SIAM - Sistema de Identificação de Aeronaves Militares

Bruno Brum[1], **Christian Vieira**[1], **Daniel Murta**[1], **Gabriel Luna**[1], **Lucas Zegrine**[1]

[1]Instituto de Ciências Exatas e Informática (ICEI)
Pontifícia Universidade Católica de Minas Gerais (PUCMINAS)
Belo Horizonte – MG – Brazil

{bruno.brum, christian.david, daniel.murta}@sga.pucminas.br
{gabriel.anjos.1280572, lucas.duarte.1327493}@sga.pucminas.br

Abstract. *This work explores approaches for recognizing and classifying military equipment using AI and machine learning. Traditional computer vision methods and advanced solutions such as neural networks are discussed. Challenges include dealing with different types of aircraft and environmental conditions. The integration of techniques such as Information Fusion and deep learning models such as YOLOv8 shows significant improvements in accuracy and speed. The proposed systems provide high accuracy and real-time identification, increasing the safety and effectiveness of military operations by eliminating human errors and optimizing data processing.*

Resumo. *Este trabalho explora abordagens para reconhecimento e classificação de equipamentos militares usando IA e aprendizado de máquina. Métodos tradicionais de visão computacional e soluções avançadas como redes neurais são discutidos. Os desafios incluem lidar com diferentes tipos de aeronaves e condições ambientais. A integração de técnicas como Fusão de Informações e modelos de aprendizado profundo, como o YOLOv8, mostra melhorias significativas em precisão e velocidade. Os sistemas propostos oferecem alta precisão e identificação em tempo real, aumentando a segurança e eficácia das operações militares ao eliminar erros humanos e otimizar o processamento de dados.*

1. Introdução

Neste trabalho, investigamos técnicas avançadas para o reconhecimento inteligente e classificação de equipamentos militares, integrando diversas tecnologias de ponta. Utilizamos o modelo YOLOv8 para detecção de objetos em tempo real, aliado ao poder computacional da AWS, garantindo escalabilidade e robustez. A implementação de FastAPI proporcionou uma interface de aplicação eficiente e responsiva, facilitando a comunicação entre os componentes do sistema. Adicionalmente, empregamos a biblioteca PyTorch para desenvolver e treinar os modelos de aprendizado profundo. A utilização de paralelismo com CUDA foi essencial para acelerar o processamento de imagens, otimizando o desempenho e permitindo inferências rápidas. Essa combinação de tecnologias resultou em um sistema altamente preciso e eficiente, minimizando erros humanos e reforçando a segurança e eficácia das operações militares.

2. Metodologia

A metodologia empregada neste trabalho abrange a coleta e o processamento dos dados, o desenvolvimento do modelo de detecção, a implementação do sistema de backend e frontend, além da validação do desempenho do sistema em condições controladas.

2.1. Aquisição de Dados

A coleta de dados foi realizada a partir de uma fonte confiável contendo imagens de aeronaves militares, todas rigorosamente anotadas. O processo de anotação foi conduzido de maneira a garantir que os rótulos fossem consistentes e de alta qualidade, atendendo aos requisitos do treinamento do modelo de aprendizado profundo.

2.2. Pré-processamento

As imagens foram submetidas a uma etapa de pré-processamento que incluiu redimensionamento, normalização e a aplicação de técnicas de aumento de dados, como rotações, reflexões e ajustes de iluminação, visando aumentar a variabilidade e a robustez do modelo. Além disso, o dataset original, que estava no formato PASCAL, foi convertido para um formato compatível com o modelo YOLO, garantindo a integração adequada com a arquitetura de detecção.

2.3. Arquitetura do Modelo

Implementamos o modelo YOLOv8 [YOL] devido à sua capacidade de detectar objetos em tempo real com alta precisão. PyTorch [PyT] foi utilizado para desenvolver e treinar o modelo, aproveitando sua flexibilidade e eficiência.

2.4. Infraestrutura de Treinamento

O treinamento foi realizado na AWS, utilizando instâncias com GPUs para acelerar o processo. O paralelismo com CUDA [CUD] foi essencial para otimizar o desempenho, permitindo que o treinamento fosse conduzido de maneira eficiente e rápida.

2.5. Desenvolvimento do APP

O desenvolvimento do aplicativo SIAM (Sistema de Identificação de Aeronaves Militares) foi realizado utilizando .NET 9 e Avalonia UI, proporcionando uma interface gráfica moderna e multiplataforma. A aplicação permite que os usuários façam o upload de imagens de aeronaves militares, que são analisadas por um modelo de rede neural convolucional personalizado, baseado no YOLOv8, para identificar e classificar as aeronaves. A comunicação entre o aplicativo e o backend é feita através de uma API permitindo que o sistema execute as previsões em tempo real. O uso do Avalonia UI garante que o aplicativo seja executado em diferentes sistemas operacionais. As Figuras:7 e 2 exibem respectivamente as topologias da aplicação inicial e final.

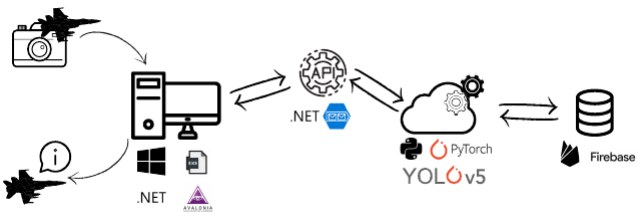

Figura 1. Topologia inicial proposta.

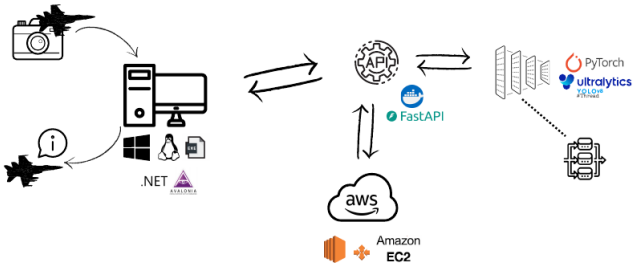

Figura 2. Topologia final implementada.

2.6. Desenvolvimento da API

A API foi desenvolvida utilizando FastAPI para fornecer uma interface eficiente e de alto desempenho para interação com o modelo de detecção. Ela permite a integração simples e rápida do modelo de identificação de aeronaves, oferecendo inferências em tempo real e facilitando a comunicação entre o aplicativo desktop e o backend, com alta escalabilidade e baixo tempo de resposta. A Figura:3 exibe as API's utilizadas no desenvolvimento.

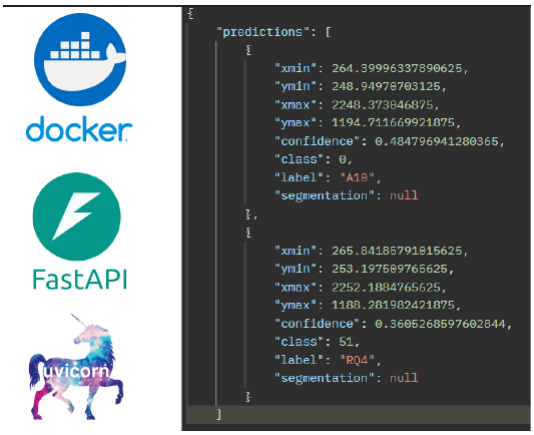

Figura 3. APIs utilizadas no desenvolvimeto.

2.7. Validação e Testes

A validação do modelo foi realizada com um conjunto de dados de teste independente, assegurando sua capacidade de generalização. Para avaliar seu desempenho, foram utilizadas métricas como acurácia, precisão, recall e F1-score, juntamente com a análise de curvas ROC e matrizes de confusão para fornecer uma visão detalhada do comportamento do modelo.

2.8. Implantação

Após a validação, o modelo foi implantado na AWS [AWS], utilizando a infraestrutura de nuvem para garantir alta disponibilidade e escalabilidade. A API, FastAPI, containerizada em Docker, gerencia as solicitações de detecção em tempo real. O sistema final foi testado sob condições reais para garantir sua eficácia e desempenho.

3. Trabalhos correlatos

A abordadem dada em [Ghazouali et al. 2024], avalia diversos algoritmos avançados de detecção de objetos em imagens de satélite, focando na identificação de aeronaves. Foram comparados modelos como *YOLOv5*, *YOLOv8, Faster RCNN, CenterNet, RetinaNet, RTMDet* e *DETR*, todos treinados a partir do zero. O *YOLOv5* se destacou como o modelo mais eficiente, apresentando alta precisão, adaptabilidade e desempenho superior em métricas como *mAP, Recall* e *IoU*. A pesquisa utilizou os conjuntos de dados *HRPlanesV2* e *GDIT* para uma validação rigorosa.

O artigo [Wang et al. 2022] apresenta o método *TransEffiDet* para detecção de aeronaves em imagens aéreas, combinando o algoritmo *EfficientDet* com um módulo *Transformer*. Embora a detecção de aeronaves seja desafiadora devido a fatores como o ambiente adverso e vastidão do céu. O *TransEffiDet* melhora a eficiência na fusão de mapas de características em diferentes escalas. O método utiliza um *Transformer* deformável para analisar correlações de longo alcance, além de um módulo de fusão que integra caracterśticas de curto e longo alcance.

4. Avaliação dos resultados

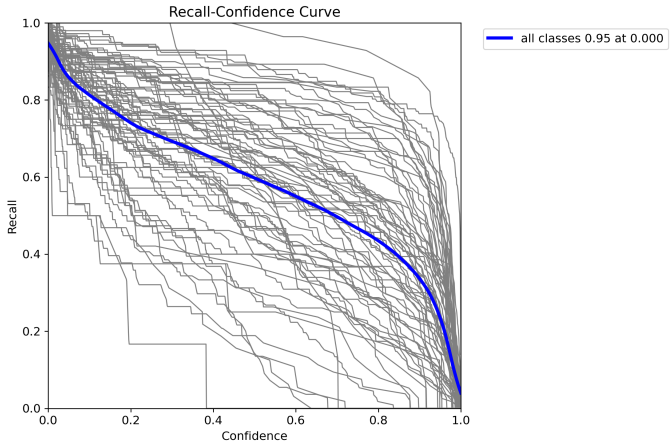

Figura 4. Curva Recall.

4.1. Curva de Precisão-Recall

A análise dos resultados do modelo de reconhecimento de aeronaves hostis começa com a avaliação da Curva de Precisão-Recall. A curva principal, representada em azul na Figura 4, indica o desempenho agregado do modelo para todas as classes, com uma precisão média (mAP) de 0.738 a um limiar de 0.5. As linhas cinzas ao fundo representam as curvas de precisão-recall para diferentes classes individuais. Este gráfico é fundamental para entender a eficácia do modelo em termos de equilíbrio entre precisão e recall, fornecendo uma visão detalhada sobre como o modelo se comporta em diferentes situações e para diferentes classes.

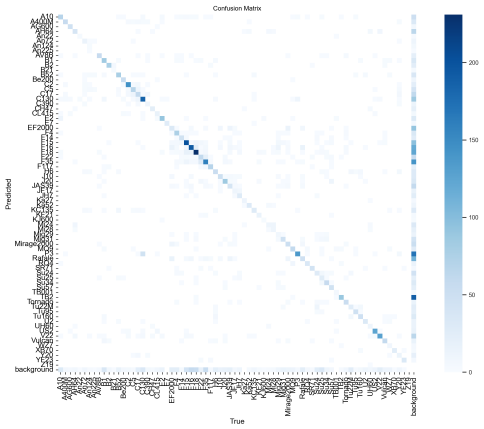

Figura 5. Matriz de confusão.

4.2. Desempenho por Classe

A análise de desempenho por classe, destacada na Figura 5, revela os pontos fortes e fracos do modelo em diferentes categorias de aeronaves. Esta informação é crucial para identificar áreas onde o modelo pode necessitar de ajustes ou aprimoramentos adicionais.

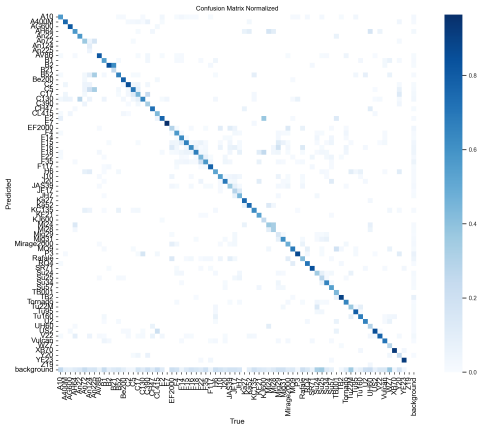

Figura 6. Matriz de confusão normalizada.

4.3. Aceleração com CUDA

A implementação do paralelismo com CUDA, conforme evidenciado na Figura 6, mostrou-se essencial para otimizar o desempenho do modelo. Ao distribuir as operações de processamento de imagem pelos núcleos da GPU, conseguimos reduzir significativamente o tempo de treinamento e inferência. Este ganho de performance é particularmente importante para aplicações em tempo real, onde a rapidez na detecção de ameaças é crítica.

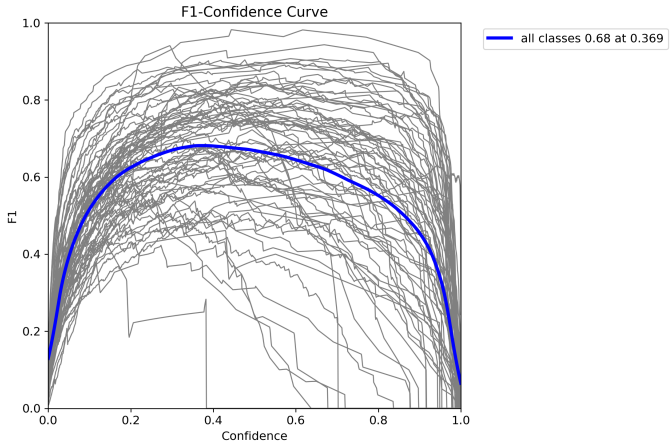

Figura 7. Curva F1 score.

4.4. Infraestrutura na AWS

O uso da infraestrutura da AWS, destacada na Figura 7, proporcionou a escalabilidade necessária para lidar com grandes volumes de dados de treinamento e teste. A combinação de instâncias com GPUs potentes e a flexibilidade da AWS permitiu um desenvolvimento ágil e eficiente do modelo, garantindo sua robustez e capacidade de operação em cenários complexos.

5. Conclusão

O sistema desenvolvido para a identificação de aeronaves militares, utilizando o modelo YOLOv8 e tecnologias avançadas de computação distribuída e paralela, oferece uma solução robusta e escalável, atendendo às exigências de desempenho e precisão em tempo real (em uma abordagem inicial ingênua). A combinação de aprendizado profundo, computação paralela e sistemas distribuídos, juntamente com a infraestrutura de microsserviços em nuvem, permite que o sistema seja eficiente, flexível e capaz de lidar com um grande volume de dados e requisições simultâneas, fundamentais para o contexto de monitoramento e segurança em larga escala. A Figura:8 exibe a aplicação em uso, onde pode-se verificar as informações das últimas predições realizadas bem como a intuitiva interface gráfica com o usuário. Já na Figura:9, temos a predição de várias aeronaves militares bem como os respectivos modelos.

As redes CNN são amplamente reconhecidas pela sua capacidade de aprender características complexas em dados visuais, o que, no contexto de sistemas de reconhecimento de imagens, permite a classificação precisa de objetos, mesmo sob condições adversas. No entanto, o treinamento de modelos como o YOLOv8 exige grandes quantidades de dados e poder computacional, o que justifica o uso de técnicas de computação paralela e distribuída.

A computação paralela, ao permitir a execução simultânea de múltiplas operações, desempenhou um papel central na otimização do tempo de treinamento do modelo. O paralelismo forte foi utilizado para distribuir tarefas de treinamento entre múltiplos processadores ou GPUs, melhorando o desempenho até certo ponto, mas com um limite

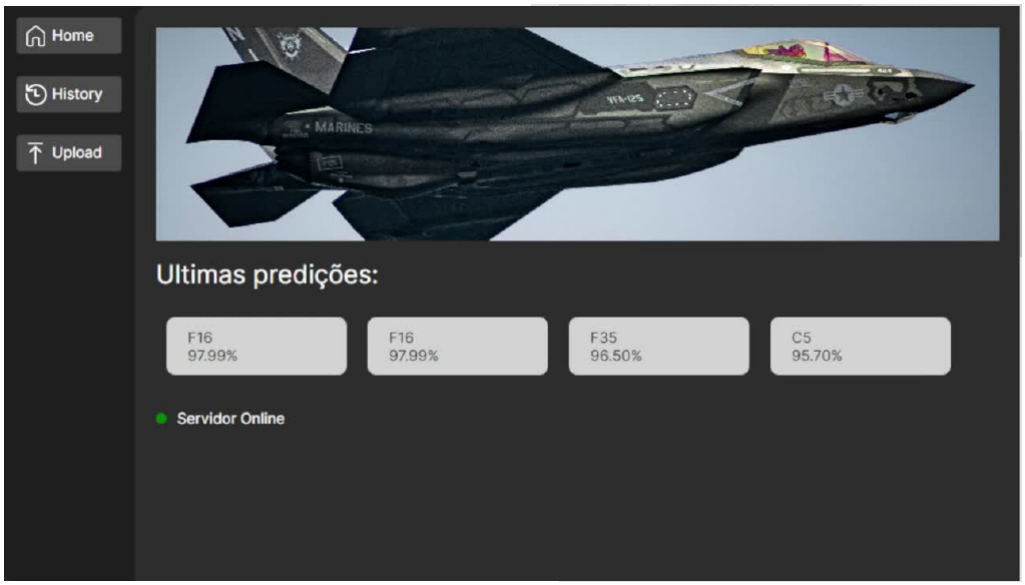

Figura 8. Aplicação de detecção de aeronaves militares em uso.

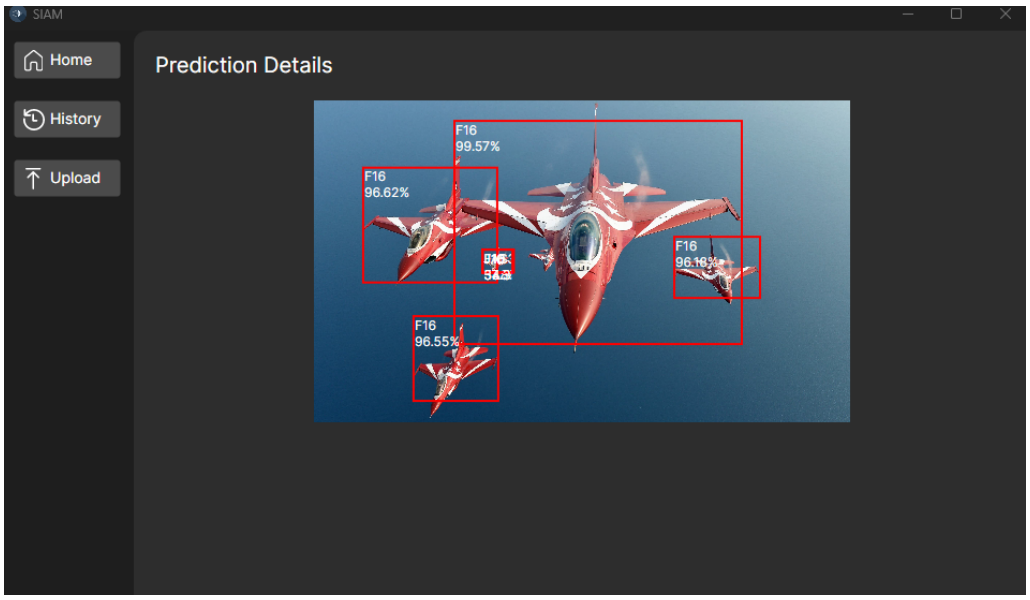

Figura 9. Predição de aeronaves militares e os seus modelos.

determinado pelo overhead de comunicação entre as unidades. Por outro lado, o paralelismo fraco foi adotado para permitir que o sistema se adaptasse ao aumento de dados, escalando de maneira quase linear com a adição de mais recursos de processamento, o que foi particularmente útil durante a execução das predições em tempo real. Essa abordagem garantiu que o sistema fosse capaz de lidar com um número crescente de imagens e requisições sem comprometer a performance.

A infraestrutura de microsserviços, baseada na Amazon Web Services (AWS), com o uso de instâncias EC2 otimizadas para CUDA e balanceamento de carga nativo, foi essencial para garantir a escalabilidade e a resiliência do sistema. A AWS EC2

forneceu uma plataforma robusta e flexível, permitindo que o sistema fosse dimensionado horizontalmente, ou seja, adicionando instâncias de maneira eficiente para lidar com um aumento na carga de trabalho. O balanceamento de carga assegurou que as requisições fossem distribuídas de forma equitativa entre as instâncias, evitando sobrecarga de qualquer componente do sistema e garantindo alta disponibilidade.

A utilização de CUDA para o treinamento e o paralelismo multi-GPU foi fundamental para reduzir o tempo de processamento de grandes volumes de dados, acelerando a execução de operações complexas de aprendizado profundo. Esse avanço tecnológico permitiu que o modelo YOLOv8 fosse treinado em um tempo significativamente menor, o que é crucial em ambientes onde a atualização e adaptação do modelo são frequentes.

Além disso, a integração do sistema com a FastAPI garantiu que o processo de predição fosse realizado em tempo real, permitindo que as imagens fossem analisadas de forma ágil e as decisões tomadas em questão de segundos. Isso é particularmente importante em cenários de defesa, onde a rapidez na identificação de ameaças pode ser um fator determinante para a segurança.

Em suma, o uso de sistemas distribuídos e computação paralela foi crucial para a criação de um sistema eficiente e de alto desempenho, capaz de lidar com grandes volumes de dados e requisições de forma escalável. A combinação das tecnologias de YOLOv8, redes CNN, paralelismo forte e fraco, CUDA, microsserviços em nuvem e AWS EC2 não só atendeu às necessidades imediatas de reconhecimento de aeronaves militares em tempo real, mas também preparou o sistema para crescer de forma flexível, atendendo a futuros desafios e ampliando suas capacidades. O sistema desenvolvido representa um avanço significativo nas tecnologias de defesa, otimizando a identificação e monitoramento de aeronaves e reforçando a segurança nacional.

Referências

[2103.16234] cuconv: A cuda implementation of convolution for cnn inference. https://arxiv.org/abs/2103.16234. (Accessed on 10/15/2024).

Amazon elastic compute cloud documentation. https://docs.aws.amazon.com/ec2/. (Accessed on 10/15/2024).

Pytorch documentation — pytorch 2.5 documentation. https://pytorch.org/docs/stable/index.html. (Accessed on 10/15/2024).

Yolo inferência thread-safe - ultralytics yolo docs. https://docs.ultralytics.com/pt/guides/yolo-thread-safe-inference/%7D%7D. (Accessed on 10/15/2024).

Ghazouali, S. E., Gucciardi, A., Venturini, F., Venturi, N., Rueegsegger, M., and Michelucci, U. (2024). Flightscope: A deep comprehensive review of aircraft detection algorithms in satellite imagery. *arXiv preprint arXiv:2404.02877*.

Wang, Y., Wang, T., Zhou, X., Cai, W., Liu, R., Huang, M., Jing, T., Lin, M., He, H., Wang, W., et al. (2022). Transeffidet: aircraft detection and classification in aerial images based on efficientdet and transformer. *Computational Intelligence and Neuroscience*, 2022(1):2262549.