançada pela equipe original da YOLO, a YOLOv7 introduziu novas técnicas de arquitetura e treinamento, como a introdução de módulos eficientes e uma melhor integração de redes de *feature pyramids*. A YOLOv7 foi projetada para maximizar a precisão sem sacrificar a velocidade. (WANG; BOCHKOVSKIY; LIAO, 2022)

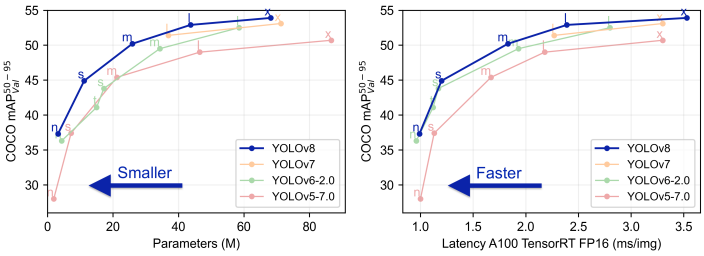
2.4.4 YOLOv8 (2023)

A YOLOv8, também desenvolvida pela Ultralytics, representa uma evolução significativa em relação às versões anteriores, com aprimoramentos na arquitetura e no fluxo de treinamento que visam aumentar ainda mais a precisão e a eficiência. Além de oferecer uma implementação ainda mais simplificada e otimizada para frameworks modernos como PyTorch, a YOLOv8 inclui novas técnicas de ajuste de hiperparâmetros e melhor compatibilidade com hardware acelerado, como GPUs. Esta versão foi projetada para suportar tarefas de detecção de objetos, segmentação e classificação com maior precisão e tempos de inferência reduzidos, mantendo-se uma das escolhas mais populares para aplicações de visão computacional em tempo real. (ULTRALYTICS, 2024).

2.5 Comparação de Desempenho das Versões do YOLO

Esta secção terá como objetivo analisar as versões do yolo desenvolvidas pela ultralytics, ou seja, da versão 5 até a versão 8, a fim de analisar qual tem o melhor custo/benefício para decidirmos qual será utilizada.

Figura 6 – Comparação das Versões YOLO.



Fonte: Ultralytics. (n.d.). YOLOv8 Comparison Plots.

A figura 6 compara o desempenho das versões YOLOv8, YOLOv7, YOLOv6-2.0 e YOLOv5-7.0 em termos de precisão (COCO mAP) e latência (ms/img) no processamento de imagens. Vamos analisar detalhadamente cada aspecto representado no gráfico.

Eixo Horizontal (Parâmetros e Latência): O gráfico à esquerda usa o eixo horizontal para representar o número de parâmetros (em milhões) em cada versão da rede. Um maior número de parâmetros geralmente indica uma rede mais complexa, mas não necessariamente uma rede mais eficiente. O gráfico à direita usa o eixo horizontal para mostrar a latência medida em milissegundos por imagem (ms/img), que reflete o tempo que a rede leva para processar cada imagem. Redes com menor latência são mais rápidas.

Eixo Vertical (COCO mAP): O eixo vertical em ambos os gráficos representa a precisão da rede, medida como COCO mAP (*mean Average Precision*) no intervalo de 50-95%. O COCO

mAP é uma métrica padrão usada para avaliar a precisão dos modelos de detecção de objetos em diferentes limiares de interseção sobre união (IoU).

YOLOv8 (linha azul): A YOLOv8 apresenta a melhor precisão (mAP) comparada às outras versões, com um menor número de parâmetros e menor latência. Isso indica que a YOLOv8 é não apenas mais precisa, mas também mais eficiente em termos de processamento. As versões menores, como YOLOv8n (*nano*) e YOLOv8s (*small*), conseguem atingir precisões competitivas com menor complexidade e tempo de processamento.

YOLOv7 (linha laranja): A YOLOv7 tem uma precisão boa, mas ligeiramente inferior à YOLOv8. No entanto, sua latência é maior, o que significa que leva mais tempo para processar cada imagem em comparação com a YOLOv8.

YOLOv6-2.0 (linha verde): A YOLOv6-2.0 mostra um equilíbrio entre precisão e latência, sendo melhor que a YOLOv5, mas inferior à YOLOv7 e YOLOv8 em termos de precisão. Sua latência é um pouco maior do que a da YOLOv8, mas comparável à YOLOv7.

YOLOv5-7.0 (linha vermelha): A YOLOv5-7.0 apresenta a menor precisão entre todas as versões comparadas e também a maior latência. Isso demonstra que as melhorias introduzidas nas versões subsequentes (YOLOv6, YOLOv7 e YOLOv8) resultaram em avanços significativos tanto em precisão quanto em velocidade. A YOLOv8 se destaca como a versão mais avançada e eficiente da série YOLO, oferecendo melhor desempenho em termos de precisão (mAP) e menor latência em relação às suas predecessoras. Esta eficiência é particularmente importante em aplicações em tempo real, onde a velocidade de processamento é crucial. A adoção de um mecanismo de detecção sem âncora e melhorias na arquitetura *backbone* e *neck* contribuem para o desempenho superior da YOLOv8, sendo assim, a versão 8 foi a escolhida para trabalharmos.

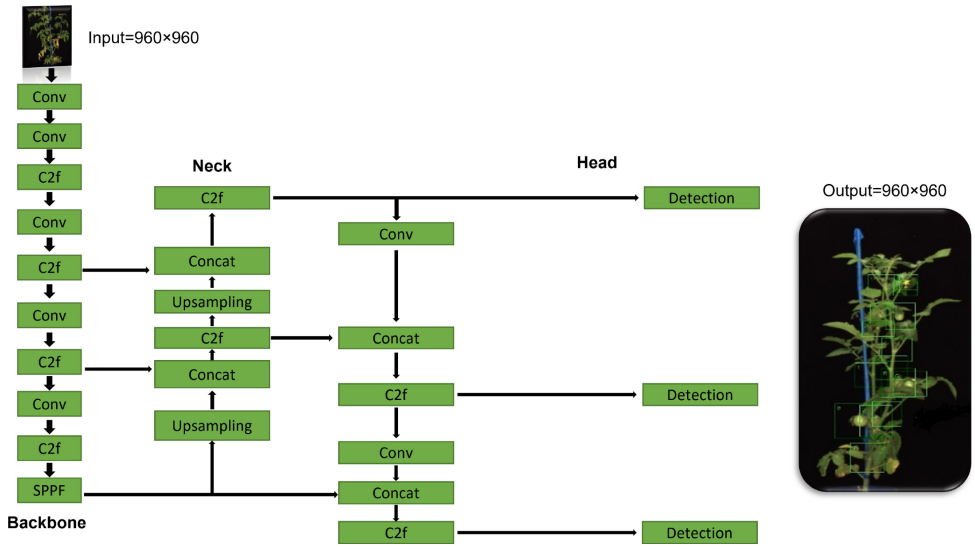
2.6 Arquitetura YOLOv8

A arquitetura da YOLOv8 é composta por três partes principais: *backbone, neck* e *head*. A imagem abaixo ilustra a estrutura completa da rede.

- ***Backbone****:* A camada *backbone* da YOLOv8 é responsável pela extração de características da imagem de entrada. Ela é baseada na CSPDarknet53, utilizada também na YOLOv4, mas com algumas modificações (BOCHKOVSKIY; WANG; LIAO, 2020). O módulo C2f (*Cross Stage Partial Networks* ou redes parciais entre estágios) foi introduzido para combinar características de alto nível com informações contextuais, melhorando a precisão da detecção (WANG et al., 2020). O fluxo de dados começa com a imagem de entrada, que passa por várias camadas de convolução e *pooling*. Estas camadas são responsáveis por reduzir a dimensionalidade da imagem e extrair características importantes.

- ***Neck****:* A camada neck da YOLOv8 tem como função refinar as características extraídas

Figura 7 – Arquitetura YOLOv8.



Fonte: (BARLYBAYEV et al., 2024)

pelo *backbone*. Esta camada utiliza o módulo C2f para melhorar a precisão, combinando características de diferentes níveis da rede (JOCHER et al., 2023). A estrutura do *neck* inclui operações de *upsampling* (aumento da resolução) e concatenação, que ajudam a preservar informações detalhadas da imagem enquanto combinam dados de diferentes resoluções. Isso permite que a rede capture informações detalhadas e contextuais, melhorando a capacidade de detectar objetos de diferentes tamanhos.

- ***Head****:* A camada *head* é onde ocorre a predição das classes dos objetos e sua localização. A YOLOv8 difere das versões anteriores da YOLO ao utilizar um mecanismo de detecção sem âncora (*anchor-free detection*). Em vez de prever deslocamentos em relação a âncoras predefinidas, a YOLOv8 prevê a localização diretamente no centro do objeto (REIS; OLIVEIRA; SILVA, 2023). Esta abordagem simplifica a arquitetura da rede e melhora tanto a velocidade quanto a precisão da detecção. Para a definição da probabilidade de uma *bounding box* conter um objeto, é utilizada a função de ativação sigmoid. Para determinar a qual classe o objeto pertence, a função *softmax* é empregada (Reis et al., 2023).

Na figura 7 acima, podemos observar o fluxo completo desde a imagem de entrada até a predição final. A imagem de entrada é processada por várias camadas de convolução (Conv) no *backbone*, onde são aplicadas operações de *pooling* (P) e o módulo C2f para extração de características. Essas características são então refinadas no neck, onde operações de *upsampling* (aumento da resolução) e concatenação (*Concat*) ajudam a combinar informações de diferentes resoluções. Finalmente, no *head*, as previsões são feitas através de camadas de convolução adicionais, sem a necessidade de caixas âncoras, utilizando funções de ativação para determinar a localização e a classe dos objetos detectados.

3 REVISÃO LITERÁRIA

A detecção de Equipamentos de Proteção Individual (EPIs) utilizando técnicas de visão computacional tem atraído a atenção de muitos pesquisadores devido à sua relevância para a segurança no trabalho. Diversos estudos demonstraram a eficácia de modelos de deep learning, em especial as redes neurais convolucionais, para esta finalidade. A seguir, são apresentados alguns dos principais trabalhos relacionados a este tema.

No artigo ***A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*** (LI et al., 2021) , os autores realizam um estudo abrangente sobre redes neurais convolucionais (CNNs), explorando suas diversas aplicações e o potencial futuro dessa tecnologia. O foco principal do trabalho foi o desenvolvimento de um sistema baseado em CNNs para detectar Equipamentos de Proteção Individual (EPIs) em vídeos de segurança, com o objetivo de promover ambientes de trabalho mais seguros, automatizando a detecção de elementos críticos como capacetes e coletes.

Os autores destacam que as CNNs, conhecidas por sua capacidade de extração automática de características de imagens e vídeos, mostraram-se altamente eficazes para tarefas de visão computacional em tempo real. No contexto da segurança no trabalho, a abordagem demonstrou uma precisão significativa na identificação de EPIs, especialmente nas classes de capacetes e coletes, que são essenciais para garantir a integridade física dos trabalhadores em áreas de risco. A alta precisão alcançada no estudo reforça a viabilidade das CNNs em aplicações críticas, onde a detecção rápida e precisa é necessária para evitar acidentes e garantir o cumprimento das normas de segurança.

Além disso, o artigo discute outras aplicações das CNNs, não apenas na área de segurança, mas também em áreas como reconhecimento facial, diagnóstico médico, veículos autônomos e monitoramento inteligente. Os autores argumentam que, devido à sua estrutura adaptável e capacidade de melhorar continuamente com o treinamento, as CNNs têm um futuro promissor, sendo uma das tecnologias mais avançadas e versáteis no campo da inteligência artificial.

Os autores também exploram os desafios ainda enfrentados pelas CNNs, como a necessidade de grandes quantidades de dados para treinamento, a alta demanda computacional e as limitações em situações de baixa qualidade de imagem ou iluminação inadequada. Apesar disso, o estudo aponta que o progresso contínuo em hardware, algoritmos de otimização e técnicas de regularização estão expandindo o alcance e a aplicabilidade das CNNs.

Em conclusão, Li et al. (2021) reforçam o enorme potencial das redes neurais convolucionais em várias áreas, destacando especialmente sua aplicação na segurança no trabalho, onde a detecção automática de EPIs pode desempenhar um papel crucial na prevenção de acidentes. O estudo sugere que, com avanços tecnológicos contínuos, as CNNs terão um impacto ainda maior

em uma variedade de indústrias no futuro.

No artigo **YOLOv3: *An Incremental Improvement*** (REDMON; FARHADI, 2018), os autores apresentam uma versão aprimorada da arquitetura YOLO (*You Only Look Once*), amplamente reconhecida por sua eficiência em tarefas de detecção de objetos em tempo real. O YOLOv3 introduz uma série de melhorias em relação às versões anteriores, com o objetivo de otimizar a precisão e a velocidade da detecção, mantendo a característica central da família YOLO: a capacidade de detectar objetos em uma única passada pela imagem.

Os autores começam destacando que a abordagem YOLO é diferenciada por seu método de detecção em tempo real, o que a torna extremamente útil em aplicações que exigem uma análise rápida e precisa de imagens e vídeos. No YOLOv3, melhorias incrementais foram introduzidas na arquitetura da rede neural, incluindo o uso de "*residual blocks*", a adição de previsões em várias escalas e a substituição da função de ativação *leaky* ReLU pela função *sigmoid* para previsões de classes. Essas mudanças permitiram à rede detectar objetos de tamanhos variados com maior precisão, algo que as versões anteriores do YOLO enfrentavam dificuldades.

A principal contribuição do YOLOv3 foi sua capacidade de balancear eficiência e precisão. Enquanto redes mais complexas e profundas podem alcançar resultados mais precisos, elas geralmente exigem muito mais tempo de processamento, tornando-as inadequadas para aplicações em tempo real. O YOLOv3, por outro lado, mantém um desempenho competitivo em termos de precisão sem sacrificar a velocidade, o que o torna adequado para uma ampla gama de tarefas que exigem processamento ágil.

Um dos cenários destacados pelos autores é a aplicabilidade do YOLOv3 em ambientes industriais, onde a detecção rápida e precisa de Equipamentos de Proteção Individual (EPIs) é crucial para garantir a segurança dos trabalhadores. Nesses ambientes, a capacidade do YOLOv3 de identificar objetos como capacetes, coletes e outros EPIs em tempo real é especialmente valiosa, pois pode ser integrada a sistemas de monitoramento contínuo para garantir que os trabalhadores estejam utilizando o equipamento de segurança necessário durante toda a operação. A versatilidade do modelo permite que ele seja adaptado para outras tarefas de detecção, incluindo veículos autônomos, segurança em vídeo, e aplicações de vigilância.

Outra contribuição importante do artigo foi a análise comparativa do YOLOv3 com outros modelos de detecção de objetos, como o Faster R-CNN e o SSD (*Single Shot Multibox Detector*). Os testes mostraram que, embora o YOLOv3 não tenha sido o modelo mais preciso em todos os casos, ele ofereceu uma excelente relação entre velocidade e precisão, especialmente em cenários que exigem um compromisso entre os dois fatores. O YOLOv3 provou ser particularmente robusto em detecções de objetos pequenos e em diferentes escalas, graças à previsão em múltiplas escalas e à capacidade de combinar características em diferentes níveis da rede.

Os autores também discutem as limitações do YOLOv3. Embora o modelo tenha melhorado significativamente a detecção de objetos menores em comparação com suas versões anteriores, ainda há dificuldades em identificar objetos com grande sobreposição ou em cenas com iluminação precária. Além disso, a precisão do YOLOv3 diminui em tarefas que envolvem a detecção de um grande número de classes diferentes.

Por fim, o estudo conclui que o YOLOv3 representa uma evolução significativa dentro da família YOLO, mantendo a simplicidade e a eficiência do modelo original, ao mesmo tempo em que implementa melhorias que permitem uma detecção mais robusta e precisa. O modelo se destaca em aplicações que exigem alta velocidade, como a detecção de EPIs em tempo real em ambientes industriais, segurança de vídeo e monitoramento de tráfego. Com seu equilíbrio entre precisão e eficiência computacional, o YOLOv3 continua a ser uma ferramenta fundamental para tarefas de visão computacional em tempo real.

O artigo **YOLOv4:** ***Optimal Speed and Accuracy of Object Detection*** (BOCHKOVSKIY; WANG; LIAO, 2020) apresenta uma nova versão da família de redes YOLO, com foco em aprimorar tanto a velocidade quanto a precisão da detecção de objetos. O YOLOv4 foi projetado para superar as limitações das versões anteriores, mantendo a eficiência em tempo real e introduzindo diversas melhorias técnicas que o tornam uma escolha robusta para uma ampla gama de aplicações, especialmente em ambientes desafiadores.

Os autores começaram identificando a necessidade de um modelo que pudesse conciliar alta precisão e velocidade, sem sacrificar a simplicidade do treinamento em hardware disponível comercialmente, como GPUs convencionais. Com esse objetivo em mente, o YOLOv4 foi desenvolvido com um conjunto de aprimoramentos na arquitetura da rede neural, incluindo inovações como o uso do C*ross Stage Partial connections (CSPDarknet53)* como *backbone*, técnicas de *data augmentation* como *Mosaic* e *Self-Adversarial Training* (SAT), além de otimizações de processamento como o uso de *heads PANet* para a detecção de objetos em diferentes escalas.

Essas melhorias permitiram ao YOLOv4 alcançar uma performance superior em termos de precisão (mAP – *Mean Average Precision*) em comparação com outros modelos de detecção de objetos, como o *EfficientDet* e o *Faster* R-CNN, sem comprometer a velocidade de inferência. O YOLOv4 manteve a essência do YOLO de ser rápido e leve, o que é fundamental para aplicações de tempo real, como a vigilância e a análise de vídeos em larga escala.

O estudo enfatiza que o YOLOv4 foi particularmente bem-sucedido em cenários de detecção em tempo real em ambientes industriais e de construção. Um exemplo disso é o trabalho de Wang et al. (2020), que implementou o YOLOv4 para monitorar o uso de Equipamentos de Proteção Individual (EPIs), como capacetes e coletes, em canteiros de obras. O modelo foi capaz de detectar rapidamente a presença desses EPIs, mesmo sob condições adversas, como iluminação inadequada e oclusão parcial dos objetos. Isso demonstra a robustez do YOLOv4 para identificar objetos em situações em que outros modelos poderiam falhar ou ter uma queda

significativa de desempenho.

Além das melhorias na precisão e na capacidade de lidar com diferentes condições ambientais, o YOLOv4 também introduziu técnicas para otimizar o treinamento e a inferência, incluindo o uso de técnicas de regularização, como *DropBlock* e *CutMix*, para evitar o sobreajuste e melhorar a generalização do modelo. A integração dessas técnicas foi fundamental para que o YOLOv4 se destacasse em ambientes com alta variabilidade, como os canteiros de obras, onde os trabalhadores podem estar em diferentes posições e ângulos de visão.

Outro ponto discutido no artigo foi a flexibilidade do YOLOv4, que pode ser adaptado para uma ampla gama de aplicações além da detecção de EPIs. A arquitetura modular do modelo permite que ele seja ajustado para tarefas de detecção específicas, como a identificação de veículos em estradas, a vigilância de áreas urbanas e a análise de fluxos de pessoas em ambientes públicos. A capacidade de detectar objetos em múltiplas escalas, associada à sua alta taxa de frames por segundo (FPS), torna o YOLOv4 adequado para cenários que exigem decisões rápidas e precisas.

Os autores também realizaram comparações entre o YOLOv4 e outros modelos populares de detecção de objetos, como o RetinaNet, *EfficientDet* e SSD, evidenciando que o YOLOv4 apresentou o melhor equilíbrio entre velocidade e precisão. Em cenários onde o tempo de processamento é crítico, como na detecção de objetos em tempo real, o YOLOv4 foi superior, mantendo um alto nível de precisão sem comprometer a eficiência computacional.

Por fim, o artigo conclui que o YOLOv4 representa uma evolução significativa dentro da família YOLO, oferecendo uma solução que atende às exigências tanto de precisão quanto de velocidade para uma ampla gama de aplicações. Sua adoção em estudos como o de Wang et al. (2020), que focam na detecção de EPIs em ambientes de trabalho, mostra seu potencial para transformar práticas de segurança, automatizando processos de monitoramento e garantindo que os trabalhadores estejam adequadamente protegidos em tempo real. O YOLOv4, com suas diversas inovações, segue como uma escolha robusta para soluções de visão computacional em tempo real, oferecendo alto desempenho em cenários complexos e adversos.

O artigo **YOLOv5:** ***A Comparative Analysis with YOLOv4 and YOLOv3*** (JOCHER et al., 2021) oferece uma análise abrangente e comparativa entre diferentes versões da família YOLO, focando principalmente no YOLOv5 em comparação com o YOLOv4 e o YOLOv3. O estudo visa fornecer uma compreensão clara das vantagens e desvantagens de cada uma dessas variantes, ajudando pesquisadores e engenheiros a escolherem a versão mais adequada para suas aplicações específicas, incluindo a detecção de Equipamentos de Proteção Individual (EPIs).

Os autores começam o artigo introduzindo a evolução da família YOLO. O YOLOv3, lançado em 2018, foi amplamente utilizado devido à sua capacidade de detectar objetos em tempo real com alta eficiência. No entanto, com o lançamento do YOLOv4 em 2020, melhorias substanciais em termos de precisão e velocidade foram introduzidas. O YOLOv5, lançado

posteriormente por Glenn Jocher, foi projetado com foco na simplicidade de implementação e otimização para hardware moderno, como GPUs de alto desempenho, facilitando o treinamento e a inferência para uma ampla gama de usuários.

A principal contribuição deste trabalho foi a análise comparativa entre essas três versões do YOLO, levando em consideração fatores como a arquitetura, precisão, velocidade, capacidade de generalização, e o uso de recursos computacionais. A pesquisa destacou que, enquanto o YOLOv4 trouxe avanços notáveis em relação ao YOLOv3, especialmente com o uso de novas técnicas como o CSPDarknet53 (*backbone*) e o PANet para detecção em múltiplas escalas, o YOLOv5 se destacou por sua abordagem simplificada e altamente eficiente para o treinamento de modelos de detecção.

Em termos de precisão, o estudo mostrou que o YOLOv5 ofereceu um desempenho competitivo em relação ao YOLOv4 e ao YOLOv3, sendo capaz de detectar objetos com precisão em uma variedade de cenários. A arquitetura do YOLOv5 inclui aprimoramentos que permitem melhor detecção em situações de oclusão e em ambientes com baixa iluminação, características fundamentais para aplicações como a detecção de EPIs em canteiros de obras, onde os trabalhadores podem estar parcialmente ocultos por outros objetos ou em condições de luz variáveis.

Quanto à velocidade, o YOLOv5 se sobressaiu em relação aos seus predecessores. A pesquisa revelou que o YOLOv5 é significativamente mais rápido que o YOLOv4 e o YOLOv3, tanto no treinamento quanto na inferência. Isso se deve, em parte, à sua otimização para hardware moderno e ao uso eficiente de técnicas de processamento paralelo. Esse equilíbrio entre velocidade e precisão fez do YOLOv5 uma escolha popular para aplicações que exigem detecção em tempo real, como vigilância de segurança, monitoramento industrial e sistemas de transporte automatizados.

Os autores também enfatizaram a facilidade de uso do YOLOv5 em comparação com as versões anteriores. O YOLOv5 foi desenvolvido com uma integração mais direta com o *PyTorch*, uma popular biblioteca de aprendizado profundo, o que simplificou o processo de treinamento e modificação do modelo. Além disso, ele conta com *scripts* prontos para treinamento e inferência, o que facilita sua implementação por desenvolvedores com diferentes níveis de experiência. Essa acessibilidade contribuiu para a rápida adoção do YOLOv5 na comunidade de visão computacional.

Por outro lado, os autores também apontaram algumas limitações. Embora o YOLOv5 ofereça uma ótima combinação de velocidade e precisão, ele não necessariamente supera o YOLOv4 em todos os aspectos. Por exemplo, o YOLOv4 ainda é considerado mais robusto em termos de capacidade de detecção em cenários mais complexos, com muitas classes de objetos e ambientes desafiadores. Além disso, como o YOLOv5 foi desenvolvido como uma variante não oficial da família YOLO, ele não tem a mesma base teórica sólida ou as mesmas técnicas

inovadoras introduzidas por Bochkovskiy e seus colaboradores no YOLOv4.

A pesquisa também aborda a aplicabilidade do YOLOv5 na detecção de EPIs, uma área crescente de interesse em segurança no trabalho. Os autores destacam que o YOLOv5 foi amplamente utilizado em estudos que visam o monitoramento de trabalhadores em canteiros de obras, identificando corretamente o uso de capacetes, coletes e outros EPIs em tempo real. A capacidade de detectar EPIs com rapidez e precisão em ambientes dinâmicos, onde os trabalhadores podem estar em constante movimento, torna o YOLOv5 uma ferramenta valiosa para a automação de monitoramento de segurança.

O artigo conclui que, embora não exista uma versão "melhor"entre YOLOv3, YOLOv4 e YOLOv5, cada uma tem suas vantagens e desvantagens dependendo do contexto de aplicação. O YOLOv5, em particular, oferece um equilíbrio atraente entre precisão, velocidade e facilidade de uso, o que o torna uma escolha popular para muitos cenários de detecção de objetos, especialmente para a detecção de EPIs e outras aplicações industriais. No entanto, para tarefas mais complexas, o YOLOv4 ainda pode ser preferível devido à sua robustez em cenários desafiadores. Assim, a escolha do modelo ideal depende das necessidades específicas da aplicação, incluindo fatores como recursos computacionais disponíveis, número de classes de objetos e requisitos de tempo de processamento.

O artigo **YOLOv8:** ***Next-Generation Object Detection*** (JOCHER et al., 2023) apresenta a versão mais recente da família de modelos de detecção de objetos YOLO, destacando suas melhorias em relação às versões anteriores, tanto em termos de precisão quanto de velocidade. O YOLOv8 traz avanços arquitetônicos que otimizam a capacidade do modelo de identificar objetos de maneira eficiente em diversas aplicações. Embora ainda haja uma escassez de pesquisas específicas sobre o uso do YOLOv8 para a detecção de Equipamentos de Proteção Individual (EPIs), os resultados promissores obtidos com versões anteriores, como YOLOv5 e YOLOv7, indicam que o YOLOv8 terá um desempenho igualmente ou mais eficaz nesta tarefa.

A evolução do YOLO tem sido marcada por melhorias incrementais, com cada nova versão trazendo mais robustez na detecção em tempo real. No caso do YOLOv8, espera-se que sua arquitetura mais refinada contribua para um aumento significativo na precisão da detecção de EPIs, mesmo em ambientes complexos ou dinâmicos. A rapidez na identificação e classificação de objetos faz do YOLOv8 uma ferramenta poderosa, que poderá ser aplicada em áreas como segurança no trabalho, automatizando a fiscalização do uso correto de EPIs em ambientes industriais e de construção.

Portanto, o YOLOv8 representa uma continuação da tendência de desenvolvimento de modelos de visão computacional mais rápidos e precisos, consolidando-se como uma solução potencialmente eficaz para a detecção automatizada de EPIs, entre outras aplicações que demandam reconhecimento de objetos com alta performance.

Os trabalhos relacionados revisados evidenciam a evolução contínua dos modelos de

detecção de objetos, em particular a família YOLO, e seu impacto significativo na área de segurança industrial. Desde as primeiras implementações com o YOLOv3 até as melhorias introduzidas no YOLOv4 e YOLOv5, há uma clara tendência de aumento na precisão e na velocidade de detecção. A introdução do YOLOv8 representa o próximo passo nesta evolução, e este trabalho visa explorar e comparar as distribuições do YOLOv8 (*nano, small, medium e large*) para a detecção de EPIs.

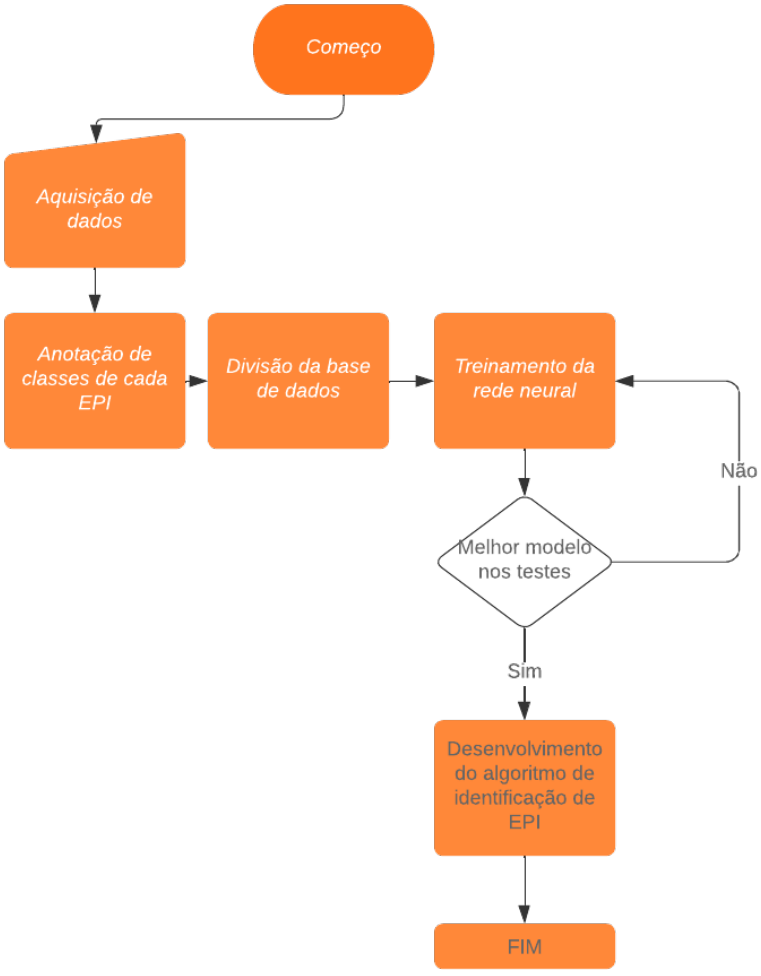
A revisão dos estudos anteriores não apenas contextualiza a importância da detecção automatizada de EPIs, mas também estabelece uma base comparativa sólida para a avaliação dos modelos YOLOv8. A análise comparativa das diferentes distribuições do YOLOv8 contribuirá para identificar a melhor abordagem para a detecção de EPIs, promovendo avanços na segurança do trabalho e na adoção de tecnologias de visão computacional em ambientes industriais.

4 METODOLOGIA

Este capítulo detalha a metodologia adotada na execução deste projeto, que envolve a identificação de Equipamentos de Proteção Individual (EPIs) específicos: capacete, óculos de proteção, luvas, colete, botas, máscara e abafador. A seleção desses EPIs foi baseada em uma revisão bibliográfica que indicou que eles são amplamente utilizados em diversos segmentos industriais. Além disso, esses equipamentos protegem membros e órgãos vitais do corpo humano, contribuindo significativamente para o bem-estar dos colaboradores tanto na vida cotidiana quanto no ambiente de trabalho.

Etapas do projeto: O projeto foi dividido em seis etapas principais, cada uma essencial para o desenvolvimento e implementação da metodologia proposta.

Figura 8 – Fluxograma das etapas do trabalho.



Fonte: Próprio Autor.

Aquisição de Imagens: Coleta de um conjunto abrangente de imagens que incluem pessoas utilizando ou não os EPIs definidos.

Anotação de Classes de cada EPI: Identificação e rotulagem dos EPIs nas imagens coletadas, criando um banco de dados anotado que servirá como base para o treinamento do modelo.

Divisão da Base de Dados: Processo em que as imagens da base de dados serão divididas para fazer o treinamento da rede neural.

Ambiente e Modelagem do algoritmo para treinamento da rede neural e ajustes de parâmetros: Explicação do ambiente de programação utilizado e desenvolvimento de um algoritmo de treinamento de rede neural capaz de classificar e delimitar a presença dos EPIs nas imagens. Esta etapa envolve a escolha da arquitetura da rede e a configuração inicial dos parâmetros.

Desenvolvimento do Algoritmo de Identificação de EPI: Implementação de um algoritmo que utiliza a rede neural treinada para identificar EPIs.

Ao seguir essas etapas, buscamos desenvolver um sistema robusto e eficiente para a identificação automática de EPIs, contribuindo para a segurança no ambiente de trabalho e minimizando riscos de acidentes.

4.0.1 Aquisição de Imagens

Para a elaboração do projeto foi utilizada uma base de dados com um total de 2295 imagens de pessoas em diversos ambientes de trabalho fazendo uso ou não dos Equipamentos de Proteção Individual (EPI's) das mais variadas formas, modelos e cores.

Dessas imagens, 2153 estão em domínio público no site "https://public.roboflow.com/" (Roboflow, 2024).

4.0.2 Anotação das Classes de Cada EPI

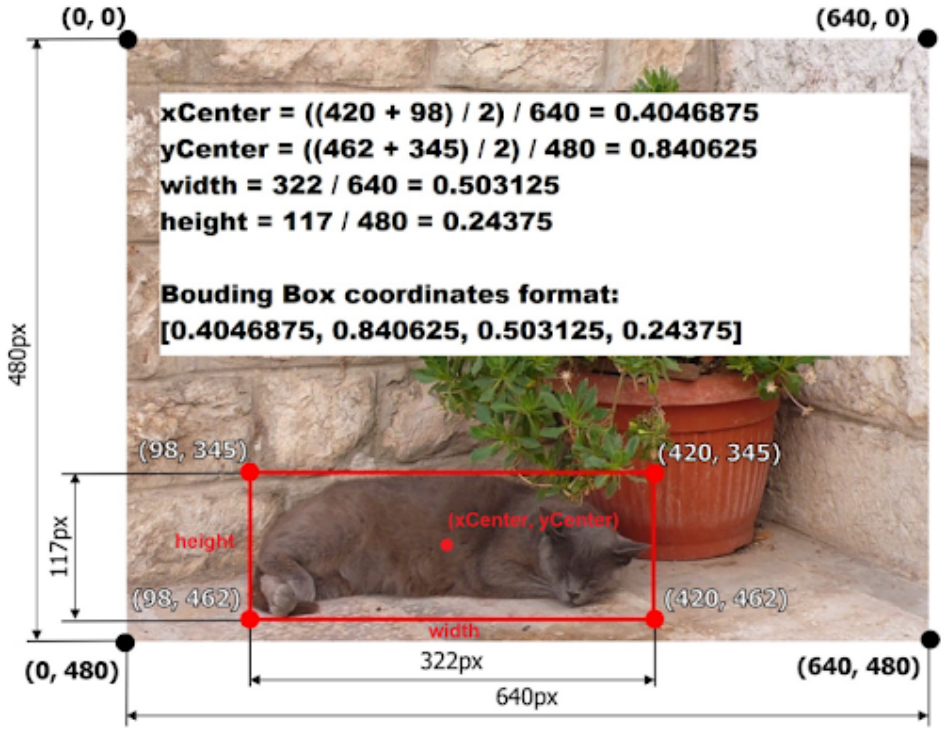
Assim como qualquer outra rede neural convolucional, o YOLO precisa ser carregado com dados pré-classificados. Por padrão, essa rede utiliza um arquivo ".txt"para cada imagem no banco de dados, com o mesmo nome da imagem correspondente. Este arquivo de texto informa à rede as classes dos objetos e suas localizações na imagem.

Para localizar os objetos, o YOLO utiliza caixas delimitadoras, representadas por coordenadas específicas. A caixa delimitadora é definida por quatro valores: [xCentro, yCentro, largura, altura], todos normalizados pela largura e altura máximas da imagem. A Figura 9 ilustra um exemplo de cálculo do padrão de anotação de um objeto no modelo YOLO para garantir seu funcionamento correto.

As classes de objetos no arquivo de texto são anotadas com números inteiros. Neste projeto, foram definidas sete classes diferentes, variando de 0 a 6, onde cada classe corresponde

a: "Botas", "Óculos", "Luvas", "Capacete", "Colete", "Abafador"e “Máscara”.

Figura 9 – Demonstração do cálculo das coordenadas da caixa delimitadora para modelo.

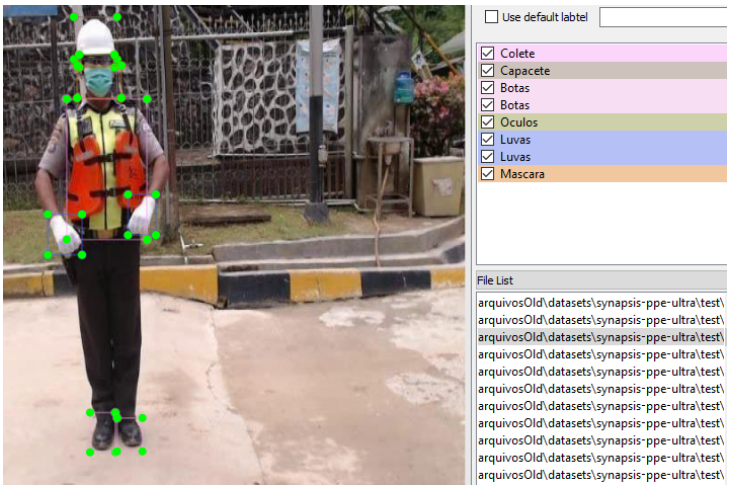


Fonte: Albumentations (2019).

Para anotar cada objeto encontrado nas 2295 imagens no formato de coordenadas padrão do YOLO, utilizamos o programa LabelIMG, um software de código aberto para anotação de imagens escrito em *Python* e que utiliza Qt, um *framework* C++ comum no desenvolvimento de interfaces gráficas. A ferramenta foi criada por Tzutalin (2017) e está disponível no GitHub.

No software, as imagens são abertas individualmente. O programa permite selecionar manualmente o objeto a ser identificado, utilizando a opção de demarcar o objeto com uma *bounding box*, que seleciona o objeto em sua totalidade, respeitando seus limites para especificar sua localização exata na imagem. O rótulo é atribuído conforme mencionado anteriormente, como mostrado na Figura 10.

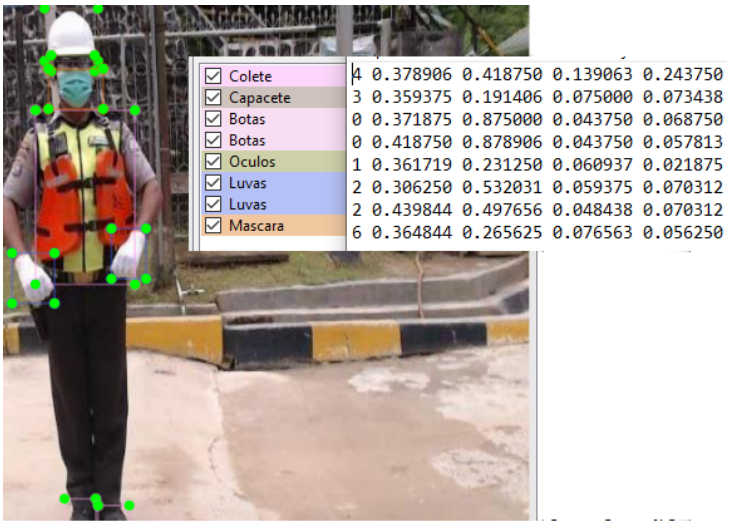
Figura 10 – Anotação de trabalhador fazendo uso de EPI's no programa LabelImg.



Fonte: Adaptado de (Roboflow Universe, 2024).

O programa então salva as coordenadas e o tipo dos objetos anotados em um arquivo ".txt"com o mesmo nome do arquivo de imagem. Note que o primeiro caractere de cada linha do arquivo .txt (número inteiro) corresponde ao número para identificação da classe. Esse número inteiro é seguido por outros quatro números decimais, responsáveis por indicar as coordenadas da caixa delimitadora do objeto detectado, Veja a seguir a Figura 11.

Figura 11 – Documento .txt produzido pelo programa, associando imagem com a coordenada da localização do objeto.



Fonte: Adaptado de (Roboflow Universe, 2024).

Foi necessário também, fazer uma alteração no arquivo "data.yaml", que vem junto ao fazer o download do dataset. É preciso adicionar o caminho das pastas com as imagens para o diretório do kaggle, veja a imagem 12. Nessa imagem, é possivel notar que as classes e, ao final, o caminho com o diretório das pastas. Há também outras informações como, o nome da base de dados e o link onde está disponibilizado.

Figura 12 – Conteúdo do arquivo data.yaml mostrando as classes existentes no dataset e o caminho para as pastas com as imagens.

```
names:
- Botas
- Óculos
- Luvas
- Capacete
- Colete
- Abafador
- Mascara
nc: 7
roboflow:
  license: CC BY 4.0
  project: synapsis-ppe
  url: https://universe.roboflow.com/computer-vision-8kpih/synapsis-ppe/dataset/513
  version: 513
  workspace: computer-vision-8kpih
test: /kaggle/input/dataset/dataset_ultra/test/images
train: /kaggle/input/dataset/dataset_ultra/train/images
val: /kaggle/input/dataset/dataset_ultra/valid/images
```

Fonte: Próprio Autor.

4.0.3 Divisão da Base de Dados

Após finalizar a anotação das imagens no banco de dados, foi necessário particionar a base de dados em duas partes, treinamento e validação. Os próprios desenvolvedores do Yolo recomendam que o dataset seja dividido em 70% para treinamento e 30% para validação, podendo variar. (ULTRALYTICS, 2024) A base de dados foi dividida da seguinte maneira: 83% treinamento, 12% validação e 5% para teste. Essa proporção foi aplicada ao número total de imagens.

4.0.4 Ambiente, modelagem do algoritmo para treinamento da rede neural e ajustes de parâmetros

Foi utilizado para o desenvolvimento do projeto a plataforma Kaggle (KAGGLE, 2024), uma plataforma em nuvem que possibilita ao usuário executar código Python e diversas linguagens de programação em um ambiente Jupyter Notebook. Optou-se por utilizar uma plataforma em nuvem pela facilidade de configuração e instalação de frameworks responsáveis por construir a rede neural, no caso desse projeto que utiliza YoloV8 esse framework é o PyTorch que nos dispõe um amplo arsenal de ferramentas e bibliotecas. A decisão pela escolha de utilizar um ambiente em nuvem também se deve ao uso de recursos computacionais, o kaggle oferece 30h semanais de uso de gpu (Nvidia Tesla P100) gratuitamente. Vale ressaltar que, para a otimização do tempo de treinamento, é necessário o uso de GPU como acelerador, porém nem todos os processadores gráficos são compatíveis com a engine CUDA utilizada pelo Yolo, somente os chips da Nvidia tem esse suporte. O ambiente do kaggle nos oferece: processador Intel(R) Xeon(R) CPU @ 2.20GHz, 32GB de memória RAM e GPU NVIDIA Tesla P100 GPU 16GB

4.0.4.1 Configurações da rede Yolo

A princípio, foi necessário fazer o upload da base de dados na plataforma em formato .zip. Após isso, é preciso clonar o repositório da Ultralytics no GitHub, nele irá conter todas as ferramentas e arquivos necessários para treinar a nossa rede utilizando YoloV8 e, em seguida, é necessário importar esses arquivos para o ambiente de programação utilizando os seguintes códigos:

Figura 13 – Instalando e importanto o ultralytics.

```
In [1]: !pip install ultralytics -q

In [2]: from ultralytics import YOLO
```

Fonte: Próprio autor.

4.0.4.2 Treinamento do Modelo

Para o treinamento das redes usadas no projeto, foi aplicada uma técnica de transfer learning (aprendizado por transferência), pois sempre um modelo pré treinado do yolo será utilizado para o treinamento das redes do projeto. Para fazer o download do modelo pré treinado basta utilizar o código abaixo:

Figura 14 – Selecionando a distribuição do YOLO a ser treinada.

```
In [3]: modelo = YOLO ('yolov8n.pt')
```

Fonte: Próprio autor.

Nesse caso, estamos usando o modelo yolov8n, mas o intuito deste projeto é utilizar também as outras distribuições (yolov8s, yolov8l e yolov8m) a fim de fazer um comparativo no final. Então, esse processo todo será repetido para os outros modelos pré treinados.

Antes de começar a parametrizar as métricas para o treinamento, foi necessário importar uma ferramenta chamada wandb, que se mostrou muito útil na hora do treinamento. Essa ferramenta é responsável por mostrar, em tempo real, todas as métricas e dados do treinamento, plotando gráficos e gerando códigos em formato .json que poderão ser usados para gerar os mais diversos gráficos, personalizáveis do jeito que o usuário preferir. Sua utilização é simples, basta criar uma conta na plataforma “https://wandb.ai/” (BIASES, 2024), acessar as configurações da

conta e gerar uma chave de acesso. Após isso, é preciso importar a biblioteca no nosso ambiente e colocar os seguintes comandos:

Figura 15 – Importando a ferramente WanDB no ambiente de programação

```
In [4]:
import wandb

wandb.login(key="b02e383d4d0a6682237fdffc5d023d40a9262206")
wandb.init(
    project='testtrainmycustomdataset',
    name="project_yolov8n",
)
```

Fonte: Próprio autor.

Na linha 2, em 'key=', é onde se deve colocar a chave de acesso do usuário. Feito isso, poderemos acompanhar na plataforma wandb as métricas de desempenho, junto com os gráficos em tempo real do treinamento.

A execução do treinamento será feita a partir do seguinte código:

Figura 16 – iniciando o treinamento

```
In [5]:
custom_model_results = modelo.train(data='/kaggle/input/dataset/dataset_ultra/data.yaml', epochs=300, wor
kers=2, batch=80)
```

Fonte: Próprio autor.

No comando acima, responsável pelo início do treinamento usamos a função modelo.train, onde modelo é o nosso modelo pré treinado (yolov8n, nesse caso) e passamos dentro dessa função o caminho do arquivo data contido dentro da base de dados. É também possível notar que possui as configurações de alguns parâmetros. Os argumentos mostrados no código desempenham uma função específica, e essas configurações podem alterar o desempenho, velocidade e a precisão do modelo. Nesse caso, será necessário apenas alterar os parâmetros de epochs, workers e batch.

Epochs=300: Define o número de épocas de treinamento, que nada mais é que a quantidade de vezes que o conjunto de dados é passado pela rede neural;

Workers=2: Número de threads de trabalho para carregamento de dados;

Batch=80: Número de imagens por lote, quanto maior o batch mais rápido o treinamento é concluído. Um batch maior significa também maior quantidade de uso de VRAM.

O yolo permite também a alteração de outros parâmetros, que nesse projeto ficarão no padrão, mas daremos alguns exemplos abaixo:

Degrees=30.0: Realiza rotações aleatórias na imagem com ângulos entre +30 e -30 graus;

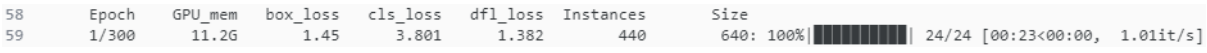
Scale=0.4: Aplica zoom de 40% na imagem aleatoriamente; probabilidade de 20%;

Mixup=0.3: Realiza mistura de imagens, inserindo uma imagem dentro de outra, com uma probabilidade de 30%;

Flipr=0.5: Realiza espelhamento horizontal na imagem com uma probabilidade de 50%.

Os valores utilizados são apenas exemplos. Existem muitos outros parâmetros que o yolo nos permite alterar a fim de "dificultar" o treinamento e, assim, aumentar a capacidade de detecção de objetos nos mais variados ambientes. Ao iniciar o treinamento, o próprio yolo já nos mostra alguns dados em tempo real como mostra a Figura 17:

Figura 17 – Visualização em tempo real do treinamento da rede Yolo.



Fonte: Próprio Autor.

Nessa imagem é possível ver que o treinamento nos fornece alguns dados em tempo real, como época, o uso de memória de vídeo, box loss (média que mostra quão bem o modelo previu a localização e tamanho da caixa delimitadora em comparação com a original) e cls loss (mede o quão bem o modelo previu a classe do objeto detectado em comparação com a classe real). Ao final do treinamento, vai ser gerado no kaggle uma pasta com alguns gráficos de desempenho e duas redes neurais, denominadas last.pt (que é o pior modelo treinado) e best.pt (melhor modelo treinado). Esse processo de treinamento foi repetido para os outros modelos pré treinados do yolov8. Foi exatamente o mesmo processo, mudando apenas o modelo e foi alterado também o número de *batch* para cada modelo em específico, uma vez que, conforme a complexidade da rede neural aumenta (*nano, small, medim e large*), o uso de memória gráfica para o treinamento também aumenta. Então se tornou necessário ajustar o *batch* para os modelos. Confira a tabela 1 no capítulo de resultados.

4.1 Desenvolvimento do Algoritmo de Identificação de EPI

Com os treinamentos dos modelos já finalizados, a próxima etapa foi implementar um código para fazer a detecção de EPI's a partir de um vídeo ou uma filmagem em tempo real através de webcam, foi utilizado o código a seguir:

Figura 18 – Algoritmo para identificação de EPI's

```
from ultralytics import YOLO
model = YOLO("yolo/yolov8n/runs/detect/train/weights/best.pt")

video_path = "Videos/epi-4.mp4"

results = model(
    source=video_path,
    stream=False,
    save=True,
    show=True
)
```

Fonte: Próprio autor.

Esse código irá carregar o modelo treinado, nesse caso o modelo treinado a partir do yolov8n, e especifica o caminho do vídeo. Caso a detecção seja a partir de uma webcam, basta excluir o comando "video_path = "Videos/epi-4.mp4"" e alterar o valor de source para "0". Ao finalizar a execução o vídeo resultante é salvo na pasta raiz do projeto.

4.2 Métricas de Desempenho

Para avaliar o desempenho dos modelos treinados serão usados alguns gráficos para ajudar na visualização e entendimento dos resultados. Esses gráficos irão conter métricas de precisão (*precision*), sensibilidade (*recall*) e *F1-score* (mede o equilíbrio entre precisão e sensibilidade).

4.2.1 Precição (*Precision*)

A precisão é uma métrica que indica a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões positivas. Em outras palavras, ela mede a exatidão das detecções feitas pelo modelo. Uma precisão alta significa que a maioria das detecções do modelo são corretas, minimizando os falsos positivos.

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos (TP)}}{\text{Verdadeiros Positivos (TP)} + \text{Falsos Positivos (FP)}} \tag{4.1}$$

Se a precisão é 0.85, significa que 85% das vezes que o modelo previu um EPI, ele estava correto.

4.2.2 Sensibilidade (*Recall*)

Mede a porcentagem de instâncias positivas corretamente identificadas pelo modelo. Ou seja, entre todas as instâncias que eram realmente positivas, quantas foram corretamente previstas como positivas

$$\text{Sensibilidade} = \frac{\text{Verdadeiros Positivos (TP)}}{\text{Verdadeiros Positivos (TP)} + \text{Falsos Negativos (FN)}} \tag{4.2}$$

Se o *recall* é 0.80, significa que o modelo conseguiu identificar 80% de todos os EPIs presentes nas imagens.

4.2.3 F1-Score

É a média harmônica da precisão e do *recall*, fornecendo um único valor que balanceia ambos.

$$\text{F1-score} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{4.3}$$

Se o F1-score é 0.825, indica que o modelo tem um bom equilíbrio entre precisão e recall, sendo um indicador geral de desempenho.

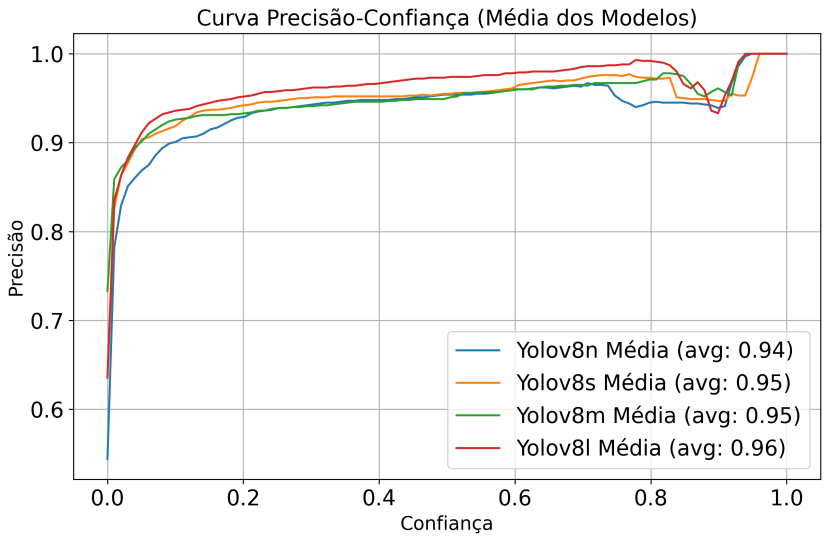
5 RESULTADOS

Neste capítulo, apresentamos e discutimos os resultados obtidos durante o treinamento e avaliação dos modelos de detecção de Equipamentos de Proteção Individual (EPI) utilizando a ferramenta YOLOv8. Os modelos foram avaliados em termos de precisão, sensibilidade, F1-score e matriz de confusão. Além disso, foram gerados gráficos para visualização das curvas de *precision-confidence, recall-confidence, F1-confidence e precision-recall*, comparando a média de desempenho de cada modelo treinado.

5.1 Precisão-Confiança (*Precisio-Confidence*)

O gráfico de precisão-confiança (Figura 19) é uma representação visual importante para entender o desempenho do modelo YOLOv8 na detecção de Equipamentos de Proteção Individual (EPI). Ele exibe a relação entre a precisão das detecções e a confiança do modelo ao fazer essas detecções.

Figura 19 – Gráfico Precisão x Confiança: curva média dos modelos YoloV8.



Fonte: Próprio Autor.

Análise gráfica: O gráfico mostra quatro curvas, cada uma representando a média dos resultados de um modelo YOLOv8 treinado com diferentes tamanhos de rede:

- **YOLOv8n (Nano)**
- **YOLOv8s (Small)**
- **YOLOv8m (Medium)**

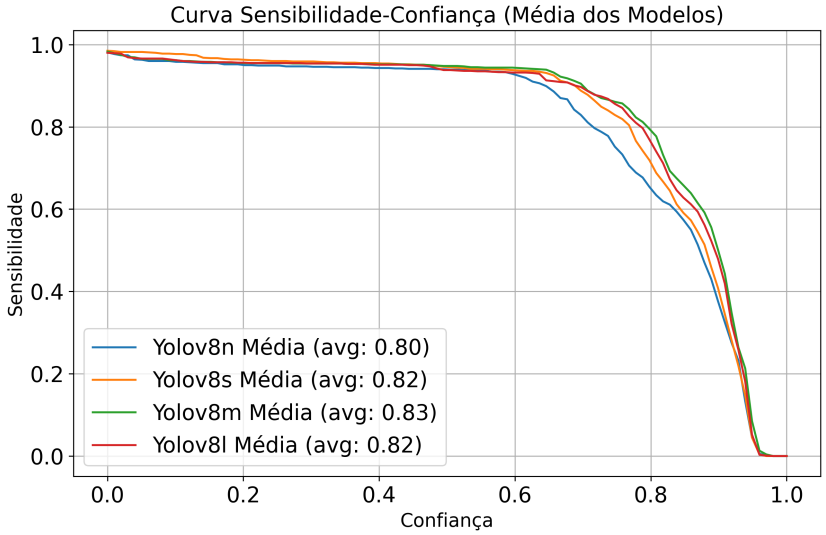
- **YOlov8l (Large)**

No início do gráfico (confiança próxima de 0), a precisão é relativamente baixa, pois o modelo faz muitas detecções, incluindo falsos positivos. À medida que a confiança aumenta, a precisão também aumenta. Isso ocorre porque o modelo se torna mais seletivo, reduzindo o número de falsos positivos. A partir de uma certa confiança (em torno de 0.3 a 0.4), a precisão das curvas começa a se estabilizar. Isso indica que o modelo está fazendo detecções mais precisas. YOLOv8l (linha vermelha) mostra a melhor performance em termos de precisão média, seguida por YOLOv8s e YOLOv8m (linhas laranja e verde), e por último, YOLOv8n (linha azul). Isso sugere que redes maiores e mais complexas tendem a ter uma precisão melhor. Em confianças mais altas (próximas de 1.0), todos os modelos atingem uma precisão de 1.0, indicando que as poucas detecções feitas nesses níveis de confiança são quase sempre corretas.

5.2 Sensibilidade-Confiança (*Recall-Confidence*)

O gráfico de Recall-Confidence (Figura 20) é uma representação visual importante para entender o desempenho do modelo YOLOv8 na detecção de Equipamentos de Proteção Individual (EPI). Ele exibe a relação entre a recall das detecções e a confiança do modelo ao fazer essas detecções. Esse tipo de gráfico é essencial para avaliar como o modelo equilibra a detecção correta de EPIs (*recall*) com a certeza de que essas detecções são precisas (confiança).

Figura 20 – Gráfico Sensibilidade X Confiança: curva média dos modelos YoloV8.



Fonte: Próprio Autor.

Todas as curvas começam com um sensibilidade alta (próximo de 1.0) em níveis baixos de confiança, indicando que os modelos detectam quase todos os EPIs quando não são exigentes. À medida que a confiança aumenta, o *recall* diminui. Isso ocorre porque o modelo começa a

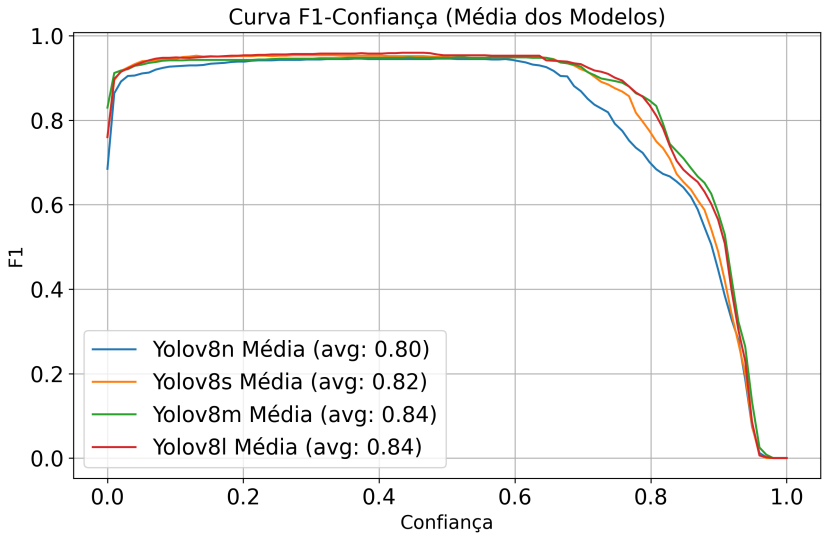
descartar detecções consideradas menos confiáveis, reduzindo o número de verdadeiros positivos. A taxa de queda do *recall* varia entre os modelos. Por exemplo, o YOLOv8m (verde) mantém uma sensibilidade mais alta em níveis de confiança intermediários em comparação com os outros modelos. O YOLOv8n (azul) tem uma queda mais acentuada na sensibilidade em níveis de confiança mais altos, indicando que ele é mais suscetível a perder verdadeiros positivos quando a confiança é aumentada. Os termos indicados por "médias" nas legendas, representam a média geral do *recall* ao longo de todos os níveis de confiança para o modelo em questão. Isso fornece uma visão geral do desempenho relativo de cada modelo.

Este gráfico de *Recall-Confidence* é essencial para entender como diferentes modelos YOLOv8 equilibram a detecção de EPIs em relação ao nível de confiança. A análise dessas curvas permite identificar qual modelo oferece o melhor desempenho para a aplicação específica de detecção de EPIs, considerando a necessidade de maximizar o *recall* ou a precisão em diferentes cenários operacionais.

5.3 F1-Confiança (*F1-Confidence*)

O gráfico de *F1-Confidence* (Figura 21) é uma representação visual importante para entender o desempenho do modelo YOLOv8 na detecção de Equipamentos de Proteção Individual (EPI). Ele exibe a relação entre a pontuação F1 das detecções e a confiança do modelo ao fazer essas detecções. Esse tipo de gráfico é essencial para avaliar como o modelo equilibra a precisão e o *recall* em diferentes níveis de confiança.

Figura 21 – Gráfico F1-score x Confiança: Curva média dos modelos YoloV8.



Fonte: Próprio Autor.

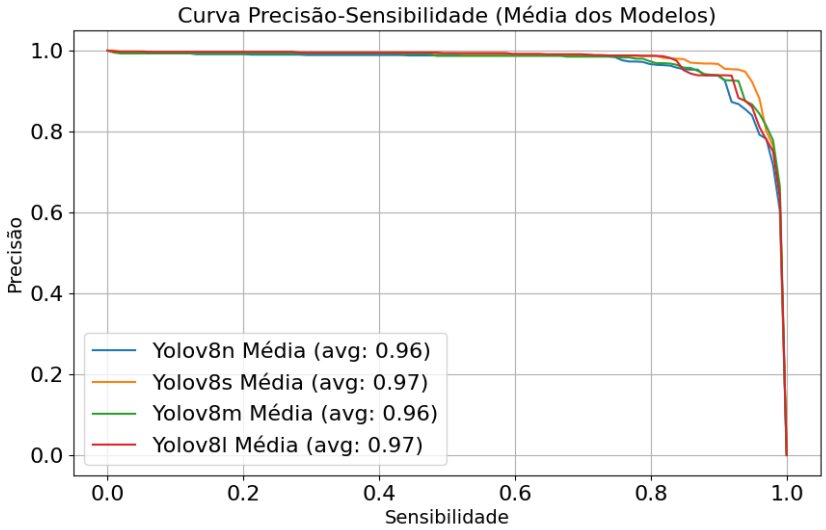
Todas as curvas começam com um *F1 score* relativamente baixo em níveis muito baixos de confiança, indicando que a precisão pode ser baixa inicialmente. À medida que a confiança

aumenta, o *F1 score* aumenta rapidamente e atinge um platô, onde o modelo equilibra bem precisão e *recall*. Em níveis de confiança mais altos, o *F1 score* começa a diminuir, pois o modelo se torna mais exigente e começa a descartar verdadeiros positivos, reduzindo o *recall*. A taxa de queda do *F1 score* varia entre os modelos. Por exemplo, o YOLOv8m (verde) e YOLOv8l (vermelho) mantêm um *F1 score* mais alto em níveis de confiança intermediários em comparação com os outros modelos. O YOLOv8n (azul) tem uma queda mais acentuada no *F1 score* em níveis de confiança mais altos, indicando que ele é mais suscetível a perder verdadeiros positivos quando a confiança é aumentada. Os termos indicados por "médias" nas legendas, representam a média geral do *F1-score* ao longo de todos os níveis de confiança para o modelo em questão. Isso fornece uma visão geral do desempenho relativo de cada modelo. Este gráfico de *F1-Confidence* é essencial para entender como diferentes modelos YOLOv8 equilibram a precisão e o *recall* em relação ao nível de confiança. A análise dessas curvas permite identificar qual modelo oferece o melhor desempenho para a aplicação específica de detecção de EPIs, considerando a necessidade de maximizar tanto a precisão quanto o *recall* em diferentes cenários operacionais

5.4 Análise da Curva Precision-Recall dos Modelos YOLOv8

O gráfico apresentado na Figura 22 é uma representação visual importante para entender o desempenho do modelo YOLOv8 na detecção de Equipamentos de Proteção Individual (EPI). Ele exibe a relação entre a precisão das detecções e o *recall* do modelo ao fazer essas detecções. Esse tipo de gráfico é essencial para avaliar como o modelo equilibra a precisão e o *recall* em diferentes situações

Figura 22 – Gráfico Precisão x Sensibilidade: Curva média dos modelos YoloV8.



Fonte: Próprio Autor.

Todas as curvas começam com uma alta precisão (próximo de 1.0) em níveis baixos de *recall*, indicando que as primeiras detecções feitas pelo modelo são quase todas corretas. À medida que o *recall* aumenta, a precisão se mantém alta por um longo intervalo, mostrando que o modelo continua fazendo detecções corretas mesmo ao detectar mais EPIs. Em níveis altos de *recall*, a precisão começa a cair rapidamente. Isso ocorre porque o modelo começa a detectar mais falsos positivos ao tentar maximizar o recall. As curvas de YOLOv8s (laranja) e YOLOv8l (vermelho) mantêm a precisão um pouco mais alta em níveis intermediários e altos de *recall* em comparação com os outros modelos. As médias indicadas nas legendas, como "YOLOv8n Média (avg: 0.96)", representam a média geral da precisão ao longo de todos os níveis de *recall*. Isso fornece uma visão geral do desempenho relativo de cada modelo. Este gráfico de *Precision-Recall* é essencial para entender como diferentes modelos YOLOv8 equilibram a precisão e o *recall* na detecção de EPIs. A análise dessas curvas permite identificar qual modelo oferece o melhor desempenho para a aplicação específica de detecção de EPIs, considerando a necessidade de maximizar tanto a precisão quanto o *recall* em diferentes cenários operacionais.

5.5 Avaliação do Desempenho do Modelo por Meio da Matriz de Confusão Normalizada

A matriz de confusão mostrada na figura 23 é uma ferramenta fundamental para a avaliação de modelos de classificação, fornecendo uma visão detalhada do desempenho do modelo em cada classe. Neste caso, usaremos como exemplo a matriz de confusão que foi gerada após o treinamento do modelo YOLOv8l (versão *large*) para a detecção de Equipamentos de Proteção Individual (EPIs). A matriz de confusão normalizada permite uma melhor interpretação dos resultados, independentemente do número de amostras em cada classe.

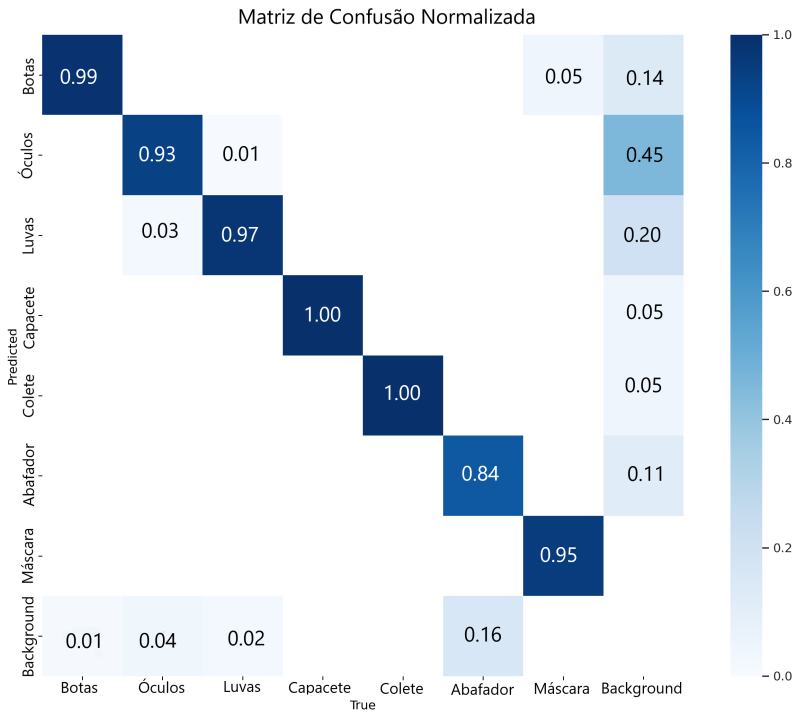
Tabela 1 – Resultados da matriz de confusão normalizada

Classe	Precisão	Confusão com Outras Classes
Botas	99%	1% Classificado como background
Óculos	93%	1% Classificado como luva, 5% classificado como background
Luvas	97%	3% Classificado como óculos
Capacete	100%	Nenhuma confusão (todas as detecções foram corretas)
Colete	100%	Nenhuma confusão (todas as detecções foram corretas)
Abafador	84%	16% classificado como background
Máscara	95%	5% classificado como background

Fonte: Próprio autor.

A matriz de confusão normalizada para o modelo YOLOv8l indica um desempenho robusto na detecção de EPIs, com altas precisões para a maioria das classes. No entanto, há algumas confusões notáveis, especialmente entre Óculos e Background, e entre Abafador e *Background*. Essas observações são cruciais para direcionar futuras melhorias no modelo,

Figura 23 – Matriz de confusão normalizada para o modelo YOLOV8l.



Fonte: Próprio Autor.

como aumentar a quantidade de dados de treinamento para classes específicas ou ajustar os hiperparâmetros para melhorar a distinção entre classes com maior confusão.

5.6 Comparação dos resultados das redes Yolov8

Nesta seção, apresentamos uma análise comparativa detalhada do desempenho dos diferentes modelos YOLOv8 (*nano, small, medium, large*) na tarefa de detecção de Equipamentos de Proteção Individual (EPIs). Os modelos YOLOv8 são avaliados em termos de precisão (*precision*) para várias classes de EPIs e também em relação ao tempo de processamento. A tabela a seguir resume os resultados obtidos para cada modelo treinado.

A precisão para a detecção de botas é consistentemente alta em todos os modelos, variando de 99.0% no YOLOv8n a 99.6% no YOLOv8l. Isso indica que a tarefa de detecção de botas é bem executada por todos os modelos, com pequenas melhorias à medida que a complexidade do modelo aumenta. A detecção de óculos apresenta uma precisão crescente conforme a complexidade do modelo aumenta, começando em 93.0% no YOLOv8n e atingindo 95.5% no YOLOv8l. Esse aumento sugere que modelos mais complexos conseguem lidar melhor com a variabilidade visual presente na classe de óculos.

Similar aos óculos, a precisão na detecção de luvas melhora com a complexidade do modelo, variando de 97.0% no YOLOv8n a 98.0% no YOLOv8l. Este resultado reforça a capacidade dos modelos mais avançados em distinguir detalhes finos que podem diferenciar

Tabela 2 – Resultados de treinamento para diferentes modelos Yolov8

Anotação	Yolov8n	Yolov8s	Yolov8m	Yolov8l
Batch	80	60	40	20
Tempo de treinamento	2h05m	2h54m	6h41m	10h27m
Botas	99.0%	99.5%	99.5%	99.6%
Óculos	93.0%	94.0%	95.0%	95.5%
Luvas	97.0%	97.5%	97.6%	98.0%
Capacete	100%	100%	100%	100%
Colete	100%	100%	100%	100%
Abafador	84.0%	85.0%	85.0%	86.0%
Máscara	95.0%	95.5%	96.0%	96.5%
mAP	94.0%	95.0%	95.0%	96.0%

Fonte: Próprio autor.

luvas de outras classes. Todos os modelos YOLOv8 apresentam uma precisão perfeita de 100.0% para as classes de capacete e colete. Isso indica que essas classes são facilmente identificáveis pelos modelos, possivelmente devido a características visuais distintivas ou uma maior representatividade nos dados de treinamento.

A precisão na detecção de abafadores varia de 84.0% a 86.0%, com um desempenho ligeiramente melhor nos modelos mais complexos. No entanto, essa classe apresenta a menor precisão entre todas, sugerindo que pode haver desafios específicos na identificação de abafadores que requerem atenção adicional.

A precisão na detecção de máscaras melhora gradualmente de 95.0% no YOLOv8n para 96.5% no YOLOv8l. Este padrão de melhoria é consistente com outras classes, indicando que os modelos mais complexos conseguem capturar melhor os detalhes necessários para uma detecção precisa.

O mAP, que é a média da precisão para todas as classes, varia de 94.0% no YOLOv8n a 96.0% no YOLOv8l. Este aumento confirma que modelos mais complexos oferecem uma performance geral melhorada, capturando mais corretamente a presença dos EPIs em diferentes cenários.

O tempo de treinamento para cada modelo é um fator crítico a considerar. Modelos mais complexos, como o YOLOv8l, requerem significativamente mais tempo para treinamento, 10h 27m comparado a 2h 05m para o YOLOv8n. A escolha do modelo ideal deve balancear a precisão e o tempo de processamento conforme as necessidades específicas da aplicação.

A análise comparativa demonstra que, embora todos os modelos YOLOv8 ofereçam um bom desempenho na detecção de EPIs, os modelos mais complexos (YOLOv8m e YOLOv8l) apresentam precisão superior, especialmente para classes mais desafiadoras como óculos e abafador. A precisão perfeita para capacete e colete em todos os modelos é um ponto positivo,

indicando uma robustez particular para essas classes.

Além disso, o tempo de treinamento deve ser considerado ao escolher o modelo a ser implementado em cenários práticos. A seleção do modelo deve equilibrar a precisão desejada com as limitações de tempo de processamento, garantindo uma solução eficaz e eficiente para a detecção de EPIs. Esta tabela e análise detalhada proporcionam uma visão clara do desempenho dos diferentes modelos YOLOv8, auxiliando na tomada de decisões informadas para a implementação prática de sistemas de detecção de EPIs.

5.7 Resultados do melhor modelo aplicado em imagens

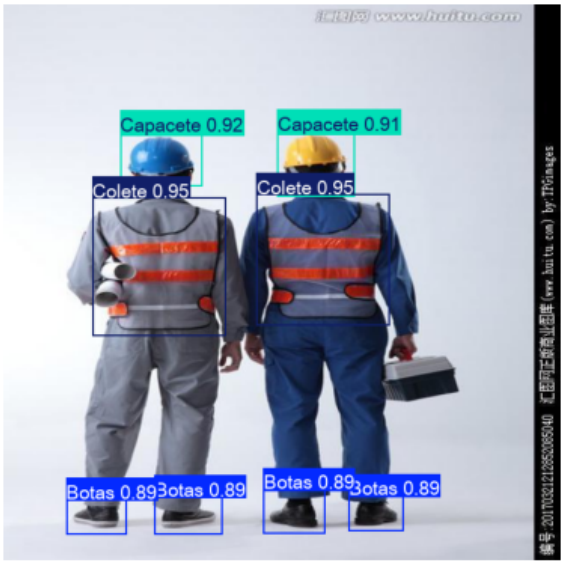
O melhor modelo foi escolhido com base no custo/benefício, onde o custo seria tempo de treinamento e recursos computacionais utilizados e o benefício seria a precisão de detecção do modelo. A partir dessa análise foi escolhido o modelo yolov8s, que exigiu um pouco mais de tempo de treinamento comparado com a distribuição mais leve do modelo, porém teve um aumento entre 0,5 a 1% de precisão na detecção das classes. As figuras a seguir vão nos mostrar exemplos da melhor rede neural em prática, como ela conseguiu classificar e segmentar corretamente os EPI's.

Figura 24 – Resultado do melhor modelo em imagens avulsas.

Fonte: Próprio Autor.

Na imagem 24 podemos notar que o nosso modelo conseguiu, com sucesso, identificar e segmentar corretamente os EPI's utilizados pelos trabalhadores, mesmo estando com um pedaço de madeira cobrindo parte do colete e por baixo das luvas.

Figura 25 – Resultado do melhor modelo em imagens avulsas.



Fonte: Próprio Autor.

Já na imagem 25, o modelo consegue identificar e segmentar corretamente os EPIs mesmo com os trabalhadores estando de costa.

Figura 26 – Resultado do melhor modelo em imagens avulsas.

Fonte: Próprio Autor.

Nesta imagem 26, o modelo consegue identificar corretamente os coletes (um em uso e outro não) mesmo estando ao fundo da imagem.

6 CONCLUSÕES E TRABALHOS FUTUROS

Pode-se concluir que todas as distribuições do YOLOv8 proporcionaram resultados consistentes e dentro das expectativas, mostrando-se altamente eficazes na detecção e segmentação dos Equipamentos de Proteção Individual (EPIs). Dentre os modelos avaliados, o melhor deles alcançou uma precisão média de 96%, o que demonstra o excelente desempenho da rede na tarefa proposta.

Além disso, a análise dos resultados revelou que o YOLOv8 se destaca pela sua capacidade de identificar com precisão certas classes de EPIs. Como observado na tabela 2, as classes de capacete e colete apresentaram 100% de precisão, o que significa que a rede neural conseguiu aprender de maneira eficiente e confiável a distinguir e segmentar corretamente esses objetos, mesmo em cenários variados. Esse nível de precisão é especialmente importante em aplicações de segurança, onde a correta identificação de EPIs é crucial para garantir a integridade dos trabalhadores.

Outro ponto relevante é a robustez do YOLOv8 em termos de processamento em tempo real. Sua alta velocidade de inferência, aliada ao desempenho preciso, torna o modelo ideal para ser implementado em cenários onde a detecção de EPIs precisa ocorrer em tempo real, como em ambientes industriais e de construção civil. A combinação do YOLOv8 com câmeras de vigilância ou outros dispositivos de captura de imagem pode permitir a criação de sistemas automatizados de monitoramento, garantindo que os trabalhadores estejam usando os EPIs adequados durante toda a jornada de trabalho, sem a necessidade de intervenção humana.

Por fim, os resultados obtidos neste estudo reforçam o potencial da rede YOLOv8 como uma ferramenta poderosa não apenas na detecção de EPIs, mas também em outras aplicações que exigem velocidade e precisão em tarefas de visão computacional. A flexibilidade da arquitetura do YOLOv8, aliada à sua capacidade de aprendizado, sugere que ela pode ser aplicada de forma eficaz em uma ampla gama de problemas que envolvem detecção e segmentação de objetos, desde segurança no trabalho até monitoramento de espaços públicos e privados. Com ajustes e treinamentos específicos para outras classes de objetos, o YOLOv8 pode continuar a entregar resultados superiores, sendo uma escolha promissora para soluções baseadas em inteligência artificial no futuro.

Futuramente pretende-se fazer algumas alterações que visam melhorar o desempenho do modelo, como por exemplo: fazer a detecção do uso ou não uso do epi, pois os modelos treinados fazem a detecção do equipamento em si, não do uso dele. Uma alteração que poderia potencializar a precisão de classificação dos modelos, seria fazer a divisão da base de dados (veja o capítulo 4 seção 4.3) a partir da quantificação das classes em cada imagem em toda a base de dados, e, após isso fazer a divisão do dataset em conjuntos de treinamento, teste e validação.

Uma outra alteração que se pode fazer para melhorar o desempenho das redes treinadas seria criar um dataset personalizado para a empresa que fosse utilizar, pois a base de dados utilizada neste projeto tenta abranger todos os tipos e variações de EPIs e acaba se tornando uma rede genérica. Mas sabemos que o padrão de EPIs pode variar de empresa para empresa. Existem empresas que não vão utilizar todos os EPIs que a rede treinada nesse projeto é capaz de identificar. E, também, treinar a rede para uso específico dos equipamentos a serem utilizados pela empresa, pois é de conhecimento geral que dentro de uma empresa eles buscam padronizar os EPIs a serem utilizados pelos funcionários, em relação a cor (o equipamento tende a ser de uma cor específica para aquela empresa), formato (os EPIs podem ter formatos diferentes) e EPIs a serem utilizados (fazer o treinamento somente dos EPIs que serão utilizados). Essas alterações melhoraram consideravelmente a detecção dos equipamentos de segurança individual em cada empresa.

REFERÊNCIAS

ALPAYDIN, E. **Introduction to Machine Learning**. 4th. ed. Cambridge, MA: The MIT Press, 2020. ISBN 9780262043793.

BARLYBAYEV, A. et al. Personal protective equipment detection using yolov8 architecture on object detection benchmark datasets: a comparative study. **Cogent Engineering**, Taylor & Francis, v. 11, n. 1, p. 2333209, 2024.

BIASES, W. . **Weights & Biases: Machine Learning Experiment Tracking**. 2024. Accessed: 2024-10-21. Disponível em: <https://wandb.ai/>.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. 2020. Disponível em: <https://arxiv.org/abs/2004.10934>.

EBERMAM, E.; KROHLING, R. A. Uma introdução compreensiva às redes neurais convolucionais: Um estudo de caso para reconhecimento de caracteres alfabéticos. **Revista Sistemas de Informação FSMA**, n. 21, p. 92–106, 2018. Acesso em: 11 set. 2024. Disponível em: <https://www.fsma.edu.br/si/edicao21/FSMA_SI_2018_1_Principal_08.pdf>.

EMPREGO, M. do Trabalho e. **Norma Regulamentadora NR 6: Equipamentos de Proteção Individual - EPIs**. [S.l.], 1978. Disponível em: <http://www.normas.com.br/NR6>.

EMPREGO, M. do Trabalho e. **Norma Regulamentadora NR 18: Condições e Meio Ambiente de Trabalho na Indústria da Construção**. [S.l.], 1995. Disponível em: <http://www.normas.com.br/NR18>.

EMPREGO, M. do Trabalho e. **Norma Regulamentadora NR 35: Trabalho em Altura**. [S.l.], 2012. Disponível em: <http://www.normas.com.br/NR35>.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2014. p. 580–587.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. [S.l.: s.n.], 2011. p. 315–323.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

JOCHER, G. et al. **YOLOv8**. 2023. <https://github.com/ultralytics/ultralytics>.

JOCHER, G. et al. Yolov5: A comparative analysis with yolov4 and yolov3. **GitHub Repository**, 2021. Disponível em: <https://github.com/ultralytics/yolov5>.

KAGGLE. **Kaggle: Your Machine Learning and Data Science Community**. 2024. Accessed: 2024-10-21. Disponível em: <https://www.kaggle.com/>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, p. 1097–1105, 2012.

KUMAR, A.; SAHOO, G. Automatic detection of personal protective equipment (ppe) compliance using deep learning models. **International Journal of Occupational Safety and Ergonomics**, Taylor Francis, v. 25, n. 4, p. 634–642, 2019.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LI, Z. et al. A survey of convolutional neural networks: Analysis, applications, and prospects. **IEEE Transactions on Neural Networks and Learning Systems**, PP, p. 1–21, 06 2021.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 21–37.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. **Proceedings of the IEEE conference on computer vision and pattern recognition**, p. 3431–3440, 2015.

LUGER, G. F. **Artificial Intelligence: Structures and Strategies for Complex Problem Solving**. 6th. ed. Boston, MA: Pearson Education, 2013. ISBN 9780136070474.

Paromed. **Saiba como evitar problemas causados pelo mau uso de EPIs**. 2024. Acesso em: 18 nov. 2024. Disponível em: <https://www.paromed.com.br/saiba-como-evitar-problemas-causados-pelo-mau-uso-de-epis/>.

PRADHAN, S.; SAMANTARAY, A. Vision-based ppe compliance monitoring system: A comprehensive review. **Journal of Safety Research**, Elsevier, v. 78, p. 223–231, 2021.

REDMON, J.; FARHADI, A. You only look once: Unified, real-time object detection. **Proceedings of the IEEE conference on computer vision and pattern recognition**, p. 779–788, 2016.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018.

REIS, J.; OLIVEIRA, A.; SILVA, P. Improved detection in yolov8: Anchor-free mechanisms and enhanced activation functions. **Journal of Advanced Machine Learning Research**, v. 12, p. 102–115, 2023.

Roboflow. **Roboflow Public Datasets**. 2024. Accessed: 2024-08-29. Disponível em: <https://public.roboflow.com/>.

Roboflow Universe. **Synapsis PPE Dataset**. 2024. Acesso em: 18 nov. 2024. Disponível em: <https://universe.roboflow.com/computer-vision-8kpih/synapsis-ppe>.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

ULTRALYTICS. **YOLOv5**. 2020. <https://github.com/ultralytics/yolov5>.

ULTRALYTICS. **YOLOv6**. 2022. <https://github.com/meituan/YOLOv6>.

ULTRALYTICS. **YOLOv8 Documentation - Ultralytics**. 2024. Accessed: 2024-10-21. Disponível em: <https://docs.ultralytics.com/pt/models/yolov8/#what-are-the-performance-metrics-for-yolov8-models>.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. **arXiv preprint arXiv:2207.02696**, 2022.

WANG, C.-Y. et al. Cspnet: A new backbone that can enhance learning capability of cnn. **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops**, p. 390–391, 2020.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. **European conference on computer vision**, p. 818–833, 2014.