Análise de Desempenho e Viabilidade de Modelos YOLOv8 para Detecção de Pás Eólicas em Ambientes Embarcados

Gustavo F. Freitas Sampaio[1], **Carlos Yan M. Ferreira**[1],
Luis Guilherme A. G. do Amaral[1], **Antonio Wendell de O. Rodrigues**[1]

[1]Laboratório de Inovação Tecnológica (LIT)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Fortaleza – CE – Brasil

{gustavo.fernandes62,carlos.yan13,guilherme.amaral61}@aluno.ifce.edu.br

wendell@ifce.edu.br

Resumo. *Este trabalho apresenta uma análise de desempenho e viabilidade de modelos YOLOv8 aplicados à detecção de pás de turbinas eólicas em cenários de inspeção aérea, considerando diferentes ambientes computacionais, incluindo plataformas em nuvem com GPU e dispositivos embarcados com restrições de recursos. A partir da análise de latência, FPS, consumo de memória e precisão (mAP50), os resultados evidenciam limitações de implantação em dispositivos embarcados quando a aceleração por hardware não está disponível, reforçando a necessidade de estratégias específicas de otimização para aplicações em edge computing.*

1. Introdução

A inspeção de turbinas eólicas por drones consolidou-se como uma alternativa eficiente para reduzir custos operacionais e riscos em ambientes de difícil acesso. Nesse contexto, técnicas de visão computacional possibilitam automatizar a análise de ativos e auxiliar na identificação precoce de falhas estruturais.

Com o avanço de aplicações de inteligência artificial em ambientes distribuídos, o conceito de *edge computing* ganhou destaque ao aproximar o processamento da fonte de aquisição dos dados [Shi et al. 2016]. Entretanto, aplicações embarcadas ainda enfrentam desafios relacionados à capacidade computacional, consumo energético e compatibilidade do ambiente de software, especialmente na execução de modelos de *deep learning* em tempo real.

Modelos YOLO apresentam bons resultados em detecção de objetos, porém sua execução em dispositivos embarcados de baixo consumo, como a NVIDIA Jetson Nano [NVIDIA 2020], ainda depende de estratégias de otimização e aceleração. Este trabalho investiga o desempenho dos modelos YOLOv8n, YOLOv8s e YOLOv8m em ambientes computacionais distintos, comparando uma plataforma em nuvem com GPU e um cenário embarcado restrito.

Como contribuição, este trabalho apresenta uma análise experimental do impacto da arquitetura dos modelos e das limitações computacionais no desempenho de inferência em aplicações de inspeção aérea.

2. Trabalhos Relacionados

Estudos recentes têm investigado o uso de visão computacional em plataformas embarcadas e drones. Cazzato [Cazzato et al. 2020] e Palmas e Andronico [Palmas and Andronico 2022] avaliaram a aplicabilidade de modelos YOLO em cenários aéreos, destacando limitações associadas ao processamento embarcado. Xu et al. [Xu et al. 2024] observaram que YOLOv8n apresenta melhor escalabilidade para detecção de objetos pequenos e médios. Rey et al. [Rey et al. 2025] demonstraram inferência embarcada com aceleração GPU. Estratégias de otimização como compressão, quantização e *pruning* também têm sido exploradas para reduzir o custo computacional [Han et al. 2016].

Diferentemente dos trabalhos apresentados, esta pesquisa busca avaliar o comportamento de modelos YOLOv8 em ambientes computacionais distintos, considerando tanto o desempenho bruto quanto as limitações práticas de implantação em hardware embarcado.

3. Métodos e Materiais

3.1. Dataset e Plataformas

O *dataset* utilizado foi desenvolvido especificamente para o cenário de inspeção aérea de pás eólicas, contendo 324 imagens anotadas capturadas por drones. As imagens foram divididas em 70% para treinamento, 20% para validação e 10% para teste, sendo padronizadas em resolução de 640 pixels.

O treinamento dos modelos foi realizado no Google Colab utilizando GPU NVIDIA T4. Os testes de inferência foram conduzidos tanto no Google Colab quanto em uma NVIDIA Jetson Nano utilizando JetPack 4.6.6.

Na plataforma embarcada, a execução foi limitada pelo ambiente de software disponível. O JetPack 4.6.6 utiliza dependências baseadas em Python 3.6, dificultando a utilização de versões recentes do framework Ultralytics e da aceleração CUDA.

Dessa forma, os experimentos na Jetson Nano foram conduzidos considerando um cenário *CPU-bound*, representando uma condição restritiva de implantação.

3.2. Metodologia de Profiling

A avaliação foi realizada utilizando um vídeo de 2 minutos capturado por drone. Cada experimento foi executado três vezes e os resultados apresentados correspondem à média das execuções.

No Google Colab, foram avaliados os modelos YOLOv8n, YOLOv8s e YOLOv8m utilizando resolução de entrada de 640 pixels.

No Jetson Nano, devido às limitações computacionais da plataforma, foram avaliados apenas os modelos YOLOv8n e YOLOv8s, representando arquiteturas leves e intermediárias da família YOLOv8. Foram utilizadas resoluções de entrada de 320 e 416 pixels.

A latência foi mensurada utilizando um script em Python baseado na biblioteca *time*. Durante o treinamento foram coletadas métricas de precisão (mAP50) e consumo de GPU. Na etapa de inferência foram analisadas latência, FPS e uso de memória.

4. Resultados

4.1. Treinamento dos Modelos

Tabela 1. Resultados de treinamento (Google Colab com GPU T4)

Modelo	Input	GPU Mem (GB)	mAP50	Treino (s)
YOLOv8n	640	2.11	0.990	236.306
YOLOv8s	640	4.47	0.993	308.540
YOLOv8m	640	6.63	0.990	779.069

A Tabela 1 demonstra que o YOLOv8s apresentou o maior mAP50 (0.993), mantendo menor custo computacional que o YOLOv8m.

4.2. Inferência em Google Colab

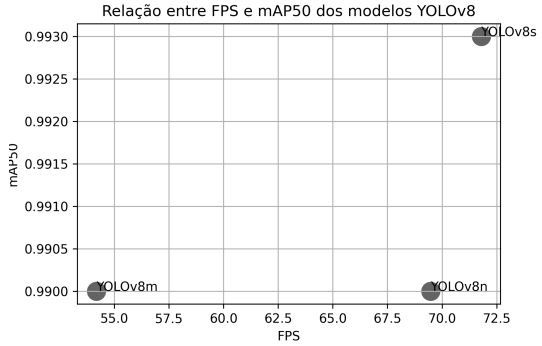

Figura 1. Trade-off entre desempenho (FPS) e precisão (mAP50)

Tabela 2. Execução dos modelos YOLOv8 em Google Colab

Modelo	GPU Mem (GB)	Latência (ms)	FPS
YOLOv8n	0.15	14.39	69.48
YOLOv8s	0.21	13.93	71.79
YOLOv8m	0.32	18.45	54.19

Analisando a Tabela 2, o YOLOv8s destacou-se com a melhor relação entre latência e taxa de processamento, confirmando os achados observados na etapa de treinamento.

4.3. Inferência em Jetson Nano

Mesmo com a redução da resolução de entrada, os ganhos de desempenho foram insuficientes para atingir operação em tempo real. A execução em cenário *CPU-bound* resultou em latência elevada, evidenciando a dependência de aceleração computacional e otimizações específicas para aplicações em edge computing [Zhang et al. 2023].

Em contraste com o ambiente em nuvem, a Jetson Nano apresentou limitações significativas. A execução em cenário *CPU-bound* resultou em latência elevada e FPS insuficientes para operação em tempo real. Mesmo com a redução da resolução de entrada,

Tabela 3. Desempenho dos modelos YOLOv8 na Jetson Nano sem aceleração GPU

Modelo	Input	RAM (%)	Latência (ms)	FPS
YOLOv8n	320	37.0	300.477	3.35
YOLOv8n	416	37.2	470.338	2.12
YOLOv8s	320	38.3	776.645	1.28

os ganhos de desempenho foram insuficientes para atingir níveis operacionais adequados, o que está alinhado com estudos recentes que indicam forte dependência de otimizações específicas para *edge computing* [Zhang et al. 2023].

5. Conclusão

Os resultados mostraram que modelos compactos, especialmente YOLOv8s, apresentam desempenho adequado em plataformas com GPU. Entretanto, em dispositivos embarcados sem aceleração disponível, a latência aumenta significativamente, inviabilizando aplicações em tempo real.

Dessa forma, a implantação de modelos de visão computacional em edge computing depende da integração entre arquitetura do modelo, hardware e técnicas de otimização como TensorRT, quantização e aceleração por GPU.

Referências

Cazzato, D., Cimarelli, C., Sanchez-Lopez, J. L., Voos, H., and Leo, M. (2020). A survey of computer vision methods for 2d object detection from unmanned aerial vehicles. *Journal of Imaging*, 6(8):78.

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*.

NVIDIA (2020). Jetson nano developer kit.

Palmas, A. and Andronico, P. (2022). Deep learning computer vision algorithms for real-time uav on-board camera image processing. In *NATO AVT-353 Research Workshop: Artificial Intelligence in Cockpits for UAVs*.

Rey, L., Bernardos, A. M., Dobrzycki, A., et al. (2025). A performance analysis of you only look once models for deployment on constrained computational edge devices in drone applications. *arXiv preprint arXiv:2502.15737*.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

Xu, L., Zhao, Y., Zhai, Y., Huang, L., and Ruan, C. (2024). Small object detection in uav images based on yolov8n. *International Journal of Computational Intelligence Systems*, 17:223.

Zhang, Q., Wang, Y., Liu, M., et al. (2023). Benchmarking deep learning inference performance on edge devices. *IEEE Access*, 11:45000–45015.