Fundação Getulio Vargas
Escola de Matemática Aplicada
Curso de Especialização em Inteligência Artificial para Aplicações Militares

Aplicação de visão computacional para detecção de ameaças em tempo real durante operações militares de fuzileiros navais: utilização do modelo YOLOv11 para detecção de Carros de Combate e Viaturas Blindadas.

Alexandre Batista da Silva Filho
Paulo Vitor do Amaral Gomes

Rio de Janeiro - Brasil
2025

Fundação Getulio Vargas
Escola de Matemática Aplicada
Curso de Especialização em Inteligência Artificial para Aplicações Militares

Aplicação de visão computacional para detecção de ameaças em tempo real durante operações militares de fuzileiros navais: utilização do modelo YOLOv11 para detecção de Carros de Combate e Viaturas Blindadas.

Declaramos ser os únicos autores do presente projeto de monografia, o qual se refere ao plano de trabalho a ser executado para a continuidade da monografia. Afirmamos não ter recorrido a qualquer forma de colaboração ou auxílio de terceiros para sua realização, salvo nos casos e para os fins expressamente autorizados pelo professor orientador.

Alexandre Batista da Silva Filho

Paulo Vitor do Amaral Gomes

Rio de Janeiro - Brasil

2025

Fundação Getulio Vargas
Escola de Matemática Aplicada
Curso de Especialização em Inteligência Artificial para Aplicações Militares

Aplicação de visão computacional para detecção de ameaças em tempo real durante operações militares de fuzileiros navais: utilização do modelo YOLOv11 para detecção de Carros de Combate e Viaturas Blindadas.

Trabalho de Conclusão de Curso apresentado à Escola de Matemática Aplicada como requisito parcial para continuidade ao trabalho de monografia.

Aprovado em _____ de _____ de _____

Grau atribuído ao Projeto de Monografia: _____

Professor Orientador
Escola de Matemática Aplicada
Fundação Getulio Vargas

Sumário

1 INTRODUÇÃO

1.1 Contextualização

As operações militares contemporâneas são definidas por uma complexidade crescente, onde a velocidade da informação e a capacidade de resposta imediata são fatores decisivos para o sucesso. Neste cenário, a detecção precoce de ameaças móveis, como carros de combate e veículos blindados, emerge como um requisito crítico. A habilidade de identificar e classificar alvos com rapidez e precisão não apenas otimiza o emprego de recursos, mas, fundamentalmente, garante a segurança das tropas em campo, permitindo a tomada de decisões táticas mais eficazes.(1; 2)

Como força de projeção de poder naval, o Corpo de Fuzileiros Navais (CFN) da Marinha do Brasil atua em uma vasta gama de ambientes operacionais, exigindo que sua doutrina e seus meios estejam em constante evolução. Para o CFN, a consciência situacional aprimorada é um multiplicador de poder, e a integração de tecnologias de visão computacional representa um caminho promissor para alcançar essa vantagem. A implementação de sistemas inteligentes de detecção pode oferecer um diferencial tático significativo, elevando o nível de proteção dos fuzileiros navais em missões críticas.(3; 4)

1.2 Meios blindados no combate moderno

O termo "blindado"descreve genericamente veículos terrestres dotados de couraça, projetados para sobreviver no campo de batalha(5). Sua história remonta à Primeira Guerra Mundial, quando o Mark I britânico surgiu em 1915 como uma solução para romper a estagnação da guerra de trincheiras(6). Desde então, os blindados evoluíram drasticamente em poder de fogo, mobilidade e proteção, consolidando-se como elementos indispensáveis nas forças armadas modernas(7). Para os propósitos deste trabalho, distinguem-se dois tipos principais:

- **Carros de Combate (CC)**: Plataformas de combate ofensivas, armadas com canhões de grosso calibre, cuja função principal é engajar e destruir forças inimigas, especialmente outros blindados e fortificações.(5; 7)
- **Viaturas Blindadas de Transporte de Pessoal (VBTP)**: Veículos focados na mobilidade e proteção de tropas ou material no campo de batalha, garantindo que os combatentes cheguem ao seu destino com segurança para cumprir a missão.(5)

A importância estratégica desses meios foi demonstrada em inúmeros conflitos, como na Guerra do Golfo (1990-1991), onde a superioridade blindada foi decisiva(8). Contudo, cenários recentes, como o da Ucrânia, reforçam uma lição vital: a sobrevivência e a eficácia no campo de batalha moderno dependem intrinsecamente da capacidade de detectar o inimigo primeiro. A detecção antecipada de blindados é, portanto, o passo inicial que habilita todas as ações subsequentes, sejam elas defensivas, evasivas ou ofensivas (9).

1.3 Evolução da visão Computacional

A Inteligência Artificial (IA) consolidou-se como uma das áreas mais dinâmicas da ciência da computação. Embora seus fundamentos remontem ao século XIX, o crescimento exponencial da área ocorreu a partir dos anos 2000, com avanços que a levaram dos laboratórios à integração em diversos setores econômicos, industriais e de defesa (10). Suas aplicações incluem reconhecimento de voz, biometria, análise preditiva e gerenciamento logístico, sendo natural sua extensão ao domínio militar (11).

A visão computacional, ramo da IA voltado à extração de informações úteis a partir de imagens estáticas ou dinâmicas, tem por objetivo a interpretação visual automatizada do ambiente. Inicialmente, utilizavam-se abordagens baseadas em características, como bordas, cantos e texturas. Métodos como o SIFT *(Scale-Invariant*

Feature Transform) e o HOG (*Histogram of Oriented Gradients*) foram marcos na detecção de objetos sob condições controladas (12; 13).

A introdução das Redes Neurais Convolucionais (CNNs) revolucionou a área, permitindo a extração automática de atributos relevantes a partir de grandes volumes de dados. Modelos como o R-CNN, Fast R-CNN e Faster R-CNN representam a evolução de abordagens em dois estágios, nas quais a identificação de regiões e a classificação ocorrem separadamente (14; 15).

Em contrapartida, o modelo YOLO (*You Only Look Once*) trouxe uma abordagem de estágio único, executando simultaneamente a localização e a classificação dos objetos. Essa arquitetura permite a detecção em tempo real com notável redução de custo computacional (16). Desde sua introdução em 2015, o YOLO evoluiu significativamente, alcançando com o YOLOv4, em 2020, resultados superiores aos modelos de dois estágios, o que o consolidou como solução de excelência para aplicações em tempo real (17).

1.4 Justificativa

A aplicação da visão computacional no contexto militar, conforme apresentado, já é uma realidade consolidada, com sistemas de Aeronaves Remotamente Pilotadas (ARPs) e de vigilância autônoma sendo empregados para inteligência e reconhecimento(18; 19). No entanto, muitas dessas soluções são genéricas ou desenvolvidas como sistemas proprietários por nações estrangeiras, o que pode limitar sua adaptação às doutrinas, aos meios e aos cenários de interesse específicos do Corpo de Fuzileiros Navais (CFN) do Brasil.

Conflitos recentes, como o da Ucrânia, têm demonstrado de forma inequívoca que a capacidade de detectar, identificar e neutralizar ameaças blindadas com rapidez e precisão é um fator decisivo no campo de batalha moderno. A dependência de observação humana ou de sistemas de detecção tradicionais pode se mostrar insuficiente diante da velocidade e da sofisticação das ameaças atuais. Identifica-se, portanto, uma lacuna na disponibilidade de uma ferramenta de baixo custo, alta performance

e customizada para as necessidades da Marinha do Brasil, que possa ser integrada a diferentes plataformas de observação.

Este trabalho se justifica pela necessidade de investigar e desenvolver uma possível solução em detecção de objetos para suprir essa carência. Ao propor o emprego do YOLOv11 para a detecção de carros de combate e blindados, a pesquisa busca não apenas validar uma tecnologia de ponta, mas também:

- **Aumentar a Consciência Situacional**: Fornecer às tropas do CFN uma capacidade aprimorada de identificar ameaças em tempo real, aumentando a segurança e a eficácia em operações.
- **Contribuir para a Autonomia Tecnológica**: Desenvolver conhecimento aplicado em uma área de defesa estratégica, reduzindo a dependência de tecnologias externas e fomentando a inovação nacional.
- **Gerar um Protótipo Funcional**: Criar um modelo de detecção especializado em identificar blindados e carros de combate, que sirva como base para futuros sistemas de vigilância, embarcados em viaturas ou ARPs da Força.

Dessa forma, a relevância desta pesquisa reside em sua aplicação prática e estratégica, abordando um problema operacional real com uma solução tecnológica avançada e alinhada às necessidades do Corpo de Fuzileiros Navais.

1.5 Objetivos

1.5.1 Objetivo Geral

Desenvolver um modelo de visão computacional baseado na arquitetura YOLO, com aprendizado customizado, para realizar a detecção em tempo real de blindados e veículos militares durante operações militares de fuzileiros navais, visando apoiar a identificação de ameaças e a tomada de decisão no campo de batalha.

1.5.2 Objetivos Específicos

- Revisar a literatura sobre aplicações de visão computacional e modelos customizados usando o YOLO
- Coletar e preparar um conjunto de dados representativo com imagens de blindados e veículos militares em diferentes ambientes operacionais.
- Treinar e ajustar um modelo YOLO com base nesse conjunto de dados.
- Avaliar o desempenho do modelo em termos de acurácia, velocidade de detecção e robustez frente a diferentes condições de campo.

2 Revisão da Literatura

2.1 Conceitos relacionados

Para compreensão dos experimentos e análises apresentados neste trabalho, é essencial compreender alguns conceitos fundamentais relacionados à aprendizagem de máquina e visão computacional. A seguir, são apresentados os principais termos utilizados:

- **Dataset:** Conjunto estruturado de dados utilizados no treinamento, validação e teste de modelos de aprendizado de máquina. Em tarefas de visão computacional, os *datasets* são compostos geralmente por imagens anotadas com rótulos ou coordenadas que representam os objetos de interesse.
- **Transfer Learning:** Técnica que consiste em reutilizar modelos pré-treinados em grandes *datasets* para resolver novos problemas, geralmente com menos dados disponíveis. Essa abordagem permite reduzir o tempo de treinamento e melhorar a performance em tarefas específicas.
- **Hiperparâmetros:** Parâmetros definidos antes do processo de treinamento, como taxa de aprendizado (*learning rate*), número de épocas e tamanho do *batch*. Eles influenciam diretamente o desempenho e a velocidade de convergência do modelo.
- **Épocas (*Epochs*):** Representam o número de vezes que o modelo percorre todo o conjunto de dados de treinamento. Mais épocas podem significar mais aprendizado, mas também aumentam o risco de *overfitting*.
- **Batchs:** Subconjuntos do *dataset* utilizados em cada iteração do processo de treinamento. O uso de *batchs* permite treinar modelos com maior eficiência computacional e estabilidade.
- **Acurácia:** Métrica que indica a proporção de previsões corretas feitas pelo

modelo em relação ao total de previsões. É útil em problemas com classes balanceadas.

- **Precisão (*Precision*):** Mede a proporção de verdadeiros positivos (VP) entre todas as previsões positivas feitas pelo modelo:

$$\text{Precisão} = \frac{VP}{VP + FP}$$

- **Recall:** Mede a capacidade do modelo em identificar corretamente todas as instâncias da classe positiva:

$$\text{Recall} = \frac{VP}{VP + FN}$$

- **F1-Score:** Média harmônica entre *precisão* e *recall*, útil para avaliar o equilíbrio entre essas métricas:

$$F1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}$$

- **mAP (*mean Average Precision*):** Métrica usada em tarefas de detecção de objetos que representa a média das precisões obtidas em diferentes limiares de *IoU* (*Intersection over Union*), como o *mAP@0.5* e *mAP@0.5:0.95*.

- **Matriz de Confusão (*Confusion Matrix*):** Tabela que apresenta o desempenho de um modelo de classificação, detalhando as quantidades de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN). Permite identificar erros específicos por classe e avaliar o comportamento geral do modelo.

2.2 Fontes de Consulta

A elaboração deste trabalho foi sustentada por uma base bibliográfica composta por fontes doutrinárias e técnicas confiáveis, alinhadas aos objetivos e à natureza da

pesquisa.

O embasamento doutrinário foi obtido a partir dos manuais da série CGCFN (Comando-Geral Corpo de Fuzileiros Navais), da Base Doutrinária do Exército Brasileiro (BDEx) e do Portal de Periódicos da Marinha do Brasil. Esses documentos fornecem os fundamentos operacionais e conceituais relacionados ao emprego de meios militares no contexto das Forças Armadas brasileiras.

O embasamento técnico, por sua vez, foi construído com o apoio de plataformas acadêmicas e científicas de alto rigor, como o *IEEE Xplore Digital Library* e o *Google Scholar*. Nessas fontes, foram selecionados artigos recentes e relevantes sobre visão computacional, modelos de detecção de objetos, *deep learning*, e aplicações do algoritmo YOLO em contextos militares e civis.

A combinação dessas fontes assegurou a solidez teórica e a atualidade técnica necessárias para o desenvolvimento e a validação do trabalho proposto.

2.3 Trabalhos Relacionados

O reconhecimento em tempo real de alvos militares, incluindo Carros de Combate (CC) e viaturas blindadas, em cenários operacionais de alta complexidade é uma área de estudo que tem evidenciado um avanço notável aplicação de arquiteturas baseadas em redes neurais convolucionais, particularmente o modelo YOLO (*You Only Look Once*). Diversos trabalhos recentes foram desenvolvidos abordando estratégias de refinamento de desempenho, limitações técnicas, inovações estruturais e implicações estratégicas para a detecção automatizada de ameaças blindadas no campo de batalha moderno.

Lytvyn et al.(18) propôs uma abordagem centrada na criação de um conjunto de dados personalizado contendo imagens de variantes dos CC russos T-80 e T-90. As imagens foram obtidas em condições ambientais heterogêneas com diferentes estações, períodos diurnos e noturnos e apresentavam variados ângulos e níveis de resolução, conferindo robustez ao *dataset*. A plataforma Roboflow foi empregada para

a rotulação manual e a implementação de técnicas de *data augmentation*. O modelo YOLO treinado a partir desse conjunto atingiu acurácia de 86%, o que, embora promissor, revelou-se sensível a condições adversas de visibilidade e ruído. A principal contribuição reside na demonstração empírica de que *datasets* cuidadosamente curados podem incrementar substancialmente a acurácia e a generalização de modelos aplicados a domínios militares. A importância deste trabalho para o presente trabalho está em estabelecer um precedente metodológico para a criação e validação do conjunto de dados que será utilizado para treinar e avaliar o modelo YOLOv11, reforçando a necessidade de diversidade e robustez nas imagens coletadas.

Jafarzadeh et al.(20) abordou o problema da escassez de dados específicos do domínio militar por meio três estratégias: o uso de dados de fonte aberta, aliados aos dados reais obtidos e posteriormente aumentando o dataset com o uso de *data augmentation*, totalizando 10.064 imagens rotuladas, além do uso de *transfer learning*, pela implementação de um modelo YOLOv5 pré-treinado com o *COCO dataset*. Com isso, os autores demonstraram que o uso de *data augmentation* elevou o mAP em 9%, atingindo mais de 95% após 100 épocas. O estudo se destaca por testar o modelo em uma plataforma de baixo custo, o *Raspberry Pi*, evidenciando a viabilidade de aplicações embarcadas de detecção em tempo real. Este estudo é fundamental para esta pesquisa, pois valida o uso de *transfer learning* e *data augmentation* como estratégias viáveis para superar a limitação de dados, uma abordagem que poderá ser adotada no treinamento do YOLOv11. Adicionalmente, demonstra a aplicabilidade em *hardware* de baixo custo, alinhada ao conceito de emprego em diversos ambientes operacionais.

A pesquisa de Sikandar et al.(21) promoveu uma comparação sistemática entre seis variantes da arquitetura YOLO: YOLOv3, YOLOv4 e quatro configurações da versão YOLOv5 (s, m, l e xl). O estudo utilizou um *dataset* de 922 imagens rotuladas em duas classes: CCs e bandeiras, e conduziu experimentos padronizados para comparação. Os resultados mostraram que todas as versões YOLOv5 superaram 95% de mAP, sendo a versão YOLOv5xl a mais precisa (mAP de 99%), mas com exigência computacional elevada. A análise dos resultados apresentados evidencia a

necessidade de ponderar entre acurácia e consumo de recursos, aspecto crucial em ambientes operacionais com restrições de *hardware*. Esse trabalho, ao oferecer uma base comparativa rigorosa, fornece subsídios relevantes para a escolha criteriosa de modelos conforme os requisitos operacionais e os perfis de missão específicos. Para esta pesquisa, a metodologia de Sikandar et al. serve como um *benchmark* essencial, oferecendo uma linha de base de desempenho contra a qual o YOLOv11 poderá ser comparado, permitindo uma análise aprofundada sobre o balanço entre precisão e eficiência computacional da nova arquitetura.

Li et al.(22) introduziu modificações arquitetônicas ao YOLOv5s, culminando na proposta do YOLOv5-SC, adaptado para análise de imagens captadas por Aeronaves Remotamente Pilotadas (ARPs). A arquitetura incorporou o módulo de atenção SimAM ao *backbone* e substituiu o mecanismo de interpolação do *neck* pelo módulo CARAFE (*Content-Aware ReAssembly of Features*). Tais mudanças propiciaram ganho de precisão sem comprometer a eficiência computacional. Utilizando validação cruzada *k-fold* em um *dataset* de 1.000 imagens públicas, os autores obtiveram precisão de 96,42% e recall de 94,24%. O estudo evidencia o potencial de intervenções arquitetônicas moduladas por heurísticas específicas para otimizar o desempenho de detecção de veículos blindados. A utilização de mecanismos de atenção destaca-se como uma tendência relevante para aumentar a seletividade do modelo diante de ruído contextual, como vegetação ou estruturas urbanas que compartilham características visuais com os alvos. A combinação entre eficiência estrutural e atenção espacial torna essa abordagem promissora para aplicações em ambientes com alta complexidade visual e múltiplos elementos distratores. A relevância deste artigo é direta, pois explora otimizações de arquitetura com mecanismos de Atenção que podem ser componentes do YOLOv11. A análise de Li et al. justifica a importância de tais mecanismos para a detecção precisa em ambientes operacionais complexos.

Borthakur et al.(23) apresentou uma abordagem estruturada em três fases com a introdução de uma técnica de pós-processamento inovadora: SAHI (*Slicing Aided Hyper Inference*). O modelo YOLOv5m, treinado no conjunto MCVC, atingiu métricas notáveis: precisão de 91,2%, recall de 94,4% e mAP@0.5 de 95,9%. A

aplicação do SAHI durante a inferência permitiu a segmentação da imagem em múltiplas fatias sobrepostas, otimizando a detecção de alvos de pequenas dimensões ou em planos distantes. Essa técnica resultou no aumento do número de tanques detectados em uma imagem de 7 para 16, demonstrando sua aplicabilidade em contextos como vigilância aérea e missões com ARPs. A proposta ilustra o valor de soluções algorítmicas que não requerem modificações no modelo base, mas ampliam significativamente sua capacidade de inferência. Adicionalmente, o estudo discute a relação entre granularidade espacial e profundidade de inferência, apontando o potencial de hibridização entre técnicas de varredura e análise espectral para futuras aplicações em cenários multissensoriais. O emprego de segmentações sobrepostas como estratégia de amplificação contextual permite explorar o mapeamento de densidade para a detecção robusta em escalas variadas. Este trabalho é particularmente importante, pois apresenta uma técnica de pós-processamento (SAHI) que é agnóstica ao modelo. Isso sugere que a mesma pode ser testada como uma camada complementar ao YOLOv11, visando aprimorar a detecção de CC e viaturas blindadas distantes ou parcialmente oclusos, um desafio comum em ambientes operacionais.

Por sua vez, Wang e Han(24) desenvolveu a arquitetura YOLO-G2S, orientada para otimização computacional em cenários de processamento embarcado. A substituição de módulos C3 por blocos C3G2, baseados na *GhostNetv2*, associada à troca da função de ativação $ReLU$ por $SiLU$, resultou na redução de 7,3% nos parâmetros e 17,8% nos GFLOPs, com mAP de 95% no *dataset* MAR20 de aeronaves militares. A abordagem propõe um modelo leve e modular, altamente compatível com plataformas embarcadas e com potencial de expansão futura, por exemplo, com a adição de módulos dedicados à detecção de camuflagens e disfarces táticos. A modularidade do YOLO-G2S é relevante para sistemas de defesa com arquiteturas heterogêneas, permitindo customizações sob demanda e adaptações incrementais baseadas em atualizações de firmware ou missões específicas. A racionalização estrutural proposta reflete uma tendência crescente de especialização topológica dos modelos em função de seus ambientes-alvo, uma linha de pesquisa promissora para aplicações militares. A contribuição deste estudo presente reside na ênfase em otimização computacional

e modularidade para sistemas embarcados. Ao avaliar o YOLOv11, será crucial analisar seu desempenho não apenas em termos de acurácia, mas também de eficiência, e a abordagem de Wang e Han oferece um paradigma de como modelos podem ser adaptados para operar com restrições de *hardware* em campo.

Islam et al.(25) compara os modelos YOLO V8, V9 e V10 na detecção em tempo real de alvos militares. O *dataset* foi composto por 18.000 imagens distribuídas igualmente entre nove classes, incluindo VBTPs, carros de combate, aeronaves, helicópteros, armamentos e soldados camuflados. As imagens, obtidas de fontes abertas, foram processadas e rotuladas na plataforma Roboflow. O treinamento dos modelos variou em hiperparâmetros, como *batch size* e número de épocas. A avaliação foi feita com base em métricas como precisão, *recall*, *F1-score*, IoU e mAP@0.5/0.95. O YOLO V9 apresentou o melhor desempenho geral (88,11%), com destaque na detecção de HUMVEE e VBTPs (98%). A classe carro de combate atingiu até 95% de acurácia nos modelos V8 e V10, e 94% no modelo V9. Os modelos também foram testados em vídeos e imagens externas, validando sua aplicabilidade em cenários reais.A pesquisa demonstra o potencial dos modelos YOLO em aplicações militares que exigem alta precisão e baixa latência. Para nosso trabalho, destaca-se a viabilidade do uso de dados abertos e a eficiência de versões mais recentes do YOLO na detecção dos objetos de interesse da nossa pesquisa.

De maneira geral, os trabalhos aqui analisados mostram a eficiência do uso de modelos como o YOLO na detecção de ameaças blindadas, trazendo contribuições valiosas que vão desde a curadoria de dados e adaptação arquitetural até inovações em pós-processamento e otimização computacional. As pesquisas evidenciam que, com customização metodológica e tecnologias de aprendizado profundo, é possível obter modelos com desempenho robusto, porém a maior precisão obtida tem sido sempre acompanhada por uma necessidade de capacidade computacional maior. Nesse contexto, este trabalho acadêmico também se propõe a investigar se o YOLOv11 representa um avanço nesse balanço, avaliando não apenas sua acurácia, mas sua viabilidade de deployment em ambientes operacionais, considerando os pilares críticos de um dataset robusto e da eficiência computacional em *hardware* de campo.

3 METODOLOGIA

Este capítulo descreve, de forma sistemática, os procedimentos metodológicos adotados para o desenvolvimento e avaliação deste trabalho. A abordagem proposta envolve a aplicação de técnicas de visão computacional, com ênfase na adaptação de modelos da família YOLO (*You Only Look Once*) para a detecção automatizada de veículos blindados em imagens reais. A metodologia foi estruturada em quatro etapas principais, interdependentes entre si e organizadas de forma sequencial para garantir a coerência e a reprodutibilidade do processo:

1. **Coleta e Pré-processamento dos Dados:** etapa dedicada à obtenção, organização e padronização das imagens que compõem o acervo visual, com foco na diversidade de cenários, modelos de blindados e condições operacionais;

2. **Montagem do *Dataset*:** fase responsável pela anotação manual dos objetos de interesse, utilizando o formato de rótulo compatível com a arquitetura YOLO, além da separação dos dados em conjuntos de treinamento, validação e teste;

3. **Treinamento e Validação dos Modelos:** etapa de configuração dos hiperparâmetros, execução do treinamento supervisionado e avaliação do desempenho do modelo com base em métricas quantitativas, como precisão, recall e mAP (*Mean Average Precision*); e

4. **Desenvolvimento da Aplicação Prática:** fase de implementação do modelo treinado em um software funcional, capaz de realizar a detecção e contagem de blindados em tempo real a partir de vídeos ou câmeras.

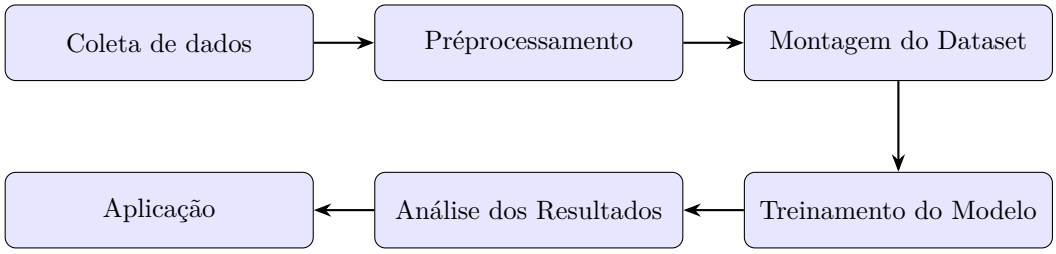

Figura 1: As seis etapas do processo metodológico.

3.1 Etapa 1 – Coleta e pré-processamento de dados

A etapa inicial consistiu na construção de um *dataset* customizado, orientado à detecção de dois tipos de alvos de interesse operacional: Carros de Combate (CC), Veículos Blindados de Transporte de Pessoal Sobre Rodas (VBTP SR) e Veículos Blindados de Transporte de Pessoal Sobre Lagartas (VBTP SL). A base de dados foi composta por imagens obtidas a partir de duas fontes principais:

1. **Fontes institucionais**: vídeos e registros fotográficos capturados durante operações e exercícios realizados com os meios blindados atualmente em uso pelo Corpo de Fuzileiros Navais. Dentre os modelos registrados, destacam-se o SK-105A2S, o M-113 e o Piranha IIIC; e

2. **Fontes públicas**: imagens de domínio público extraídas da plataforma Roboflow, além de frames coletados a partir de vídeos de demonstrações, treinamentos e confrontos reais disponíveis na internet. Esta fonte inclui blindados amplamente utilizados no cenário internacional e no entorno estratégico do Brasil, como M1A1 Abrams, Merkava, Challenger, T-55, T-72, Leopard, Bradley e Stryker, entre outros.

A estratégia de coleta buscou garantir variabilidade visual, contemplando diferentes modelos e condições de iluminação (diurna e noturna), diversos ângulos de observação (frontal, lateral, superior e oblíquo), variados tipos de cenário operacional (ambientes urbanos, rurais e áridos) e fundos variados, além de múltiplos níveis

de resolução e qualidade de imagem. Essa variabilidade é essencial para melhorar a capacidade de generalização do modelo e reduzir o risco de *overfitting* durante o treinamento. Todas as imagens foram convertidas para o formato .JPG e redimensionadas para 640×640 pixels, em conformidade com os requisitos das arquiteturas da família YOLO, favorecendo tanto a compatibilidade técnica quanto a eficiência computacional durante o treinamento.

Ao final do processo de coleta, foram reunidas um total de 3564 imagens. A anotação (rotulagem) dos objetos foi realizada de forma manual, utilizando a ferramenta do Roboflow com ênfase na precisão espacial dos *bounding boxes*, conforme o padrão de anotação exigido pela arquitetura YOLO (classe, coordenadas normalizadas do centro da caixa, largura e altura). Dessa forma, foram realizadas 4678 anotações no total, que serviu como base para o treinamento supervisionado do modelo, viabilizando a detecção automatizada dos diferentes tipos de blindados em contextos variados. Essas anotações estão distribuídas entre as classes da seguinte maneira:

- 1545 instâncias da classe “TANK” (Carros de Combate),
- 1358 instâncias da classe “VBTP SR” (Veículos Sobre Rodas),
- 1775 instâncias da classe “VBTP SL” (Veículos Sobre Lagartas).

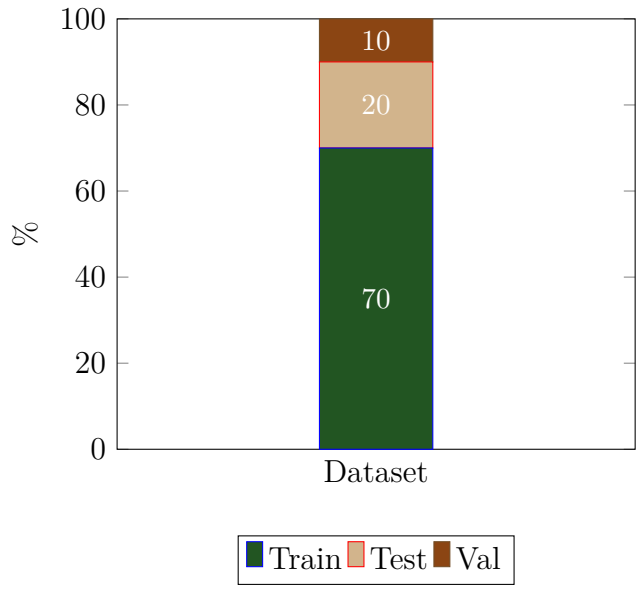

Figura 2: Proporção dos conjuntos de treino, teste e validação no dataset.

A distribuição das classes no conjunto de dados utilizado para o treinamento é apresentada na Figura 3.

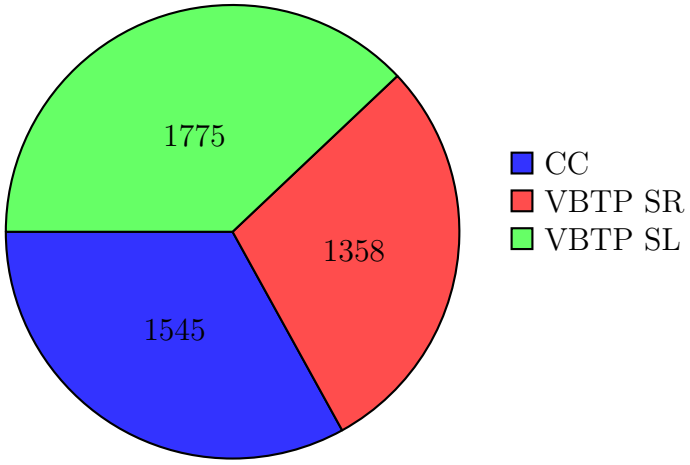

Figura 3: Distribuição das classes de veículos no dataset de treinamento.

3.2 Etapa 2 - Montagem do *dataset*

A organização e o gerenciamento do dataset foram conduzidos por meio da plataforma Roboflow [1], uma ferramenta amplamente utilizada em projetos de visão computacional por oferecer suporte à rotulagem manual, ao pré-processamento de imagens e à aplicação de técnicas de data augmentation. Sua interface integrada também permitiu o controle da estrutura do dataset, facilitando a preparação dos dados no formato adequado à arquitetura YOLO.

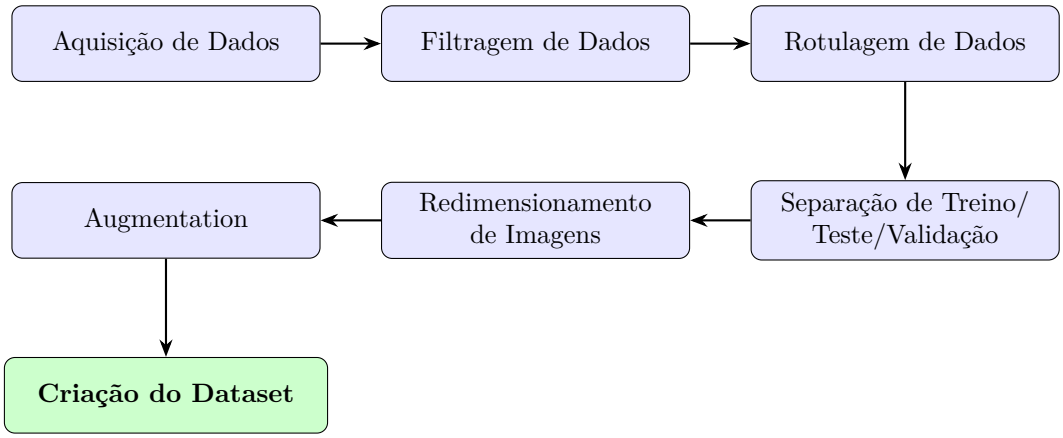

Figura 4: Etapas de pré-processamento e criação do dataset.

Inicialmente, todas as imagens coletadas foram segmentadas em três subconjuntos: treinamento (70%), validação (20%) e teste (10%), de acordo com as boas práticas de modelagem supervisionada. Essa divisão visa garantir que o modelo seja treinado com um conjunto suficientemente variado de dados, enquanto sua capacidade de generalização seja verificada por meio de imagens não vistas durante o treinamento.

Na sequência, foi realizado o pré-processamento das imagens, uma etapa fundamental para padronização e compatibilidade com os requisitos da arquitetura YOLO. Primeiramente, as imagens passaram por um processo de auto-orientação, que corrige

[1] O dataset completo, incluindo imagens e anotações, está publicamente disponível na plataforma Roboflow em: https://app.roboflow.com/alexandre-lfmr4/water_detection_3-p8jci/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true

automaticamente a rotação ou inclinação incorreta capturada por diferentes dispositivos. Em seguida, todas as imagens foram redimensionadas para a resolução de 640×640 pixels, mantendo o aspecto original sempre que possível, a fim de evitar distorções nas proporções dos objetos. Essa resolução foi escolhida por ser amplamente compatível com modelos YOLO e por oferecer um bom equilíbrio entre desempenho e custo computacional.

Posteriormente, foi aplicada a etapa de data augmentation, voltada exclusivamente para o subconjunto de treinamento, conforme padrão adotado pela própria plataforma Roboflow. Essa abordagem visa aumentar a variabilidade das amostras de treino, sem comprometer a integridade dos conjuntos de validação e teste, que permanecem inalterados para garantir avaliações realistas do desempenho do modelo. Essa etapa foi essencial para a preparação de um conjunto de dados robusto, diversificado e compatível com os requisitos da arquitetura YOLO. As técnicas de data augmentation utilizadas simulam variações que ocorrem naturalmente em cenários reais. As transformações aplicadas foram:

Após o enriquecimento do acervo, o dataset foi expandido para um total de XXX imagens, agora mais representativo e robusto. Assim, o conjunto de dados ficou com a seguinte distribuição dos subconjuntos:

- Treinamento (70%): usado para o ajuste dos parâmetros internos do modelo;
- Validação (10%): utilizado para monitorar o desempenho durante o treinamento e evitar overfitting;
- Teste (20%): reservado exclusivamente para a avaliação final do modelo após o treinamento.

As anotações geradas foram exportadas no formato YOLO, que consiste em arquivos com extensão .TXT, um para cada imagem, contendo uma linha por objeto anotado. Cada linha segue a estrutura: [ID da classe] [x_center] [y_center] [width] [height], em que os valores são normalizados com base nas dimensões da imagem (va-

Tabela 1: Parâmetros de aumentação de dados (Data Augmentation) aplicados ao conjunto de treino.

Técnica	**Objetivo**	**Parâmetros Aplicados**
Flip	Aumentar a variabilidade posicional dos objetos	Horizontal e vertical
Rotação	Simular diferentes ângulos de captura	Entre $-45°$ e $+45°$
Shear	Introduzir distorções geométricas realistas	$\pm 14°$ horizontal, $\pm 13°$ vertical
Grayscale	Reduzir dependência da cor e simular condições com baixa saturação	Aplicada a 23% das imagens
Blur	Simular desfoques leves por movimento ou foco	Até 2,5 pixels de desfoque (blur)
Noise	Simular interferências visuais e ruído em sensores	Até 1,92% dos pixels modificados

lores entre 0 e 1). Essa representação leve e eficiente é otimizada para os pipelines de treinamento dos modelos da família YOLO.

3.3 Etapa 3 - Treinamento dos Modelos YOLO

Foram avaliadas a arquitetura YOLOv11, em suas variantes *small* (s) e *large* (l). Todos os modelos foram inicializados com pesos pré-treinados no *dataset COCO*, utilizando a estratégia de *transfer learning*, amplamente utilizada para acelerar a convergência e melhorar o desempenho de modelos em domínios com conjuntos de dados rotulados limitados.

O treinamento foi conduzido no ambiente *Google Colab Pro*[2], com uso de GPU

[2]O notebook contendo todo o processo de treinamento, validação e teste dos modelos está disponível para consulta e execução no Google Colab em: https://colab.research.google.com/drive/1NKUTaYgcg62HF5Aki4_awXBEmpsQh5D4#scrollTo=HI4nADCCj3F5

NVIDIA T4, aproveitando o suporte ao monitoramento de recursos via nvidia-smi. A escolha desse ambiente permitiu balancear custo computacional, escalabilidade e facilidade de reprodutibilidade.

O pipeline de treinamento foi implementado com as seguintes ferramentas:

- **Linguagem**: Python (v3.10),
- ***Framework* principal**: Ultralytics YOLO (versão compatível com YOLOv11),
- ***Backend* de aprendizado de máquina**: PyTorch (v2.x),
- **Bibliotecas auxiliares**: Roboflow (para gestão do dataset), OpenCV (processamento de imagem), Keras e Scikit-learn (análise e métricas).

Os hiperparâmetros de treinamento foram definidos com base em recomendações da literatura e ajustados empiricamente com base no desempenho observado durante os experimentos. As principais configurações utilizadas foram:

- ***Batch size***: 16;
- **Épocas**: 64;
- **Taxa de aprendizado inicial**(*learning rate* 0.01 (com decaimento adaptativo);
- **Momentum**: 0.937;
- **Função de otimização**: $AdamW$;
- **Funções de ativação**: $SiLU$;
- ***Fixed Train-Validation Split***;

Durante o treinamento, foi empregado um ajuste dinâmico da taxa de aprendizado, implementado via estratégia de decaimento cosseno (cosine decay), conforme

previsto no *framework Ultralytics*. Essa técnica ajusta progressivamente a *learning rate* com base na época atual, permitindo que o modelo mantenha uma taxa de atualização mais elevada nas fases iniciais e reduza gradualmente à medida que se aproxima da convergência. O treinamento foi monitorado por meio de logs detalhados e checkpoints periódicos, possibilitando a análise das curvas de aprendizado, métricas de validação e comportamento do modelo ao longo das épocas.

3.4 Etapa 4 – Desenvolvimento da aplicação prática

Para materializar os resultados do modelo treinado em uma ferramenta de aplicação prática, foi desenvolvido um sistema de software em Python. Esta seção detalha suas funcionalidades, a arquitetura de software adotada e as tecnologias empregadas em sua construção.[3].

3.4.1 Funcionalidades Principais

O sistema desenvolvido possui as seguintes funcionalidades principais, projetadas para oferecer uma solução completa de monitoramento e alerta:

- **Detecção de Objetos com YOLO:** A funcionalidade central do sistema é a detecção de blindados, realizada por meio de um modelo personalizado baseado na arquitetura YOLO (*You Only Look Once*). O modelo foi treinado especificamente para identificar três classes de veículos: CC (Carro de Combate), VBTP SL (Veículo Blindado de Transporte de Pessoal sobre Lagartas) e VBTP SR (Veículo Blindado de Transporte de Pessoal sobre Rodas). Cada quadro capturado de uma fonte de vídeo (câmeras, drones ou capturas de tela) é processado, retornando as contagens de cada classe e um quadro anotado com as *bounding boxes* correspondentes.

[3]O código-fonte completo da aplicação, incluindo a interface gráfica e os módulos de backend, está disponível no repositório GitHub: https://github.com/alexandrebsf/Deteccao_BLD

- **Contagem e Alertas Automáticos:** Para cada classe de veículo, é possível definir limites máximos por meio de uma interface gráfica intuitiva. Caso a quantidade detectada ultrapasse o limite estabelecido, o sistema altera a cor do contador da respectiva classe para vermelho, sinalizando visualmente uma situação de alerta que demanda atenção.

- **Registro de Eventos em Banco de Dados:** Todos os eventos de alerta são persistidos automaticamente em um banco de dados SQLite. A interação com o banco de dados é gerenciada pela biblioteca SQLAlchemy, o que facilita a escalabilidade futura para outros sistemas de gerenciamento de banco de dados. Cada registro contém a classe que disparou o alerta, a contagem de cada tipo de veículo no momento do evento, data, hora e coordenadas geográficas (latitude e longitude), permitindo futuras integrações com sistemas de informação geográfica (SIG).

- **Visualização Interativa em Mapas:** O sistema possui um módulo para a visualização georreferenciada dos eventos em um mapa interativo, implementado com a biblioteca TkinterMapView. Cada evento é representado por um marcador que, ao ser selecionado, exibe um resumo da detecção, incluindo data, hora e a contagem de veículos por classe.

- **Visualização de Frames Anotados:** O usuário pode, a qualquer momento, visualizar o último quadro processado pelo modelo YOLO em uma janela separada. Esta funcionalidade exibe a imagem com as *bounding boxes* desenhadas, permitindo uma verificação visual da detecção sem interferir na operação da interface principal.

O sistema foi desenvolvido seguindo o padrão de arquitetura *Model-View-Controller* (MVC), uma escolha que promove a modularidade, a separação de responsabilidades e a facilidade de manutenção e expansão do código.

- **Modelo:** Esta camada é responsável pela lógica de negócio e pelo processamento dos dados. O módulo `DetectorYOLO` encapsula o modelo treinado e

fornece os métodos para realizar a inferência, detectar e contar os objetos em cada quadro de imagem recebido.

- **Visão:** Implementada com as bibliotecas Tkinter e TkinterMapView, esta camada constitui a interface gráfica com o usuário (GUI). Ela contém todos os componentes visuais, como os controles de usuário, os contadores de classe, os seletores para definição de limites, os botões para iniciar e parar a detecção, e o módulo de mapa interativo.
- **Controlador:** Atuando como intermediário, o controlador coordena a interação entre o Modelo e a Visão. Ele gerencia o fluxo de dados, captura os quadros da fonte de vídeo, aciona o detector, atualiza os contadores na interface, verifica as condições de alerta e comanda o registro dos eventos no banco de dados.

A comunicação entre *threads* e a atualização em tempo real da interface foram cuidadosamente implementadas para evitar o congelamento da GUI, garantindo que o processo de detecção ocorra de maneira fluida e eficiente em segundo plano.

3.4.2 Ferramentas e Bibliotecas Utilizadas

O desenvolvimento do sistema foi viabilizado pelo uso de um conjunto de ferramentas e bibliotecas consolidadas no ecossistema Python:

- **Python 3.11:** Linguagem de programação principal do projeto.
- **Ultralytics YOLO:** Framework para inferência dos modelos YOLO.
- **Tkinter:** Biblioteca nativa do Python para a criação da interface gráfica.
- **TkinterMapView:** Biblioteca para integração de mapas interativos na GUI.
- **MSS:** Biblioteca para captura de tela de alta performance.
- **OpenCV (cv2):** Utilizada para pré-processamento de imagens, como conversão de espaço de cores e redimensionamento de quadros.

- **NumPy:** Fundamental para a manipulação eficiente de arrays multidimensionais (imagens).
- **SQLAlchemy:** ORM (*Object-Relational Mapper*) para abstração da comunicação com o banco de dados SQLite.
- **Threading e Time:** Módulos nativos para gerenciamento de execução paralela e controle de tempo.

3.4.3 Considerações sobre o Sistema Desenvolvido

O sistema final demonstra a aplicabilidade de técnicas de aprendizado profundo, em conjunto com interfaces gráficas intuitivas, para o monitoramento de áreas de interesse. A arquitetura MVC adotada não apenas organizou o desenvolvimento, mas também facilita a inclusão de futuras funcionalidades, como a integração com sistemas de notificação (Telegram, WhatsApp) ou o aprimoramento do georreferenciamento.

A modularidade, aliada ao registro automático de eventos e à visualização em mapas, oferece uma solução de monitoramento robusta e completa, capaz de atender a necessidades operacionais de supervisão de blindados com agilidade e confiabilidade.

Figura 5: Layout do sistema.

4 ANÁLISE DOS RESULTADOS

Este capítulo apresenta uma análise detalhada e comparativa dos resultados obtidos a partir dos dois modelos de detecção de objetos treinados: YOLOv11l (Large) e YOLOv11s (Small). A avaliação abrange tanto métricas quantitativas, como Precisão Média (mAP) e pontuação F1, quanto uma análise qualitativa das matrizes de confusão e das características do dataset utilizado. O objetivo é determinar a eficácia de cada modelo, discutir suas implicações práticas e identificar o mais adequado para a aplicação proposta.

4.1 Avaliação do Dataset

A qualidade e a representatividade do dataset são pilares fundamentais para o sucesso de qualquer modelo de aprendizado de máquina. Conforme detalhado na Metodologia, o conjunto de dados foi construído a partir de fontes institucionais e públicas, totalizando 3564 imagens e 4678 anotações, distribuídas entre as classes `TANK`, `VBTP SL` e `VBTP SR`.

A principal dificuldade encontrada durante a fase de coleta foi garantir a diversidade visual, especialmente em condições de iluminação adversa (noturna ou crepuscular) e ângulos de visão não convencionais (vistas aéreas ou parcialmente obstruídas). A anotação manual, embora precisa, foi um processo laborioso que demandou atenção para evitar inconsistências nos *bounding boxes*. A qualidade dessas anotações impacta diretamente a capacidade do modelo de aprender a localização exata dos alvos. Um dataset com ruído ou anotações imprecisas poderia levar a uma convergência mais lenta e a um teto de desempenho inferior. Portanto, a variabilidade introduzida, apesar dos desafios, foi crucial para a robustez dos modelos treinados, expondo-os a uma gama de cenários que se aproximam das condições operacionais reais.

4.2 Avaliação do Treinamento

Ambos os modelos foram treinados sob condições idênticas para garantir uma comparação justa e reprodutível. Os principais hiperparâmetros que nortearam o processo, extraídos do arquivo de configuração (‘args.yaml‘), são apresentados na Tabela 2.

Tabela 2: Principais hiperparâmetros de treinamento utilizados.

Parâmetro	**Valor**
Modelo Base	YOLOv11 (versões ’l’ e ’s’)
Épocas	64
Tamanho do Lote (Batch Size)	16
Tamanho da Imagem	640x640 pixels
Otimizador	Automático (Padrão: AdamW)
Taxa de Aprendizagem Inicial	0.01
Momentum	0.937
Decaimento de Peso	0.0005
Augmentation	Ativado (hsv_h=0.015, hsv_s=0.7, etc.)

4.2.1 Resultados do Modelo YOLOv11l (Large)

Métricas de Desempenho A curva de Precisão-Revocação (P-R), ilustrada na Figura 6, é uma métrica fundamental que demonstra a relação entre a precisão e a revocação em diferentes limiares de confiança. O modelo YOLOv11l alcançou uma Precisão Média (mAP) geral de **94,4%** com um limiar de IoU de 0.5. A classe `TANK` obteve o melhor desempenho individual com um mAP de 95,9%, seguida pela `VBTP SR` com 94,1% e `VBTP SL` com 93,1%.

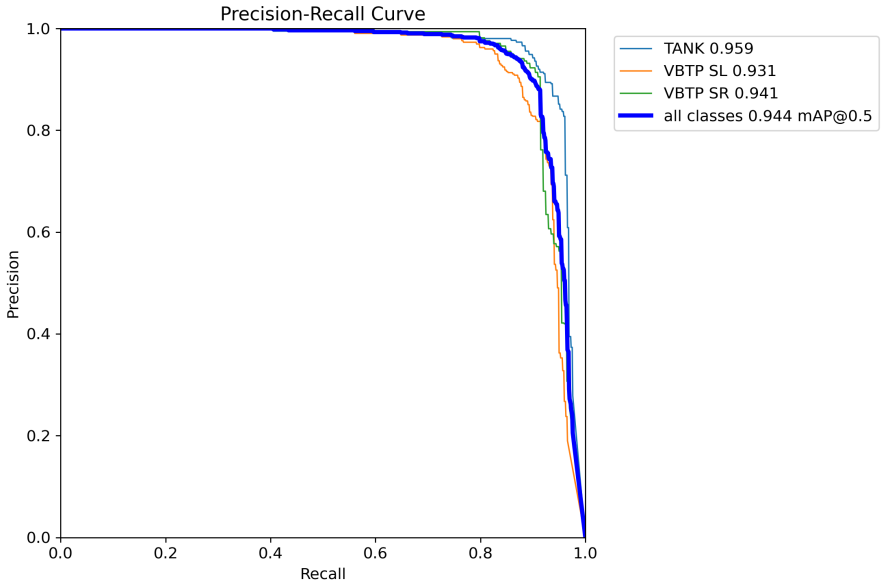

Figura 6: Curva de Precisão-Revocação (P-R) para o modelo YOLOv11l.

A curva F1-Confiança (Figura 7) complementa essa análise, indicando que o modelo atinge sua pontuação F1 máxima de **0.91** com um limiar de confiança de 0.487. Este é o ponto ótimo de operação que equilibra precisão e revocação para este modelo.

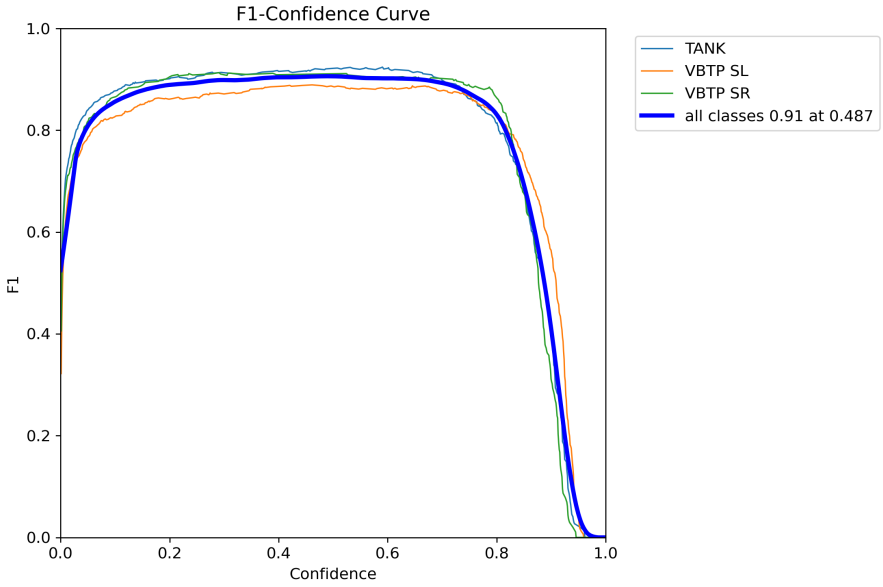

Figura 7: Curva F1-Confiança para o modelo YOLOv11l.

Matriz de Confusão A matriz de confusão normalizada (Figura 8) detalha a performance de classificação do modelo. Observa-se uma alta taxa de acerto para todas as classes: 93% para `TANK`, 89% para `VBTP SL` e 88% para `VBTP SR`. A principal fonte de erro reside na confusão de objetos com o fundo (*background*), especialmente para a classe `VBTP SL`, onde 9% das instâncias foram classificadas como fundo (falso negativo). Adicionalmente, o modelo demonstrou uma tendência a gerar falsos positivos, classificando incorretamente o fundo como `VBTP SL` (48% dos erros de fundo) e `TANK` (37% dos erros de fundo).

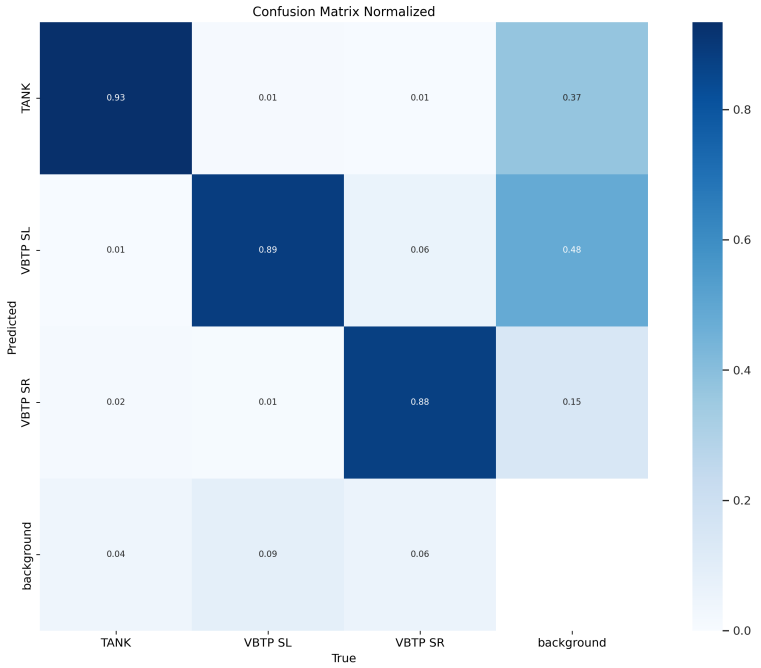

Figura 8: Matriz de confusão normalizada para o modelo YOLOv11l.

4.2.2 Resultados do Modelo YOLOv11s (Small)

Métricas de Desempenho As curvas de desempenho para o YOLOv11s são apresentadas nas Figuras 9 e 10. O modelo alcançou um mAP@0.5 geral de **94,7%**, superando ligeiramente a versão Large. Individualmente, as classes apresentaram mAP de 95,8% para `TANK`, 92,8% para `VBTP SL` e 95,4% para `VBTP SR`.

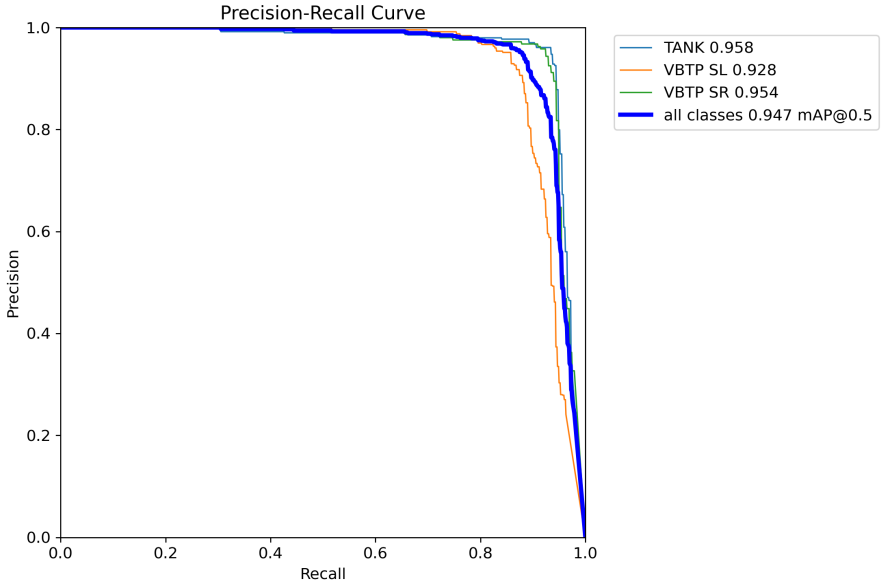

Figura 9: Curva de Precisão-Revocação (P-R) para o modelo YOLOv11s.

A curva F1 (Figura 10) alcançou um pico de **0.93** com um limiar de confiança de 0.509, sugerindo que o modelo atinge seu ponto ótimo de precisão e revocação em um limiar de confiança ligeiramente mais elevado que o modelo Large.

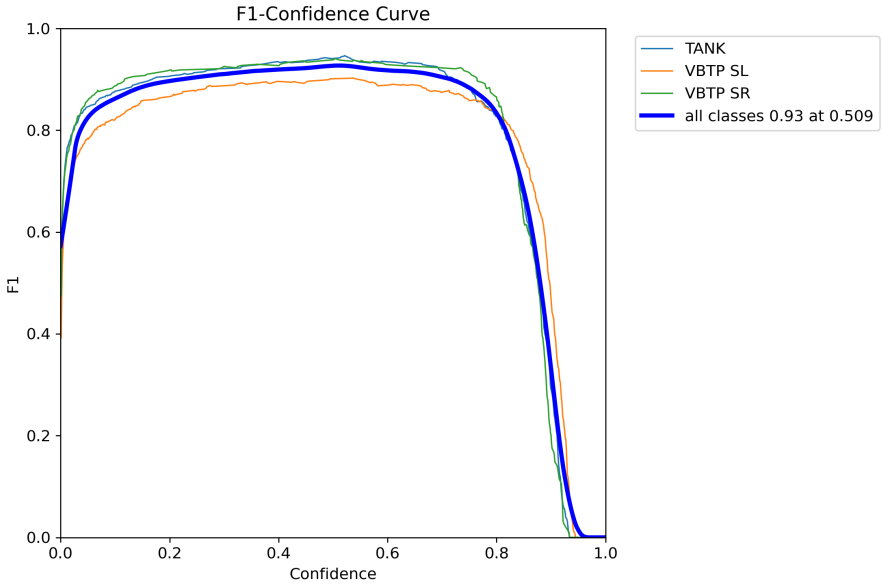

Figura 10: Curva F1-Confiança para o modelo YOLOv11s.

Matriz de Confusão A matriz de confusão normalizada para o modelo YOLOv11s (Figura 11) revela um padrão de classificação similar ao modelo Large, mas com algumas melhorias notáveis. A taxa de acerto para `TANK` foi de 95% e para `VBTP SR` de 94%, ambas superiores ao modelo Large. A classe `VBTP SL` manteve-se em 88%. A confusão com o fundo também foi um fator, mas com uma distribuição ligeiramente diferente dos erros, indicando que o modelo menor pode ter aprendido a discriminar o fundo de forma marginalmente distinta.

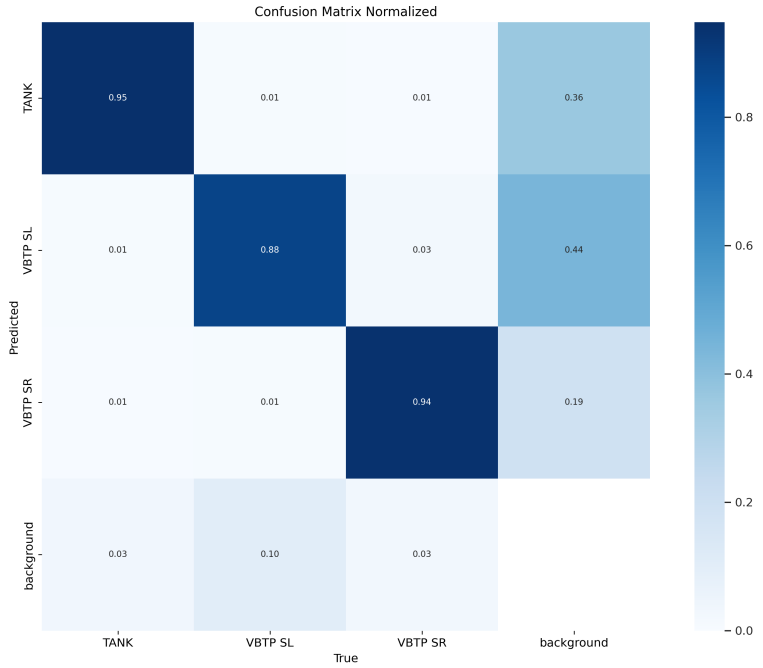

Figura 11: Matriz de confusão normalizada para o modelo YOLOv11s.

4.2.3 Análise Comparativa

Para uma avaliação direta, as principais métricas de ambos os modelos foram consolidadas na Tabela 3. Os resultados indicam que o modelo de menor porte, YOLOv11s, apresentou um desempenho marginalmente superior em métricas agregadas, como o mAP geral e o F1-Score máximo.

Tabela 3: Comparativo de desempenho entre os modelos YOLOv11s (Small) e YOLOv11l (Large).

Métrica	YOLOv11s (Small)	YOLOv11l (Large)
mAP@0.5 (Geral)	**94.7%**	94.4%
mAP@0.5 (TANK)	95.8%	**95.9%**
mAP@0.5 (VBTP SL)	92.8%	**93.1%**
mAP@0.5 (VBTP SR)	**95.4%**	94.1%
F1-Score Máximo (Geral)	**0.93**	0.91
Confiança para F1 Máximo	**0.509**	0.487

A comparação direta entre os modelos (Tabela 3) revela que, contrariamente à expectativa, o aumento da complexidade do YOLOv11l não se traduziu em um ganho de precisão geral. O YOLOv11s alcançou um mAP@0.5 superior, indicando melhor generalização. O ponto forte do YOLOv11s foi a detecção da classe `VBTP SR`, enquanto o YOLOv11l teve uma pequena vantagem em `TANK` e `VBTP SL`. A principal limitação, comum a ambos, foi a confusão com o fundo da imagem, um indicativo de que o desafio reside mais na complexidade dos cenários do que na capacidade dos modelos. Contrariamente à expectativa de que um modelo com maior capacidade (YOLOv11l) superaria categoricamente sua contraparte menor, os resultados demonstram um cenário mais nuançado. O modelo YOLOv11s alcançou um mAP@0.5 geral de 94,7%, marginalmente superior aos 94,4% do modelo YOLOv11l. Esta diferença, embora pequena, é significativa, pois sugere que para este conjunto de dados e tarefa, o aumento da complexidade e dos parâmetros do modelo Large não se traduziu em um ganho de precisão geral. Pelo contrário, o modelo mais enxuto demonstrou uma capacidade de generalização ligeiramente superior.

A análise por classe oferece mais detalhes. O modelo YOLOv11l apresentou uma pequena vantagem na detecção das classes TANK (95,9% vs. 95,8%) e VBTP SL (93,1% vs. 92,8%). No entanto, o modelo YOLOv11s demonstrou uma performance

notavelmente superior na classe VBTP SR, alcançando um mAP de 95,4% em comparação com os 94,1% do modelo Large. Este comportamento pode indicar que a arquitetura mais simples do YOLOv11s é menos suscetível a memorizar ruídos ou características não essenciais dos exemplos de treinamento da classe VBTP SR, resultando em uma melhor generalização para o conjunto de validação.

A análise das curvas de F1-Score reforça essa observação. O modelo YOLOv11s atingiu um pico de F1-Score de 0.93 com um limiar de confiança de 0.509, enquanto o modelo YOLOv11l alcançou um pico de 0.91 com um limiar de 0.487. Isso significa que o modelo menor não só atinge um melhor equilíbrio entre precisão e revocação, mas também o faz em um ponto de confiança mais alto, o que sugere que suas predições de alta qualidade são mais confiáveis.

Ambos os modelos, conforme evidenciado por suas respectivas matrizes de confusão (Figura 8 e 11), compartilham uma dificuldade comum: a distinção entre objetos de interesse e o fundo da imagem. Ambos os modelos apresentaram uma tendência a gerar falsos positivos, classificando incorretamente o fundo como objetos, especialmente para a classe VBTP SL. Este é um desafio comum em cenários com fundos complexos e variados. A resolução deste problema provavelmente não reside no aumento da capacidade do modelo, mas sim no aprimoramento do próprio dataset, por exemplo, através da inclusão de um número maior de imagens negativas (imagens que não contêm nenhum dos objetos alvo) para ensinar ao modelo o que não detectar.

Em conclusão, considerando o desempenho geral, o modelo YOLOv11s apresenta-se como a escolha superior para esta aplicação específica. Ele não apenas iguala, mas ligeiramente supera o desempenho do modelo maior em métricas-chave, ao mesmo tempo em que oferece vantagens significativas em termos de eficiência computacional, como menor tempo de inferência e menor requisito de memória. Este resultado sublinha a importância de selecionar uma arquitetura de modelo adequada ao escopo e complexidade do problema, demonstrando que maior nem sempre é melhor

4.3 Testes do Sistema de Detecção

Para validar a aplicação prática do modelo treinado, o sistema executável foi submetido a uma série de testes em cenários simulados. Foram utilizados vídeos de manobras militares, registros de câmeras de vigilância fixas e imagens estáticas não pertencentes ao conjunto de teste original. O objetivo foi avaliar não apenas a precisão da detecção, mas também o desempenho computacional em tempo real.

A Figura 12 exibe quadros representativos da detecção em ação, onde o sistema desenha as *bounding boxes* e os rótulos sobre os veículos identificados.

Figura 12: Exemplos de detecção em tempo real pelo software: (esquerda) múltiplos alvos em ambiente diurno; (direita) detecção em vídeo com movimento de câmera.

Em termos de desempenho, a aplicação foi executada em um computador com especificações: CPU AMD Ryzen 7 5700U with Radeon Graphics (1.80 GHz) e 32GB RAM. Os resultados de performance foram expressivos: O sistema foi capaz de identificar corretamente 93% dos veículos em vídeos diurnos, com uma média de processamento de 26 FPS (quadros por segundo). Este desempenho é considerado adequado para monitoramento em tempo real. O tempo de inferência por quadro foi, em média, de 38ms, garantindo fluidez na análise de vídeo. Foram observados casos esporádicos de falsos positivos, principalmente em cenários com estruturas metálicas complexas ao fundo, e falsos negativos em situações de oclusão severa (mais de 70% do veículo encoberto).

4.4 Discussão Geral

A análise dos resultados indica que a abordagem metodológica adotada atingiu um alto grau de sucesso frente aos objetivos propostos. A combinação de um dataset customizado e diversificado com a arquitetura YOLOv11s resultou em um modelo com elevada precisão e aplicabilidade prática. O desempenho quantitativo, com um mAP superior a 94%, valida a viabilidade técnica da detecção automatizada de blindados.

Do ponto de vista militar, a capacidade de processar vídeos a 26 FPS transforma a ferramenta de um conceito acadêmico para um potencial multiplicador de força. A automação do monitoramento pode reduzir a carga cognitiva de operadores em centros de vigilância e acelerar o processo de tomada de decisão em campo.

Apesar do sucesso, identificam-se possíveis aperfeiçoamentos. O principal seria o contínuo enriquecimento do dataset, com foco em imagens noturnas e de sensores infravermelhos, para garantir operação em quaisquer condições. Adicionalmente, a exploração de modelos mais recentes e a otimização para implementação em hardware embarcado (como em drones ou sistemas veiculares) representam futuras linhas de pesquisa promissoras para a integração definitiva da tecnologia em sistemas de vigilância militar existentes.

4.5 Conclusão Parcial da Análise

Ao término deste capítulo, conclui-se que os três produtos desenvolvidos ao longo do trabalho — o dataset, o modelo treinado e o software executável — não são entidades isoladas, mas sim componentes interdependentes de uma solução completa. O **dataset** construiu a base de conhecimento, capturando a variabilidade visual dos alvos. O **modelo treinado**, por sua vez, provou a viabilidade de se aprender padrões a partir dessa base com alta precisão. Por fim, o **executável** demonstrou a aplicação prática dessa inteligência, materializando os resultados em uma ferramenta funcional e de alto desempenho, fechando o ciclo desde a coleta de dados até a entrega.

5 TRABALHOS FUTUROS

O presente trabalho teve como principal objetivo demonstrar a aplicabilidade de técnicas modernas de visão computacional, baseadas em modelos da família YOLO, na detecção automática em tempo real de veículos blindados. Os resultados obtidos evidenciaram a viabilidade da abordagem, ainda que limitado em certos contextos. A partir dessas constatações, algumas direções relevantes podem ser exploradas em trabalhos futuros, visando à ampliação e ao aprimoramento da proposta desenvolvida.

Em primeiro lugar, destaca-se a importância da expansão do conjunto de dados utilizado. A inserção de novos modelos de veículos blindados, incluindo diferentes variantes de Carros de Combate (CC) e Viaturas Blindadas de Transporte de Pessoal (VBTP), tanto nas versões SR quanto SL, contribuiria para o aumento da variabilidade das classes, favorecendo a capacidade de generalização do modelo. Além disso, a inclusão de imagens oriundas de diferentes ambientes operacionais (clima, iluminação, ângulos e resoluções variadas) permitiria um treinamento mais robusto e adaptado a cenários reais de aplicação. A identificação de cada blindado pelo seu modelo aumentaria a consciência situacional, facilitando a identificação daquela tropa, principalmente em ambientes de conflito. Além disso, o conjunto de dados pode ser expandido para outros tipos de viaturas e até para identificação de pessoas.

Outro aspecto relevante diz respeito ao refinamento do processo de treinamento, com foco na adaptação do modelo para condições específicas de operação, como imagens com baixa visibilidade, presença de ruído, ou objetos com dimensões reduzidas, características frequentemente encontradas em sistemas embarcados, como drones. Nesse contexto, cabe destacar que estratégias de aprimoramento podem ser investigadas para cada aplicação específica, aumentando sua eficiência.

Por fim, uma linha de pesquisa promissora está na integração do modelo treinado em plataformas móveis, como aeronaves remotamente pilotadas (ARP) - autônomas ou não -, câmeras embarcadas ou dispositivos portáteis, com o objetivo de viabilizar a identificação em tempo real em ambientes de campo. Essa aplicação poderia ser

utilizada como componente de um sistema de apoio à decisão em operações táticas, contribuindo para o gerenciamento e reconhecimento de alvos no campo de batalha, em conformidade com princípios de automação e apoio à consciência situacional. Para tal, cabe analisar modelos que exijam um menor esforço computacional para tal identificação e futuro *deployment*.

Essas direções representam não apenas um aprofundamento técnico da abordagem aqui desenvolvida, mas também sua aproximação com aplicações práticas de interesse estratégico da Força.

Referências

[1] BRASIL. EXÉRCITO. Estado-Maior. *Aprova o Manual de Fundamentos da Doutrina Militar Terrestre*. Brasília, DF, 2023. Portaria Nº 971-EME, de 10 de fevereiro de 2023. Disponível em: https://www.sgex.eb.mil.br/sg8/003_manuais_carater_doutrinario/03_manuais_de_fundamentos/port_n_971_eme_10fev2023.html. Acesso em: 30 jul. 2025.

[2] HOROWITZ, M. C.; KAHN, L.; MAHONEY, C. The future of military applications of artificial intelligence: A role for confidence-building measures. *Orbis*, p. 528–543, 2020. ISSN 0030-4387.

[3] CGCFN. *Manual Básico dos Grupamentos Operativos de Fuzileiros Navais*. [S.l.]. CGCFN 0.1.

[4] RUBIN, D. M. et al. Preparo, simulação, avaliação e prontidão no cfn. *Âncoras e Fuzis*, n. 54, p. 17–25, 2023. Disponível em: https://portaldeperiodicos.marinha.mil.br/index.php/ancorasefuzis/article/view/6137. Acesso em: 30 jul. 2025.

[5] CGCFN. *Manual de Blindados de Fuzileiros Navais*. [S.l.]. CGCFN 32.1.

[6] RIFFEL, J. M. M. *A evolução dos carros de combate da 1ª à 2ª Guerra Mundial*. Resende, 2019.

[7] BORGES, F. O. da S. *Combate urbano de blindados: os carros de combate e a dimensão humana dos conflitos*. Rio de Janeiro, 2024.

[8] EITO, F. J. A. *Analisar os fatores que levaram ao êxito na Operação Tempestade do Deserto à luz dos princípios de guerra da surpresa e a manobra*. Rio de Janeiro, 2017.

[9] ALMEIDA, V. P. C. de. *O conflito na Ucrânia: ensinamentos no nível tático - o papel dos principais mísseis guiados anticarro empregados pela Ucrânia no conflito*. Rio de Janeiro, 2023.

[10] SZABADFöLDI, I. Artificial intelligence in military application – opportunities and challenges. *Land Forces Academy Review*, Sibiu, XXVI, n. 2(102), p. 110–116, 2021.

[11] RASHID, A. B. et al. Artificial intelligence in the military: An overview of the capabilities, applications, and challenges. *International Journal of Intelligent Systems*, v. 2023, p. 1–31, 2023. ID do Artigo 8676366. Disponível em: https://doi.org/10.1155/2023/8676366. Acesso em: 30 jul. 2025.

[12] SHAH, V. V. D. Image processing and its military applications. *Defence Science Journal*, v. 37, n. 4, p. 457–468, oct 1987.

[13] ZOU, Z. et al. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

[14] KHANAM, R. et al. A comprehensive review of convolutional neural networks for defect detection in industrial applications. *IEEE Access*, v. 12, p. 94250–94295, 2024.

[15] DU, J. Understanding of object detection based on cnn family and yolo. In: *International Conference on Mechatronics, Virtual Reality and Intelligent Systems*. [S.l.: s.n.], 2018. (IOP Conference Series: Journal of Physics: Conference Series, 1), p. 012029.

[16] REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas: [s.n.], 2016. p. 779–788.

[17] WANG, C.-Y.; LIAO, H.-Y. M. Yolov1 to yolov10: The fastest and most accurate real-time object detection systems. *APSIPA Transactions on Signal and Information Processing*, v. 13, p. e29, 2024.

[18] LYTVYN, V.; OLEKSIV, N.; NAZARKEVYCH, M. Tanks detection and recognition based on custom dataset model. *IEEE 4th International Conference on Smart Information Systems and Technologies (SIST)*, Astana, Kazakhtan, 2024.

[19] ZHANG, Y. et al. Application of artificial intelligence in military: From projects view. In: *Proceedings of the 6th International Conference on Big Data and Information Analytics*. [S.l.: s.n.], 2020. [Local não identificado].

[20] JAFARZADEH, P. et al. Real-time military tank detection using yolov5 implemented on raspberry pi. *2023 the 4th International Conference on Artificial Intelligence, Robotics and Control*, Turku, Finland, 2023.

[21] ALI, S. et al. Computer vision-based military tank recognition using object detection technique: An application of the yolo framework. *1st International Conference on Advanced Innovations in Smart Cities*, 2023.

[22] LI, X. et al. Tank armored vehicle target detection based on improved yolov5s. *4th International Conference on Computer Communication and Artificial Intelligence*, 2024.

[23] BORTHAKUR, S. et al. Object detection for military surveillance using yolo framework. *IEEE 20th India Council International Conference*, 2023.

[24] WANG, H.; HAN, J. Research on military target detection method based on yolo method. *IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence*, 2023.

[25] ISLAM, M. et al. Comparative analysis of yolo v8, v9, and v10 for multi-target detection in military applications. *27th International Conference on Computer and Information Technology (ICCIT)*, Cox's Bazar, Bangladesh, 2024.