Detecção de armas de fogo por câmeras de segurança em espaços públicos ou privados

João Victor Rosa Tagliarini[1], José Antônio Soares Pinto[1], Julio César Bonow Manoel[1], Kauan Da Silva Vieira[1], Laryssa Gabrielly Marquesin[1], Lucas Fernandes Tolotto[1], Lucas Ribeiro Bonfílio de Lemos[1], Matheus Aparecido de Oliveira Ramos[1]

[1]Engenharia da Computação – Centro Universitário Facens
Rodovia Senador José Ermírio de Moraes, 1425, Castelinho km 1,5
Alto da Boa Vista – Sorocaba/SP – CEP 18087-125

{joaovictorosa.tagliarini, josesoarespinto1, juliobonow, kauanvieira004, lmarquesin, lucasfernandestolotto, lucasrblemos, matheusaparecido101}@gmail.com

Abstract. *In 2022, Brazil recorded 33,580 firearm-related homicides, corresponding to 15.7 killings per 100,000 inhabitants and accounting for 72.4% of all homicides in the country. In this context, this project aims to develop a computer vision system capable of identifying, in real time, the presence of firearms in public or private environments through the detection of weapons in surveillance camera images. The proposal seeks to prevent risks to public safety, protect property, and support security forces. The system uses a dataset composed of real images and includes data augmentation, training, comparison of the YOLOv11, EfficientDet, and Faster R-CNN models, and performance evaluation using precision, recall, F1-score, mAP50, mAP50-95, and inference speed. As a result, the best-performing model was YOLOv11, achieving 91.00%, 81.10%, 85.77%, 88.30%, 68.60%, and 15 ms, respectively.*

Resumo. *Em 2022, o Brasil registrou 33.580 homicídios por armas de fogo, o que representa 15,7 assassinatos por 100 mil habitantes e 72,4% do total de homicídios no país. Diante desse cenário, este projeto tem como objetivo desenvolver um sistema de visão computacional para identificar, em tempo real, a presença de armas de fogo em ambientes públicos ou privados, por meio da detecção em imagens de câmeras de vigilância. A proposta visa prevenir riscos à integridade física da população, proteger o patrimônio e auxiliar as forças de segurança. O sistema utilizará uma base de dados com imagens reais, envolvendo aumentação de dados, treinamento, comparação dos modelos YOLOv11, EfficientDet e Faster R-CNN e avaliação do desempenho pelas métricas de precisão, recall, F1-score, mAP50, mAP50-95 e velocidade de inferência. Como resultados, o modelo de melhor performance foi o YOLOv11, com 91,00%, 81,10%, 85,77%, 88,30%, 68,60% e 15 ms, respectivamente.*

Keywords. *Computer vision, Guns, Security.*

Palavras-chave. *Visão computacional, Armas de fogo, Segurança.*

1. Introdução

No Brasil, em 2022, foram registrados 33.580 homicídios por armas de fogo e uma taxa de 15,7 assassinatos para cada 100 mil habitantes, com aumento crescente desde 2019.

Quanto à proporção geral de mortes, 72,4% dos homicídios no país são cometidos com armas de fogo [Brasil 2024]. Além disso, 14% da população já sofreu com assaltos a mão armada entre 2018 e 2022, aumentando para 21% nas capitais estaduais [Soares 2022].

Em 2023, 10.935 armas foram apreendidas, um aumento de cerca de 29% em comparação ao ano anterior, com 8.466 registros. Dentre as ações da nação para combater a problemática estão o investimento em equipamentos tecnológicos e de inteligência, com integração das forças de segurança pública na troca de informações e participação ativa da população na busca de soluções alternativas, para prevenir as ocorrências dos crimes [Brasil 2024].

Sendo assim, nas últimas décadas, as técnicas de visão computacional e aprendizado profundo têm avançado rapidamente, especialmente no contexto de aplicações em vigilância e segurança pública. Paralelamente, observa-se uma preocupação prática crescente com a latência para atendimento em tempo real, limitações computacionais em *edge devices* e a robustez diante de variações de cena, como iluminação, oclusões e ângulos de câmera, são desafios centrais, assim como o tempo de resposta.

Em vista desse cenário, o objetivo geral é construir um sistema baseado em visão computacional capaz de identificar, em tempo real, a presença de armas de fogo em ambientes públicos ou privados, por detecção de câmeras de vigilância, para prevenir riscos à saúde da população, ao patrimônio, e auxiliar a tarefa das forças de segurança.

Como objetivos específicos, listam-se os seguintes:

- Selecionar e analisar uma base de dados composta por imagens reais capturadas por câmeras de vigilância, contemplando diferentes condições de iluminação, ângulo e ambiente;
- Aplicar técnicas de aumentação de dados que aprimorem a qualidade e a diversidade do conjunto de treinamento;
- Treinar e ajustar os modelos YOLOv11, EfficientDet e Faster R-CNN;
- Comparar o desempenho dos modelos com base nas métricas de precisão, *recall*, *F1-score*, mAP50, mAP50-95 e velocidade de inferência;
- Avaliar a adequação e viabilidade dos modelos em cenários de detecção em tempo real, identificando o mais eficiente para aplicações práticas de segurança.

Como contribuições, espera-se com o projeto auxiliar na ação dos agentes de vigilância, identificando previamente a presença de armas de fogo e agilizando a tomada de decisão para proteção da vida e bens materiais.

2. Fundamentação teórica

Este tópico apresenta os conceitos fundamentais necessários ao entendimento do trabalho. Sendo assim, a visão computacional é uma área da inteligência artificial que busca permitir que máquinas interpretem e compreendam imagens e vídeos de maneira semelhante ao olho humano. Já o aprendizado profundo utiliza redes neurais artificiais com múltiplas camadas para extrair automaticamente características visuais complexas, impulsionando tarefas de detecção e reconhecimento. Na sequência, descreve-se a arquitetura YOLO, EfficientDet e Faster R-CNN e suas principais características, além das métricas de avaliação utilizadas para medir o desempenho dos modelos de detecção de objetos.

2.1. YOLO

O YOLO (*You Only Look Once)* é um modelo de detecção de objetos criado por Joseph Redmon e Ali Farhadi em 2015 que se destacou por realizar localização e classificação em uma única passagem pela rede neural convolucional, tornando-o extremamente rápido e adequado para aplicações em tempo real. Diferentemente de abordagens de duas etapas, que primeiro propõem regiões e depois classificam, o YOLO trata a detecção como um problema de regressão direta, alcançando alta eficiência com menor custo computacional [Redmon et al. 2015].

A arquitetura divide a imagem em uma grade S×S, e cada célula prevê B caixas delimitadoras com coordenadas, dimensões, confiança (presença de objeto + IoU) e C probabilidades de classe, formando um tensor S×S×(B×5 + C). No YOLOv1, utilizando PASCAL VOC, a grade escolhida foi 7×7 com duas caixas por célula e 20 classes (tensor 7×7×30). Inspirado na GoogLeNet, o modelo possui 24 camadas convolucionais, aplica *Non-Maximum Suppression* para remover caixas redundantes e é treinado com uma função de perda que combina erros de localização e classificação [Redmon et al. 2015].

O YOLOv11 trouxe melhorias de precisão e eficiência, alcançando maior mAP no conjunto COCO, com quase 19% menos parâmetros em comparação ao YOLOv8, versão mais estabelecida, na versão nano. Além disso, o modelo foi otimizado para desempenho em tempo real em diversas plataformas, desde dispositivos móveis e embarcados, até sistemas em nuvem, reduzindo o custo computacional e acelerando o tempo de inferência, conforme a Tabela 1 abaixo [Ultralytics 2025].

Tabela 1. Comparação YOLOv11n e YOLOv8n

Modelo	Tamanho imagem (pixels)	mAP50-95	Velocidade CPU ONNX (ms)	Velocidade T4 TensorRT10 (ms)	Parâmetros (milhões)
YOLOv11n	640	39,5	56,1	1,5	2,6
YOLOv8n	640	37,3	80,4	1,47	3,2

2.2. EfficientDet

Como outra abordagem de arquitetura de único estágio, a EfficientDet foi criada nos laboratórios de pesquisa do Google, mais especificamente no *Brain Team*. Dessa maneira, utiliza como base os modelos EfficientNet pré-treinados no ImageNet, que funcionam como *backbone* responsável por extrair mapas de características em diferentes escalas. Esses mapas passam pelo BiFPN (*Bi-directional Feature Pyramid Network*), um módulo de fusão de recursos bidirecional que combina informações de diferentes resoluções, tanto de cima para baixo quanto de baixo para cima. Depois dessa etapa de integração, os recursos resultantes alimentam duas *heads* de predição, uma para classificar objetos e outra para gerar caixas delimitadoras, cujos pesos são compartilhados entre todos os níveis, reduzindo o custo computacional [Tan *et al*. 2020].

Para criar uma linha de modelos que se adaptam a diferentes níveis de *hardware* e demanda, o EfficientDet introduz uma estratégia chamada *compound scaling*. Em vez de ampliar apenas um componente da rede (como a profundidade do *backbone*, o tamanho da imagem ou a quantidade de camadas do FPN), o método propõe ajustar todas as

dimensões relevantes ao mesmo tempo. Essa abordagem conjunta permite aumentar a capacidade do modelo de forma equilibrada, melhorando o desempenho sem desperdiçar recursos [Tan *et al*. 2020].

2.3. Faster R-CNN

A abordagem de duas etapas tem como princípio identificar primeiro as áreas mais prováveis de conter objetos e, depois, realizar a classificação e o ajuste de localização apenas nessas regiões promissoras. Sendo assim, na primeira etapa, o modelo analisa a imagem para gerar regiões de interesse (RoIs), feita por uma sub-rede conhecida como *Region Proposal Network* (RPN). Em seguida, cada região proposta é enviada para duas *heads* de rede: uma de classificação e outra de regressão [Zaidi *et al*. 2021].

A primeira identifica o tipo de objeto ou indica que é fundo; a segunda ajusta as coordenadas da *bounding box* para melhor encaixar o objeto. Essa análise direcionada permite priorizar a precisão, pois o processo em dois passos possibilita extração e refinamento detalhados das características de cada objeto, mas é mais complexa e exige maior poder computacional, resultando em menor velocidade de processamento [Zaidi *et al*. 2021].

Entre as principais variações estão os modelos da família R-CNN (Redes Neurais Convolucionais baseadas em Região). O Faster R-CNN incorporou a geração de regiões à própria rede por meio da RPN, tornando o sistema totalmente integrado. Em geral, como dito anteriormente, essa abordagem é mais lenta que a anterior, porém pode apresentar melhores métricas, principalmente em pequenos objetos e parcialmente obstruídos [Zaidi *et al*. 2021].

2.4. Métricas

Dentre as métricas utilizadas, a precisão mede a proporção de exemplos classificados como positivos que realmente pertencem à classe positiva. Já o *recall* avalia a capacidade do modelo em identificar corretamente todos os exemplos que pertencem à classe positiva O *F1-score* é a média harmônica entre precisão e *recall*, sendo uma métrica que busca equilibrar essas duas medidas [Paddila *et al.* 2021].

A IoU (Interseção sobre União) avalia o grau de sobreposição entre a *bounding box* prevista e a real, indicando a precisão espacial da detecção. A AP (Precisão Média) corresponde à área sob a curva *precision–recall*, sintetizando o equilíbrio entre precisão e *recall* em diferentes limiares de confiança. A mAP (Média da Precisão Média) generaliza esse conceito ao calcular a média das APs de todas as classes do modelo, representando o desempenho global do detector [Paddila *et al.* 2021].

As métricas mAP50 e mAP50–95 especificam esse cálculo considerando, respectivamente, um limiar fixo de IoU de 0,5 e uma variação entre 0,50 e 0,95, o que permite avaliar desde detecções mais simples até aquelas que exigem maior precisão. Além dessas métricas, a velocidade de inferência também é um fator essencial, pois indica o tempo necessário para o modelo processar cada imagem, sendo determinante para aplicações em tempo real [Paddila *et al.* 2021].

3. Trabalhos relacionados

Burnayev et al. (2023) desenvolveram um sistema baseado em *edge computing* para detecção de armas em cenários de conflito. O sistema utilizava um Raspberry Pi 4 Model

B (4 GB) com câmera, fone de ouvido e *power bank*, executando um modelo EfficientDet de visão computacional treinado em 1.588 imagens do Kaggle. O modelo detectava armas e enviava relatórios de texto para a nuvem (Thingspeak), além de emitir um aviso sonoro ao soldado. O projeto foi implementado em Python, usando TensorFlow, Numpy, OpenCV, PIL e Picamera no Google Colab.

O EfficientDet foi escolhido pela capacidade de exportar o modelo em arquivos menores, consumir menos recursos e executar com mais eficiência em dispositivos IoT. Ainda, a rede nas configurações padrões apresentava mAP de 26,41, 3,2 milhões de parâmetros e 36 ms de tempo médio de inferência em ambiente *mobile*. Nos testes, o protótipo desenvolvido demorava 1,48 s para realizar todo o processo e trafegava para a plataforma *cloud* apenas 53kB [Burnayev *et al.* 2023].

Em continuidade, Ahmed *et al.* (2022) desenvolveram um artigo sobre detecção de armas em tempo real por câmeras de vigilância, usando Scaled-YOLOv4 em dois ambientes diferentes. Em um cenário de aplicação *desktop* com acesso a maior processamento e GPU RTX 2080Ti, foram usadas 8.727 imagens, separadas em 88% treino e 12% teste, com duas classes rotuladas, sendo armas e não-armas, esta última utilizada para diferir objetos de manejo similares.

Como pré-processamento, foram utilizadas técnicas de *data augmentation* para melhorar a generalização, como redimensionamento, equalização, rotação, espelhamento e alteração na escala de cor. Dessa forma, obteve-se como resultados 91,2 de mAP, 90% de precisão, 91% de *recall* e *F1-score*, conseguindo entregar a detecção a uma taxa de 85,7 quadros por segundo (FPS) [Ahmed *et al*. 2022].

Posteriormente, esse modelo passou por um processo de otimização pela ferramenta TensorRT Network Optimization para poder ser implementado em uma placa Jetson Nano, o que fez a performance diminuir para 4,26 FPS e queda pequena das outras métricas, não descritas pelo trabalho, porém, foi observado que houve maior confusão de objetos que não são armas com a classe de arma [Ahmed *et al*. 2022].

Por fim, Thakur *et al.* (2024) também seguiu a mesma lógica de detecção em tempo real, porém utilizando o YOLOv8. A base de dados foi combinada de várias fontes, dividida em arma e não-arma, e também passou por *data augmentation* com rotação, redimensionamento, espelhamento, alterações de brilho, saturação e contraste. Os resultados podem ser visualizados na Tabela 2 a seguir, que vão decrescendo quanto mais se aumenta IoU, indicando que o modelo se torna mais rigoroso na correspondência entre as predições e as anotações reais.

Tabela 2. Resultados de Thakur *et al*

IoU	Precisão	*Recall*	*F1-score*	mAP
0,5	0,85	0,80	0,82	0,78
0,55	0,83	0,78	0,80	0,76
0,60	0,80	0,75	0,77	0,74
0,65	0,78	0,72	0,75	0,71
0,70	0,75	0,70	0,72	0,68

Sobre o tempo de inferência, realizou-se testes com diferentes tamanhos de imagem,

sendo 320x240 pixels com aproximadamente 18 ms, 640x480 com 30 ms, 1280x720 com 60 ms, 1920x1080 com 100 ms e 2560x1440 com 160 ms [Thakur *et al*. 2024]. O tempo de resposta apresentado é satisfatório, assim como o mAP que é muito superior ao do trabalho analisado com o modelo EfficientDet, entretanto, os autores destacam que os detectores apresentam sensibilidade reduzida a objetos pequenos em imagens com *zoom out*, degradação de performance diante de oclusões parciais e queda de confiança em condições de baixa luminosidade, sendo esse um ponto de atenção importante na confecção do presente trabalho.

Portanto, com base nos trabalhos analisados, observa-se que os modelos da família YOLO apresentam o melhor equilíbrio entre velocidade e precisão, sendo mais adequados para aplicações de detecção em tempo real. Assim, optou-se pelo uso não só do YOLOv11 por incorporar avanços recentes que podem ampliar a eficiência e a capacidade de generalização, mas também da EfficientDet, por motivos de comparação. O Faster R-CNN será utilizado representando os modelos de dois estágios, o que permitirá demonstrar a vantagem dos detectores de estágio único em cenários que exigem baixa latência e alta responsividade.

4. Materiais e métodos

Esta seção discorre sobre as tecnologias utilizados no projeto, a metodologia e o desenvolvimento da aplicação em si.

4.1. Materiais

Para o desenvolvimento, foi empregada a linguagem de programação Python, executada em ambiente de notebook local com 16 GB de memória RAM e GPU NVIDIA GeForce GTX 1060 (6 GB). Durante o desenvolvimento, utilizou-se as bibliotecas PyTorch, Torchvision, Ultralytics, EffDet, Albumentations, OpenCV, NumPy e Matplotlib.

O modelo YOLOv11n foi implementado por meio das ferramentas oficiais da Ultralytics, que fornecem suporte completo para treinamento, inferência e avaliação de desempenho. Para a EfficienteDet, o modelo foi implementado utilizando a biblioteca EffDet, que disponibiliza as arquiteturas oficiais, sendo escolhida a tf_efficientdet_d1. Já o modelo Faster R-CNN, adotado como referência de abordagem de dois estágios, foi utilizado em sua versão Faster R-CNN ResNet-50 FPN, disponibilizada pela biblioteca Torchvision Models.

Todos os modelos foram treinados e avaliados sobre um conjunto de dados personalizado, criado com a junção de oito *datasets* de armamento em diversas situações, inclusive já com imagens de câmeras de segurança, fornecidas pela plataforma Kaggle.

4.2. Metodologia

Inicialmente, obteve-se os oito conjuntos de dados, realizando um processo extenso de adequação ao tema do projeto. Dessa forma, foi necessário excluir imagens e classes que não fossem armas dos *datasets,* por meio de *script* desenvolvido em Python. Além disso, ajustou-se as notações de cada imagem para o formato esperado pela YOLO, junto da adequação nos diretórios e configuração do arquivo YAML.

Em continuidade, o conjunto completo resultou em 21.451 imagens, divididas em três pastas, *train, val* e *test*, conforme a proporção 70%, 20% e 10%. Por outro lado, para o treinamento dos outros modelos, foi necessário converter o conjunto de dados

originalmente anotado no formato YOLO para o formato COCO, adotado como padrão pelos outros *frameworks* utilizados na implementação.

Como técnicas de *data augmentation* para melhorar a generalização dos modelos e simular as restrições do ambiente em que a câmera de vigilância estaria, aplicou-se transformações no espaço de cor por meio do modelo HSV, ajustando a tonalidade (*hue*), saturação (*saturation*) e brilho (*value*) das imagens com frações de 0,015, 0,7 e 0,4, respectivamente. Essas variações contribuem para aumentar a robustez do modelo diante de diferentes condições de iluminação e coloração.

No domínio geométrico, aplicaram-se translações de até 10% da dimensão da imagem, variações de escala de até 50% e uma probabilidade de 0,5 para o espelhamento horizontal (*horizontal flip*). Entre as técnicas compostas, adotou-se o *mosaic augmentation* com probabilidade total (1,0), que combina quatro imagens distintas em uma única amostra, promovendo maior variabilidade contextual e enriquecendo a diversidade espacial.

Por último, implementou-se o *random erasing*, com probabilidade de 0,4, que remove aleatoriamente pequenas regiões da imagem, incentivando o modelo a se concentrar em características mais discriminativas. Ainda, as imagens foram submetidas a um recorte parcial (*crop fraction*) de 0,8, garantindo a manutenção de uma proporção significativa da área original, mas com variações estruturais suficientes para enriquecer o conjunto de treinamento.

Para o treinamento do modelo YOLOv11n, foram estabelecidos parâmetros voltados ao equilíbrio entre desempenho e custo computacional. O processo foi configurado para 100 épocas de treinamento, *batch size* de 16, valor que proporciona boa estabilidade no cálculo dos gradientes sem comprometer a utilização de memória da GPU. As imagens foram redimensionadas para 480×480 pixels e o modelo inicializado com pesos pré-treinados, permitindo que a rede se beneficiasse de representações previamente aprendidas, acelerando a convergência e melhorando o desempenho inicial.

O otimizador foi configurado como *Stochastic Gradient Descent* (SGD), com a taxa de aprendizado inicial definida em 0,01, que determina o tamanho do ajuste feito nos pesos a cada atualização, orientando o quanto o modelo se move na direção de redução do erro. Além disso, empregou-se *momentum* de 0,937, auxiliando na suavização das atualizações dos gradientes e na aceleração da convergência, e *weight decay* de 0,0005, atuando como regularizador para mitigar o *overfitting* e dar maior estabilidade ao treinamento.

Para os outros modelos, buscou-se manter a coerência experimental em relação às configurações empregadas na YOLO, de modo a permitir uma comparação justa entre as arquiteturas. Contudo, algumas adaptações foram necessárias em virtude das limitações de *hardware* enfrentadas e diferenças estruturais entre as arquiteturas, como, por exemplo, as técnicas de *data augmentation* que precisaram ser feitas antes do treinamento, com a biblioteca Albumentations.

Em sequência, para a EfficientDet, configurou-se 20 épocas, *batch size* de 4, tamanho da imagem de 640x640 pixels, otimizador SGD, *learning rate* de 0,001, *momentum* 0,9, *weight decay* de 0,0005 e novamente a inicialização feita com os pesos pré-treinados. Já para a Faster R-CNN, a única diferença para a rede anterior foi uma taxa de aprendizado maior, de 0,005 para acelerar o ajuste dos pesos, já que o modelo costuma

convergir de forma mais lenta.

Quanto ao dimensionamento dos conjuntos de dados, ajustes adicionais foram necessários em razão das especificidades de cada arquitetura e dos recursos computacionais disponíveis. Sendo assim, a YOLO foi o único modelo capaz de operar com o conjunto integral, 15.015 imagens para treinamento, 4.290 para validação e 2.146 para teste, durante um período de execução mais longo, sem esgotar a memória. No caso da EfficientDet e da Faster R-CNN, adotou-se uma redução significativa do volume de dados, restringindo-o a 1.500 imagens para treinamento, 100 para validação e 100 para teste, de modo a evitar excedentes de memória e assegurar a execução completa.

Por fim, o treinamento foi conduzido exclusivamente com o conjunto de treino, enquanto o conjunto de validação foi utilizado para o monitoramento do desempenho e ajuste dos hiperparâmetros ao longo das épocas. Após a finalização do treinamento, os modelos foram avaliados de forma crítica e comparativa sobre o conjunto de teste, considerando as métricas de precisão, *recall*, *F1-score*, mAP50, mAP50-95 e tempo médio de inferência, a fim de mensurar tanto a qualidade das detecções quanto a eficiência computacional das arquiteturas.

5. Resultados

A seguir, a Tabela 3 exibe os resultados das métricas obtidas pelos modelos no conjunto de teste, sem *data augmentation*, já que não apresentaram diferenças significativas.

Tabela 3. Comparação dos resultados dos modelos

Modelo	mAP50	mAP50-95	Precisão	*Recall*	*F1-score*	Inferência (ms/img)
YOLOv11	88,30%	68,60%	91,00%	81,10%	85,77%	15
EfficientDet	17,90%	7,80%	10,00%	12,00%	10,91%	47,80
Faster R-CNN	84,20%	57,60%	64,70%	65,00%	64,85%	225,28

O modelo YOLOv11n apresenta desempenho substancialmente superior nas métricas de detecção, atingindo 88,30% de mAP50 e 68,60% de mAP50-95, além de precisão e *recall* elevados, de 91,00% e 81,10%, respectivamente. Esses valores indicam que o modelo possui boa capacidade de localizar e classificar corretamente o objeto-alvo mesmo em diferentes escalas e condições de imagem. O tempo médio de inferência de apenas 15 ms reforça sua adequação para aplicações em tempo real, aspecto essencial no contexto de vigilância inteligente.

Em contraste, o desempenho da EfficientDet foi consideravelmente inferior, com mAP50 de 17,90%, mAP50-95 de 7,80% e métricas de precisão e *recall* extremamente baixas (10,00% e 12,00%). Essa limitação está associada principalmente à restrição imposta pelo volume reduzido de amostras disponível para o treinamento (apenas 1.500 imagens), uma vez que arquiteturas baseadas na família EfficientDet geralmente dependem de grandes quantidades de dados e recursos computacionais mais amplos para alcançar bom desempenho. O tempo de inferência intermediário (47,8 ms) demonstra eficiência razoável, mas insuficiente para compensar o baixo desempenho preditivo.

Já o Faster R-CNN apresentou desempenho intermediário entre as duas arquiteturas. Com mAP50 de 84,20% e mAP50-95 de 57,60%, o modelo demonstrou boa capacidade

de detecção, embora inferior à YOLOv11n em todos os indicadores. As métricas de precisão (64,70%) e recall (65,00%) revelam maior propensão a falsos positivos e falsos negativos quando comparado à YOLO, comportamento esperado para métodos de dois estágios quando treinados com volume reduzido de dados. Além disso, o tempo de inferência de 225,28 ms evidencia sua limitação para aplicações de tempo real, reforçando a característica intrínseca desse tipo de arquitetura ser mais custosa computacionalmente.

Como uma visão mais detalhada do treinamento, a Figura 1 abaixo mostra o comportamento da métrica mAP50-95 ao longo da porcentagem de progresso do treino.

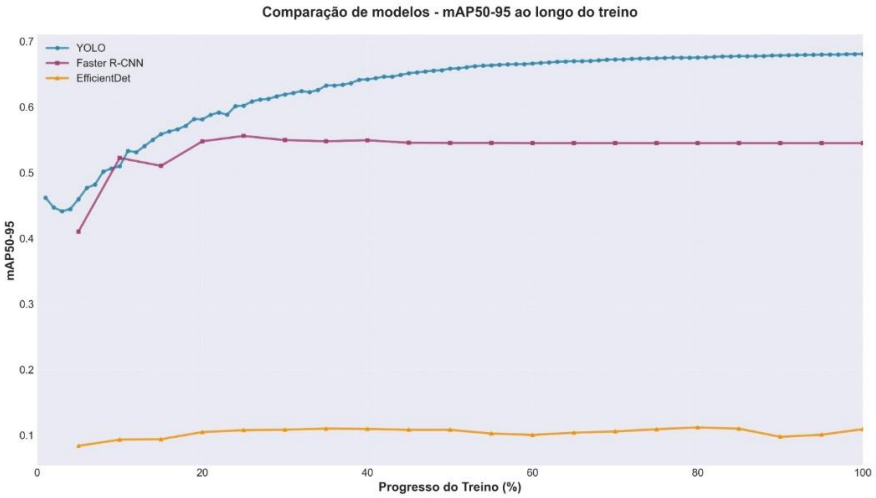

Figura 1. mAP50-95 ao longo do treino.

A curva da YOLO apresenta crescimento contínuo e gradual após as primeiras iterações, indicando um processo de aprendizado estável, com flutuações mínimas e convergência progressiva até atingir um platô próximo ao final do treino. Já o Faster R-CNN demonstra uma evolução inicial relativamente rápida até cerca de 20% do progresso, seguida de estabilização precoce, permanecendo praticamente constante sem avanços significativos, o que sugere limitação de capacidade ou insuficiência de dados para melhorar além desse ponto. Em contraste, a EfficientDet apresenta a curva mais instável e de baixa amplitude, caracterizada por pequenas oscilações e ausência de crescimento consistente, reforçando que o modelo não conseguiu encontrar representações robustas ao longo do treinamento.

6. Conclusão

De forma geral, os resultados destacam a superioridade técnica da YOLOv11n no cenário investigado, combinando alta acurácia e velocidade de execução, o que a torna particularmente apropriada para sistemas embarcados e aplicações de vigilância contínua. Por outro lado, as restrições impostas pelo *hardware* impactaram negativamente o desempenho da EfficientDet e do Faster R-CNN, sobretudo pela redução drástica do conjunto de treinamento, comprometendo sua capacidade de generalização. Esses achados reforçam a importância da compatibilidade entre arquitetura, tamanho do conjunto de dados e recursos computacionais disponíveis, evidenciando que modelos mais leves e otimizados, como YOLOv11n, tendem a apresentar melhor custo-benefício em ambientes com limitação de *hardware* e necessidade de resposta em tempo real.

Como perspectivas para trabalhos futuros, recomenda-se ampliar o conjunto de

dados utilizado pelos modelos de dois estágios e pela EfficientDet, bem como o período de treinamento, de modo a avaliar seu potencial pleno sem as restrições. Além disso, a investigação da implementação da YOLO em dispositivos embarcados, como microcontroladores com aceleração dedicada ou câmeras inteligentes de vigilância, configura uma direção promissora para validar o modelo em condições operacionais reais e avaliar sua viabilidade em sistemas de detecção distribuída e de baixo consumo energético.

Referências

AHMED, Soban *et al.* Development and Optimization of Deep Learning Models for Weapon Detection in Surveillance Videos. 07 jun. 2022. Disponível em: https://www.mdpi.com/2076-3417/12/12/5772. Acesso em: 11 out. 2025.

BRASIL. Fórum Brasileiro de Segurança Pública. Homicídios por arma de fogo e falha na fiscalização no Brasil. 10 jul. 2024. Disponível em: https://fontesegura.forumseguranca.org.br/homicidios-por-arma-de-fogo-e-falha-na-fiscalizacao-no-brasil/. Acesso em: 05 out. 2025.

BRASIL. Secretaria de Comunicação Social. Número de apreensões de arma de fogo em 2023 cresce 28% em relação a 2022. 16 jun. 2024. Disponível em: https://www.gov.br/secom/pt-br/acesso-a-informacao/comunicabr/noticias/numero-de-apreensoes-de-arma-de-fogo-em-2023-cresce-28-em-relacao-a-2022. Acesso em: 05 out. 2025.

BURNAYEV, Zufar R. *et al.* Weapons Detection System Based on Edge Computing and Computer Vision. International Journal of Advanced Computer Science and Applications, v. 14, n. 5, p. 812-820, 2023.

PADILLA, Rafael *et al*. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. 25 jan. 2021. Disponível em: https://doi.org/10.3390/electronics10030279. Acesso em: 13 nov. 2025.

REDMON, Joseph *et al*. You Only Look Once: Unified, Real-Time Object Detection. 08 jun 2015. Disponível em: https://arxiv.org/pdf/1506.02640. Acesso em 11 out. 2025.

SOARES, Rafael. Globo. Pesquisa Ipec/O GLOBO: 14% dos brasileiros foram vítimas de assaltos com arma de fogo nos últimos quatro anos. 10 set. 2022. Disponível em: https://oglobo.globo.com/blogs/pulso/post/2022/09/pesquisa-ipeco-globo-14percent-dos-brasileiros-foram-vitimas-de-assaltos-com-arma-de-fogo-nos-ultimos-quatro-anos.ghtml. Acesso em: 05 out. 2025.

TAN, Mingxing *et al*. EfficientDet: Scalable and Efficient Object Detection. 27 jul. 2020. Disponível em: https://arxiv.org/pdf/1911.09070. Acesso em: 24 nov. 2025.

THAKUR, Ayush *et al*. Real-Time Weapon Detection Using YOLOv8 for Enhanced Safety. 23 out. 2024. Disponível em: https://arxiv.org/pdf/2410.19862. Acesso em: 11 out. 2025.

ULTRALYTICS. YOLO11 vs YOLOv8: Comparação Detalhada. 2025. Disponível em: https://docs.ultralytics.com/pt/compare/yolo11-vs-yolov8/. Acesso em: 11 out. 2025.

ZAIDI, Syed Sahil Abbas *et al*. A Survey of Modern Deep Learning based Object Detection Models. 12 maio 2021. Disponível em: https://arxiv.org/pdf/2104.11892. Acesso em: 13 nov. 2025.