

YOLOv8 para Detecção de Embarcações em Ambiente Fluvial Amazônico

Paulo Henrique Barros Diniz¹, Sarita Mazzini Bruschi¹

¹ Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo (ICMC – USP)
São Carlos – SP – Brasil
paulo.diniz@usp.br, sarita@icmc.usp.br

Resumo. A família YOLOv8 é investigada neste artigo para detecção de embarcações em ambientes fluviais amazônicos, cenário pouco coberto por bases genéricas como COCO e ImageNet. As cinco variantes oficiais (n, s, m, l, x) foram treinadas em um dataset próprio de embarcações regionais (regional, voadeira, expresso, balsa, ferry), coletado em sete pontos do Amazonas e totalizando 1.344 instâncias anotadas. Os experimentos, com 150 épocas, batch size 8 e resolução 640×640 , foram executados em uma NVIDIA RTX 3050. O YOLOv8l alcançou a melhor precisão geral (71,14% mAP@50), enquanto o YOLOv8s ofereceu o melhor equilíbrio entre precisão (68,87% mAP@50) e tempo de treinamento (16,9 min), mostrando-se mais adequado para instituições locais com poder computacional limitado, como órgãos de fiscalização e equipes de monitoramento fluvial.

Abstract. The YOLOv8 family is investigated in this paper for vessel detection in Amazonian fluvial environments, a scenario poorly covered by generic datasets such as COCO and ImageNet. The five official variants (n, s, m, l, x) were trained on a custom dataset of regional vessels (regional, voadeira, expresso, balsa, ferry), collected at seven points in the state of Amazonas and totaling 1,344 annotated instances. Experiments, with 150 epochs, batch size 8, and 640×640 resolution, were run on an NVIDIA RTX 3050. YOLOv8l reached the best overall accuracy (71.14% mAP@50), while YOLOv8s offered the best balance between accuracy (68.87% mAP@50) and training time (16.9 min), proving more suitable for local institutions with limited computational resources, such as environmental agencies and river monitoring teams.

1. Introdução

O monitoramento de embarcações é uma tarefa crítica para logística, fiscalização ambiental, combate à pesca ilegal e gestão de tráfego fluvial. Na Amazônia, a atividade ganha importância particular: são contabilizados na região cerca de 1.084 km de rodovias [1], frente a aproximadamente 25.000 km de hidrovias navegáveis [2], e os rios são utilizados como principal via de transporte.

A detecção de objetos passou a ser realizada por técnicas baseadas em aprendizado profundo [3], das quais representações hierárquicas dos dados são aprendidas. A família de detectores divide-se em dois grupos: os de dois estágios, como o Faster R-CNN [4], nos quais a precisão é priorizada via regiões candidatas, e os de estágio único, como o SSD [5] e a série YOLO [6], nos quais a detecção é tratada como regressão direta e o tempo real é favorecido. O YOLOv8 [7] consolidou-se como escolha em aplicações fluviais recentes, com *design anchor-free*, cabeça desacoplada e cinco variantes escalonáveis (n, s, m, l, x)¹.

¹ As letras indicam a escala da arquitetura da rede: *nano* (n), *small* (s), *medium* (m), *large* (l) e *extra-large* (x).

Apesar do avanço da área, a maioria dos detectores é treinada em bases genéricas como COCO² e ImageNet³, nas quais a realidade das embarcações fluviais amazônicas é mal representada. A generalização diminui porque as embarcações, a iluminação e os reflexos na água são muito diferentes do cenário para o qual os modelos foram calibrados. Não encontramos, na literatura, estudos que avaliem detectores modernos no contexto fluvial amazônico, e foi essa lacuna que motivou o trabalho.

Trabalhos recentes recorrem ao YOLO em detecção de embarcações, mas em outros contextos. [8] avaliaram o detector em monitoramento marítimo aéreo; [9] combinaram YOLOv10 com super-resolução para imagens degradadas; e [10] compararam arquiteturas em datasets públicos de navios. Nenhum aborda o cenário fluvial amazônico.

Diferentemente desses estudos, este trabalho apresenta uma avaliação experimental das cinco variantes do YOLOv8 aplicadas a um *dataset* de embarcações coletado em sete pontos do estado do Amazonas. São contribuições: (i) avaliação sistemática do YOLOv8 em cenário fluvial amazônico; (ii) base anotada com cinco classes típicas.

2. Metodologia

O estudo foi conduzido nas seguintes etapas: curadoria da base de dados, configuração do ambiente computacional, execução do treinamento e análise comparativa das métricas. Considerando as restrições computacionais e a variabilidade visual característica da região amazônica. Detalhes adicionais sobre o protocolo de aquisição, anotação e curadoria do *dataset* serão apresentados em trabalho futuro dedicado.

2.1. Curadoria da Base de Dados

Um conjunto de dados próprio foi utilizado, composto por imagens de embarcações capturadas em múltiplos pontos do estado do Amazonas, cobrindo áreas portuárias, vias navegáveis e regiões de operações fluviais. As embarcações foram anotadas em cinco classes típicas da região (*regional*, *voadeira/canoa*, *expresso*, *balsa* e *ferry*), totalizando mais de mil instâncias rotuladas. O *dataset* foi dividido de forma estratificada em subconjuntos de treinamento, validação e teste, preservando a proporção de instâncias por classe (Tabela 1).

Tabela 1. Distribuição de objetos anotados por classe e subconjunto.

Classe	Treinamento	Validação	Teste
Regional	398	42	74
Voadeira/canoa	301	29	51
Expresso	187	16	28
Balsa	150	21	32
Ferry	34	3	9
Total	1.070	111	163

2.2. Ambiente Computacional

Os experimentos foram conduzidos em um notebook Dell G15, equipado com processador Intel Core i5-13450HX, 16 GB de memória RAM e GPU NVIDIA RTX 3050 (6 GB de VRAM), o que limitou o *batch size* a 8. O protocolo foi padronizado: 150 épocas, resolução de 640×640 pixels, otimização AdamW e monitoramento via MLflow. Os melhores pesos por mAP@50 na validação foram usados na avaliação final.

²<https://cocodataset.org/>

³<https://www.image-net.org/>

2.3. Métricas de Avaliação

A avaliação foi feita exclusivamente no conjunto de teste, com *Precision*, *Recall*, *mAP@50* e *mAP@50-95*. O tempo total de treinamento foi também reportado como indicador de custo computacional.

3. Resultados

Os resultados não apontam um único vencedor. A Tabela 2 e a Figura 1 mostram que o YOLOv8l leva o *mAP@50* com 71,14%, mas perde em *mAP@50-95* (39,85%) e *Precision* (68,45%). YOLOv8m lidera em *mAP@50-95* (43,01%) e fica em segundo em *mAP@50* (70,97%) e *Recall* (67,94%). YOLOv8s tem a *Precision* mais alta (75,29%). YOLOv8n é o mais rápido (11,7 min), mas não vence em nenhuma métrica de acurácia. A melhor opção, portanto, depende do critério que se prioriza.

Tabela 2. Desempenho das variantes do YOLOv8 (GPU RTX 3050, 150 épocas).

Variante	Tempo (min)	mAP@50 (%)	mAP@50-95 (%)	Precision (%)	Recall (%)
YOLOv8n	11,7	67,56	40,25	67,29	59,01
YOLOv8s	16,9	68,87	42,34	75,29	61,70
YOLOv8m	33,9	70,97	43,01	73,08	67,94
YOLOv8l	51,9	71,14	39,85	68,45	72,97
YOLOv8x	78,0	66,81	40,57	64,98	67,78

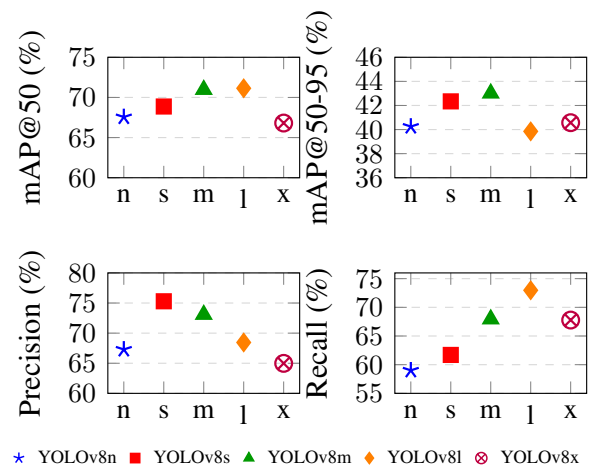


Figura 1. Comparação visual das métricas de avaliação.

O YOLOv8x apresenta resultados que mais se afastam do esperado. Ele não vence em nenhuma métrica, fica em último em *mAP@50* (66,81%) e em *Precision* (64,98%), e ainda assim demanda 78 min de treinamento, o equivalente a 6,7 vezes o tempo do YOLOv8n e 2,3 vezes o do YOLOv8m.

Uma hipótese para esse desempenho anômalo está na interação entre o *batch size* reduzido (8) e o volume de parâmetros das variantes maiores. Redes mais profundas, como YOLOv8l e YOLOv8x, possuem um número maior de camadas de normalização em lote (*batch normalization*), cujas estatísticas de média e variância são estimadas a partir dos exemplos do próprio lote. Com *batch sizes* pequenos, essas estimativas tornam-se mais ruidosas, o que tende a desestabilizar o treinamento e prejudicar a convergência, efeito que se acentua conforme a profundidade da rede aumenta [11]. Essa hipótese é compatível com o padrão observado: o ganho de capacidade das variantes maiores não se traduziu em ganho de acurácia, e o YOLOv8x, a mais profunda delas, foi a que mais sofreu com essa limitação.

O trade-off entre as variantes intermediárias também merece atenção. Entre YOLOv8m e YOLOv8l, o ganho de *mAP@50* é de apenas 0,17 ponto percentual (70,97% para 71,14%), mas o tempo de treinamento do YOLOv8l é 53% maior (51,9 min contra 33,9 min). O YOLOv8l compensa o custo extra apenas no *Recall*, métrica em que lidera com 72,97%, o que o torna interessante quando minimizar falsos negativos é prioridade. Para hardware limitado, a escolha recai sobre o YOLOv8s, que entrega 97% do *mAP@50* do YOLOv8m (68,87% contra 70,97%) em metade do tempo (16,9 min contra 33,9 min) e, por isso, oferece a melhor razão acurácia/custo entre as variantes competitivas.

4. Conclusão

Este trabalho comparou as cinco variantes do YOLOv8 em um *dataset* de embarcações fluviais do Amazonas. O resultado é que a escolha da variante depende da restrição de cada aplicação, e não existe uma vencedora única, embora o YOLOv8m se destaque pela consistência, sendo a única variante que figura entre as duas primeiras posições em três das quatro métricas avaliadas (mAP@50, mAP@50-95 e Precision), o que a torna a opção mais equilibrada quando nenhum critério específico é priorizado. Para hardware limitado, YOLOv8s é a opção de melhor razão acurácia/custo: entrega 97% do mAP@50 do YOLOv8m em metade do tempo de treinamento, e ainda lidera em Precision (75,29%). Quando a acurácia estrita é prioridade, YOLOv8m é a melhor escolha, com 43,01% de mAP@50-95 e tempo intermediário (33,9 min). Para cenários em que minimizar falsos negativos é o mais importante, YOLOv8l lidera com 71,14% de mAP@50 e 72,97% de Recall, embora exija 53% mais tempo que o YOLOv8m. O YOLOv8x, por outro lado, não vence em nenhuma métrica, o que indica que sua capacidade adicional não se converte em ganho para o porte deste *dataset*.

Como limitações, o *dataset* ainda é restrito (1344 instâncias) e desbalanceado: a classe *ferry*, com 34 instâncias, foi a pior detectada. O *dataset* será disponibilizado publicamente em trabalho futuro, permitindo sua avaliação e comparação por outros pesquisadores. Os próximos passos incluem ampliar a base, em especial para *ferry*, aplicar técnicas de aumento de dados e *focal loss* para atenuar o desbalanceamento, avaliar o YOLOv8 em conjunto com técnicas de super-resolução para imagens degradadas, e estender o estudo a outras regiões fluviais para verificar se os padrões se mantêm fora do Amazonas.

Referências

- [1] Agência Nacional de Águas e Saneamento Básico. Regiões Hidrográficas Brasileiras. <https://www.gov.br/ana/pt-br/assuntos/gestao-das-aguas/panorama-das-aguas/regioes-hidrograficas>, 2017. Accessed: 2026-05-15.
- [2] Agência Nacional de Transportes Aquaviários. Anuário Estatístico da Hidrovia do Amazonas. <https://www.gov.br/antag/pt-br>, 2022.
- [3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.
- [5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.
- [6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [7] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, 2023. Software.
- [8] J Sharafaldeen, M Rizk, D Heller, Amer Baghdadi, and J-Ph Diguët. Marine object detection based on top-view scenes using deep learning on edge devices. In *2022 International Conference on Smart Systems and Power Management (IC2SPM)*, pages 35–40. IEEE, 2022.
- [9] Ao Wang, Hui Chen, Zhuangwei Liu, Jianlong Chen, Ziyi Liu, Haoyu Ke, Jian Sun, Mingyu Jia, et al. YOLOv10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.
- [10] Hyago VLB Silva, Felipe AP de Figueiredo, and Samuel B Mafra. Performance evaluation of edge computing object detection models for maritime surveillance on a raspberry pi. In *2024 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE, 2024.
- [11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 2015.