Analysis of CNN architectures for object detection in drone images with a focus on Aedes aegypti

Euller S. Miranda[1], **Tércio A. dos Santos Filho**[1], **Sérgio F. da Silva**[1]

[1]Departamento de Computação – Instituto de Biotecnologia,
Universidade Federal de Catalão (UFCAT)
Caixa Postal 15.064 – 91.501-970 – Catalão – GO – Brazil

eullersm12@gmail.com, {tercioas,sergio}@ufcat.edu.br

Abstract. *Diseases transmitted by the Aedes aegypti mosquito, such as dengue, Zika, and chikungunya, represent a critical challenge to public health. Manual identification of breeding sites is an exhaustive task. This work proposes an automated system for detecting potential sources of stagnant water using Convolutional Neural Networks (CNNs). The methodology consisted of creating annotated databases from MBG and MBG version 2 images, as well as training and evaluating the YOLOv8, SSD, and Faster R-CNN models, exploring the use of different optimizers and variations in batch size. The results demonstrate that CNNs are promising for detecting potential sources of stagnant water, especially the YOLO architecture with the SGD optimizer.*

Resumo. *As doenças transmitidas pelo mosquito Aedes aegypti, como dengue, zika e chikungunya, representam um desafio crítico para a saúde pública. A identificação manual de focos de reprodução é uma tarefa exaustiva. Este trabalho propõe um sistema automatizado para a detecção de potenciais focos de água parada, utilizando Redes Neurais Convolucionais (CNNs). A metodologia consistiu na criação de bases de dados anotadas a partir de imagens da MBG e da MBG versão 2, além do treinamento e da avaliação dos modelos YOLOv8, SSD e Faster R-CNN, explorando o uso de diferentes otimizadores e variações no tamanho dos lotes. Os resultados demonstram que as CNNs são promissoras para a detecção de potenciais focos de água parada, especialmente a arquitetura YOLO com otimizador SGD.*

1. Introdução

Doenças transmitidas por mosquitos respondem por mais de 17% das doenças infecciosas, causando 700 mil mortes anuais [World Health Organization 2020, Centers for Disease Control and Prevention 2025]. No Brasil, o *Aedes aegypti* é o principal vetor de dengue, Zika e chikungunya [Ministério da Saúde 2025]. O mosquito prolifera em áreas urbanas devido à alta densidade populacional e à oferta de criadouros (água parada). O crescimento urbano desordenado exige mecanismos públicos para eliminação desses focos [Ministério da Saúde 2025]. Em 2024, mais de 6 mil mortes foram registradas [Ministério da Saúde 2025]. Com a baixa disponibilidade e resistência à vacina, o combate ao mosquito é o foco principal. As estratégias atuais (inspeções domiciliares) são corretas, mas carecem de ferramentas para mapear focos em áreas extensas e de difícil acesso [Ministério da Saúde 2025].

Diante desse cenário, nesta pesquisa foram investigados métodos de visão computacional baseados em redes neurais convolucionais para a detecção de focos de água parada (lixo, recipientes expostos, pneus abandonados, piscinas e caixas d'água sem o devido cuidado) através de imagens capturadas por drone.

Para viabilizar a investigação, foram utilizadas duas bases de dados provenientes do *Mosquitoes Breeding Grounds* (MBG) Video Database [Passos et al. 2021, Signals, Multimedia, and Telecommunications Laboratory (COPPE/Poli/UFRJ) 2026]. A MBG-v1 contém 13 sequências de vídeo com 6 classes de objetos anotados, totalizando 3.038 instâncias, enquanto a MBG-v2, mais robusta e realista, é composta por 26 sequências com 10 classes e mais de 24 mil anotações, com voos padronizados a 40 metros de altitude. A partir dessas bases, foram conduzidos quatro experimentos com diferentes objetivos: (i) avaliar o impacto de otimizadores e tamanhos de lote na primeira base; (ii) investigar o comportamento dos modelos na segunda base, mais desafiadora; (iii) aplicar e avaliar técnicas de aumento de dados em classes minoritárias; e (iv) avaliar o desempenho dos modelos especificamente na detecção de objetos menores.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta os fundamentos teóricos sobre redes neurais e detecção de objetos; a Seção 3 descreve trabalhos correlatos; a Seção 4 detalha a metodologia empregada, incluindo a construção das bases de dados e os experimentos realizados; a Seção 5 apresenta e discute os resultados obtidos; e a Seção 6 traz as conclusões e perspectivas para trabalhos futuros.

2. Trabalhos Correlatos

Esta seção apresenta uma análise consolidada de trabalhos recentes que aplicam técnicas de visão computacional e aprendizado profundo para identificação de potenciais criadouros de vetores transmissores de doenças, com ênfase no *Aedes aegypti*. Em vez de uma descrição individual trabalho por trabalho, os estudos são examinados sob um conjunto de aspectos comuns como modo de coleta de imagens, algoritmos, objetos detectados, métricas e aplicações, permitindo traçar o panorama atual e identificar tendências e lacunas da área.

2.1. Plataformas de coleta de imagens

A maioria dos trabalhos utiliza VANTs (DJI Phantom, Mavic) com câmeras RGB, devido ao baixo custo e acesso a áreas de difícil fiscalização [Yu2 2024, Cunha et al. 2021, Moran-Landa et al. 2022, Carrasco-Escobar et al. 2022, Passos et al. 2021, Trujillano et al. 2023]. Sensores multiespectrais e térmicos também são explorados [Carrasco-Escobar et al. 2022]. Diferentemente, [Laranjeira et al. 2024] emprega imagens de fachadas do Google Street View, dispensando drones.

2.2. Algoritmos de visão computacional

Todos os trabalhos encontrados para esta tarefa empregam Redes Neurais Convolucionais (CNNs), sendo que a maioria opta por arquiteturas consolidadas de detecção de objetos. Os detectores de estágio único, como YOLO (nas versões v3 e v7), são amplamente utilizados por seu equilíbrio entre precisão e velocidade, alcançando métricas como F1-score e mAP superiores a 0,99 em conjuntos de dados controlados [Yu2 2024, Moran-Landa et al. 2022]. Já arquiteturas de dois estágios, como Faster R-CNN, são empregadas quando a precisão da localização é crítica, obtendo mAP de 87,5%

para caixas d'água e 90,2% para piscinas [Cunha et al. 2021]. [Passos et al. 2021] combinam Faster R-CNN com um módulo de consistência espaço-temporal para explorar a informação temporal de vídeos, reduzindo falsos positivos e melhorando a detecção de objetos como pneus e reservatórios. U-Net e attention U-Net são usados em segmentação de corpos d'água (Dice até 0,75) [Trujillano et al. 2023]. [Laranjeira et al. 2024] propõem a PCINet (EfficientNetV2) para classificação de fachadas, mostrando que a análise indireta pode ser tão eficaz quanto a detecção direta.

2.3. Objetos de detecção e métricas

A diversidade de objetos-alvo encontrados na literatura reflete a variedade de possíveis criadouros, como baldes, pneus, vasos, garrafas, caixas d'água, piscinas e plantas aquáticas [Yu2 2024, Moran-Landa et al. 2022, Carrasco-Escobar et al. 2022], com o número de classes variando de 3 a 9 conforme o estudo [Yu2 2024, Trujillano et al. 2023]. As métricas de avaliação mais comuns foram precisão, revocação, F1-score e mAP [Cunha et al. 2021, Laranjeira et al. 2024, Moran-Landa et al. 2022, Carrasco-Escobar et al. 2022, Passos et al. 2021]. Embora precisões acima de 98% sejam relatadas em cenários controlados [Moran-Landa et al. 2022, Carrasco-Escobar et al. 2022], os conjuntos de dados são frequentemente pequenos [Passos et al. 2021]. [Passos et al. 2021] destaca que a detecção em vídeos é mais robusta que ortomosaicos estáticos, por preservar consistência temporal.

2.4. Tendências e lacunas

A análise conjunta revela que o campo está em franca evolução, com clara preferência por abordagens baseadas em aprendizado profundo e *transfer learning* a partir de bases gerais (ImageNet, COCO). No entanto, algumas lacunas persistem: (i) a escassez de bases de dados públicas e anotadas para criadouros em diferentes contextos geográficos; (ii) a necessidade de validação em larga escala e em condições operacionais reais, com variações climáticas e de iluminação; (iii) a integração de múltiplas fontes de dados para modelos preditivos mais robustos; e (iv) a exploração de arquiteturas mais leves para execução embarcada em drones de baixo custo.

3. Metodologia

Esta seção descreve os procedimentos de obtenção das imagens para a construção das bases de dados e os métodos utilizados.

3.1. Construção das Bases de Dados

Foram utilizadas duas versões da base *Mosquitoes Breeding Grounds* MBG, disponível em [Laboratório de TeleMídia - UFRJ 2026]. A MBG-v1 contém 13 vídeos, 6 classes e 3.038 anotações, com altitudes variadas [Passos et al. 2021]. A MBG-v2 é mais realista, com 26 vídeos, 10 classes, voos padronizados a 40 m e dados de telemetria.

Quatro bases de dados foram desenvolvidas a partir de imagens extraídas da MBG-v1 e MBG-v2. Para cada vídeo, foram selecionadas no máximo 100 imagens, coletadas em intervalos definidos pela razão entre o total de imagens do vídeo e 100. As anotações foram realizadas na plataforma Roboflow [Roboflow 2025]. As imagens foram padronizadas para resolução 1024×1024 pixels: imagens maiores foram particionadas em

recortes; recortes menores receberam *padding* para manter consistência, preservando as caixas delimitadoras.

A MBG-v1 possui 6 classes (*bottle, bucket, pool, puddle, tire, watertank*), com 1.626 imagens e 3.038 anotações [Roboflow Universe 2026a].

A MBG-v2 unifica as classes de lixeiras em *bin* e remove *dumpsters*, resultando em 10 classes (*bin*, *bottle*, *bucket*, *plastic bag*, *pool*, *potted plant*, *puddle*, *storm drain*, *tire* e *watertank*), com 10.654 imagens e 21.698 anotações [Roboflow Universe 2026b].

A MBG-v2-augmented aplica aumento de dados (rotação, inversão, brilho, contraste, HSV) a imagens com *bin, bottle, bucket, tire*, afetando também a quantidade dos outros objetos presentes. O aumento incide apenas sobre treino e validação. A base resultante tem 14.001 imagens e 31.569 anotações [Roboflow Universe 2026c].

A MBG-v2-small-objects contém apenas as classes *bottle, bucket, plastic bag, tire*, totalizando 1.716 imagens e 3.052 anotações [Roboflow Universe 2026d].

3.2. Descrição da Metodologia Aplicada

Visando à investigação de arquiteturas de CNNs e ao treinamento de modelos para a detecção de criadouros de mosquitos, os experimentos foram executados em Python 3.10, com suporte às bibliotecas torch 1.13 e Ultralytics 8.0.228, além de CUDA v12.1.66. Os experimentos foram realizados em uma estação de trabalho equipada com processador Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz. Em relação à GPU, foram utilizadas, conforme a disponibilidade, uma NVIDIA GeForce RTX 3090 ou uma NVIDIA Quadro P6000. A seguir serão descritos os experimentos feitos em cada base de dados.

Layout dos experimentos sob MBG-v1: a base foi dividida em 1.109 imagens (68%) para treinamento, 360 (22%) para validação e 157 (10%) para teste. As imagens que compõem cada partição foram selecionadas aleatoriamente pela plataforma Roboflow e redimensionadas para 512×512 pixels. A Tabela 1 apresenta a distribuição das anotações de cada classe entre as partições. Foram utilizadas as arquiteturas YOLOv8 (implementação da Ultralytics [Jocher et al. 2026]) nos tamanhos *Nano*, *Small*, *Medium*, *Large* e *Extra Large*; SSD com *backbone* VGG16 e pesos pré-treinados no COCO (implementação em PyTorch [Paszke et al. 2019]); e Faster R-CNN com *backbone* ResNet-50, FPN e pesos pré-treinados no COCO (implementação em PyTorch). Foram avaliados sete otimizadores (SGD, Adam, AdamW, Adamax, NAdam, RAdam e RMSProp) e três tamanhos de lote (4, 8 e 16). Para o SSD e o Faster R-CNN, foram adotadas taxas de aprendizado de 0,001 para o SGD e de 0,0001 para os demais otimizadores. Essa diferenciação foi necessária porque, nas execuções preliminares dos modelos implementados em PyTorch, taxas de aprendizado iniciais mais elevadas produziram valores `NaN` na função de perda (*loss*). Para o YOLO foi usada a taxa de aprendizado inicial padrão da Ultralytics, de 0,01. Todos os modelos tiveram foram treinados por até 100 épocas, usando *early stopping* com paciência 10.

Layout dos experimentos sob MBG-v2: a base foi dividida em 7.458 imagens (70%) para treinamento, 2.131 (20%) para validação e 1.065 (10%) para teste, novamente por meio da plataforma Roboflow. A Tabela 2 apresenta a distribuição das anotações de cada classe entre as partições. As imagens foram redimensionadas

Tabela 1. Quantidade de anotações por classe na base MBG-v1.

Classe	Total	Treinamento	Validação	Teste
watertank	2107	1469	454	184
tire	405	262	103	40
bucket	229	146	52	31
bottle	165	102	53	10
pool	124	88	26	10
puddle	8	4	4	0

para 512×512 pixels. Foram mantidas as mesmas arquiteturas e os mesmos sete otimizadores do experimento anterior. O tamanho de lote foi definido como 32 para os modelos YOLO e SSD e como 16 para o Faster R-CNN, devido à limitação de memória de vídeo. O aumento do tamanho dos lotes teve como objetivo reduzir o tempo de treinamento, uma vez que a MBG-v2 possui maior volume de dados. Assim, adotou-se, para cada arquitetura, o maior tamanho de lote permitido pelos recursos computacionais disponíveis. O treinamento foi configurado para 160 épocas, com paciência de 20 épocas, a fim de proporcionar mais tempo para a convergência dos modelos. Para YOLO com RMSProp, ajustou-se a taxa de aprendizado para 0,00001 e desativou-se AMP, pois a configuração padrão resultava em baixo desempenho (orientação da Ultralytics).

Tabela 2. Quantidade de anotações por classe na base MBG-v2.

Classe	Total	Treinamento	Validação	Teste
watertank	10945	7626	2216	1103
pool	2687	1846	574	267
storm drain	2467	1791	438	238
potted plant	2218	1696	383	139
bucket	1569	1100	340	129
tire	959	667	146	146
plastic bag	327	212	82	33
bottle	197	156	27	14
bin	178	111	42	25
puddle	151	101	30	20

Layout dos experimentos sob MBG-v2-augmented: para os experimentos realizados na MBG-v2-augmented, foi novamente adotada a divisão de 70%, 20% e 10% para treinamento, validação e teste, respectivamente. Em seguida, aplicou-se aumento de dados às imagens que continham instâncias das classes *bin*, *bottle*, *bucket* e *tire*. Como as transformações foram realizadas sobre a imagem completa, objetos pertencentes a outras classes presentes nessas imagens também foram afetados. As transformações foram implementadas em Python com o auxílio da biblioteca Albumentations e aplicadas somente às imagens das partições de treinamento e validação. Após esse processo, a MBG-v2-augmented passou a conter 9.834 imagens de treinamento (70%), 2.800 de validação (20%) e 1.367 de teste (10%), todas redimensionadas para 512×512 pixels durante a exportação. A Tabela 3 apresenta a distribuição das anotações de cada classe entre as partições.

Para o treinamento, foram selecionadas as duas configurações de melhor desempenho de cada arquitetura na MBG-v2, considerando a métrica mAP50-95: YOLOv8l e YOLOv8x com SGD; SSD com AdamW e RAdam; e Faster R-CNN com AdamW e RAdam. As demais configurações de treinamento e o procedimento de teste permaneceram iguais aos adotados no experimento sob MBG-v2.

Tabela 3. Quantidade de anotações por classe na base MBG-v2-augmented.

Classe	Total	Treinamento	Validação	Teste
watertank	13042	9128	2616	1298
bucket	4577	3349	948	280
pool	3353	2304	725	324
potted plant	3019	2209	531	279
storm drain	3013	2173	544	296
tire	2770	1897	442	431
bottle	637	430	112	95
bin	498	327	104	67
plastic bag	447	300	109	38
puddle	213	143	45	25

Layout dos experimentos sob MBG-v2-small-objects: nos experimentos realizados na MBG-v2-small-objects, adotou-se novamente como referência a proporção de 70%, 20% e 10% para treinamento, validação e teste, respectivamente. Essa base contém apenas as classes *bottle*, *bucket*, *plastic bag* e *tire*, com o objetivo de avaliar o desempenho das configurações selecionadas na detecção de objetos menores. As imagens foram distribuídas em 1.219 para treinamento, 327 para validação e 170 para teste, mantendo-se o redimensionamento para 512×512 pixels durante a exportação. A Tabela 4 apresenta a distribuição das anotações de cada classe entre as partições. O treinamento foi realizado exclusivamente com a arquitetura YOLOv8, nos tamanhos *Large* e *Extra Large*, pois essas configurações obtiveram os melhores valores de mAP50-95 nos experimentos com a MBG-v2-augmented. Manteve-se o otimizador SGD e, devido ao menor número de imagens dessa base em comparação com a MBG-v2 e a MBG-v2-augmented, o tamanho de lote foi reduzido para 8. Além disso, considerando os ganhos observados com a MBG-v2-augmented, aplicou-se aumento de dados *online* às imagens de treinamento, ou seja, as transformações foram realizadas durante o treinamento. Foram utilizadas as mesmas transformações aplicadas à base anterior. As demais configurações de treinamento permaneceram iguais às do experimento sob MBG-v2, assim como o procedimento de teste dos modelos.

Tabela 4. Quantidade de anotações por classe na base MBG-v2-small-objects.

Classe	Total	Treinamento	Validação	Teste
bucket	1569	1081	336	152
tire	959	721	176	62
plastic bag	327	234	52	41
bottle	197	106	70	21

3.3. Métricas de Avaliação

Para avaliar os modelos, foram utilizadas as métricas padrão da detecção de objetos. A *Precisão* (Precision), dada por $Precision = \frac{TP}{TP+FP}$, mede a proporção de detecções corretas entre todas as positivas. A *Revocação* (Recall), dada por $Recall = \frac{TP}{TP+FN}$, mede a fração de objetos de uma dada classe que foram detectados. O *F1-score*, dado por $F = \frac{2 \times precision \times recall}{precision+recall}$, é a média harmônica entre *Precision* e *Recall*. A principal métrica usada é a *mean Average Precision* (mAP), calculada como a média da área sob a curva precisão-recall (AP) para todas as classes. Adotaram-se dois limiares de *Intersection over the Union* (IoU): mAP50 (IoU = 0,5) e mAP50-95 (média das APs para IoU de 0,5 a 0,95, com passo 0,05). Este último é mais rigoroso e penaliza localizações imprecisas.

4. Resultados

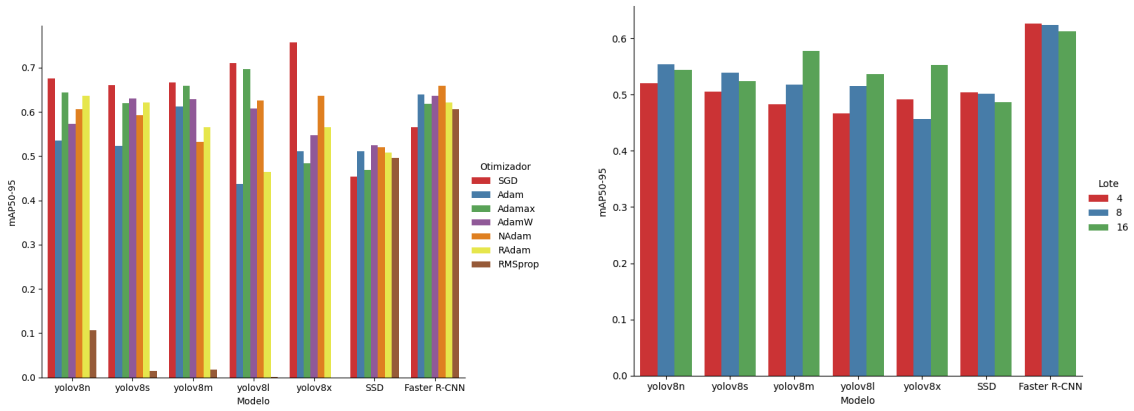
Todas as métricas reportadas referem-se à aplicação dos modelos ao conjunto de teste. A Tabela 5 apresenta os melhores resultados do primeiro experimento para cada modelo base.

Nos experimentos com a MBG-v1, as melhores configurações utilizaram o otimizador SGD para o YOLO e Adam para o SSD e o Faster R-CNN. A combinação YOLOv8x/SGD/lote 8 alcançou mAP50-95 de 0,764. As Figuras 1a e 1b mostram que o otimizador influencia mais que o tamanho do lote. O comportamento da melhor configuração em cada classe pode ser observado na matriz de confusão da Figura 2. A linha adicional referente ao *background* (fundo) representa os falsos negativos, isto é, objetos reais que não foram detectados pelo modelo. A baixa concentração de valores nessa linha indica que a maior parte dos objetos presentes no conjunto de teste foi corretamente detectada. A classe *puddle* não aparece no teste (Tabela 1), logo não é avaliada.

Tabela 5. Melhores resultados por modelo e otimizador na base MBG-v1.

Modelo	Otimizador	Lote	mAP50	mAP50-95
yolov8n	SGD	8	0.942	0.703
yolov8s	SGD	8	0.919	0.714
yolov8m	SGD	16	0.926	0.756
yolov8l	SGD	16	0.920	0.741
yolov8x	SGD	8	0.940	0.764
SSD	Adam	8	0.81	0.53
Faster R-CNN	Adam	8	0.9363	0.6861

Nos experimentos com a MBG-v2, o mAP50-95 dos melhores modelos YOLO, nas versões *Large* e *Extra Large*, caiu para 0,39, conforme a Tabela 6. Essa redução é consistente com a maior complexidade da base, que contém objetos capturados a 40 metros de altitude e dez classes, em comparação com as seis classes da MBG-v1. Na matriz de confusão da Figura 3a, as classes *pool*, *water tank* e *storm drain* apresentam proporções de detecções corretas de 94%, 93% e 90%, respectivamente. Em contraste, os valores observados para *bin*, *bottle* e *bucket* foram de apenas 44%, 21% e 45%. As classes *plastic bag* e *tire* também apresentaram desempenho limitado, com 52% e 51% de detecções corretas, respectivamente. Isso é atribuído à dificuldade de detectar objetos pequenos a 40 m de altitude e ao desbalanceamento das classes (Tabela 2). Além disso, os valores elevados



(a) Relação entre modelo base e otimizador.

(b) Relação entre modelo base e tamanho de lote.

Figura 1. Análise comparativa do mAP50-95 em função dos otimizadores e tamanhos de lote por modelo base.

na linha referente ao *background*, especialmente para as classes menos representadas, indicam uma proporção expressiva de falsos negativos. Esse comportamento sugere que o modelo tende a classificar muitos desses objetos como fundo, deixando de detectá-los.

Tabela 6. Melhores resultados por modelo e otimizador para a base MBG-v2.

Modelo	Otimizador	Lote	mAP50	mAP50-95
yolov8n	SGD	32	0.52	0.31
yolov8s	SGD	32	0.62	0.37
yolov8m	SGD	32	0.62	0.38
yolov8l	SGD	32	0.62	0.39
yolov8x	SGD	32	0.63	0.39
SSD	AdamW	32	0.3760	0.2086
Faster R-CNN	RAdam	16	0.4932	0.2746

No terceiro experimento, o aumento de dados elevou o mAP50-95 do YOLOv8x de 0,39 para 0,4966, conforme a Tabela 7. As matrizes de confusão (Figuras 3a e 3b) mostram ganhos nas classes priorizadas: as taxas de acerto de *bin, bottle, bucket, tire* subiram de 44/21/45/51% para 69/47/65/73%. Também houve redução dos falsos negativos dessas classes, indicada pela diminuição dos valores correspondentes na linha *background*. Esses resultados demonstram que o aumento de dados foi benéfico para reduzir os falsos negativos da base.

No quarto experimento, com apenas classes de objetos menores (*bottle*, *bucket*, *plastic bag* e *tire*), o mAP50-95 caiu para 0.2903 (YOLOv8l) e 0.1085 (YOLOv8x) (Tab. 8 e Fig. 3c), evidenciando a dificuldade inerente à detecção de objetos pequenos.

4.1. Comparação de resultados com estado da arte e análise geral

Em relação à comparação com o estado da arte, é necessário que ela seja realizada considerando as mesmas bases de dados utilizadas nos trabalhos de referência, como a MBG e a MBG-v2. O trabalho de [Passos et al. 2021], utiliza a MBG como base para os experimentos; porém, os resultados apresentados contemplam apenas as classes *tire* e *water*

Figura 2. Matriz de confusão normalizada do YOLOv8x no primeiro experimento.

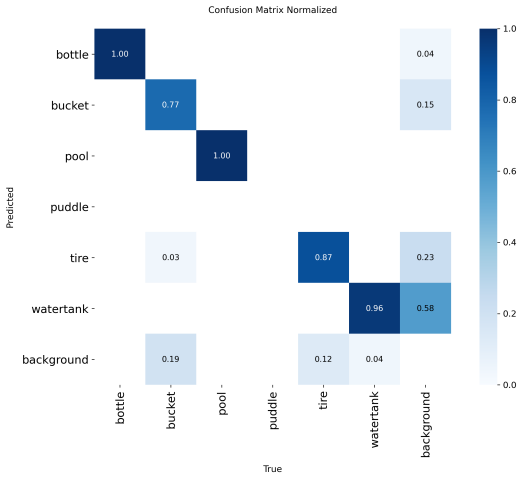
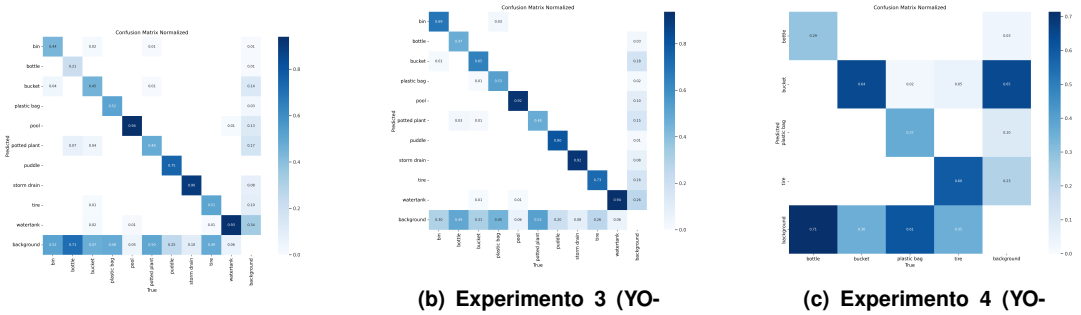


Tabela 7. Melhores resultados por modelo e otimizador terceiro experimento.

Modelo	Otimizador	Lote	mAP50	mAP50-95
yolov8l	SGD	32	0.7392	0.4749
yolov8x	SGD	32	0.7621	0.4966
SSD	AdamW	32	0.4344	0.2438
SSD	RAdam	32	0.4307	0.2410
Faster R-CNN	RAdam	16	0.5552	0.3197
Faster R-CNN	AdamW	16	0.5444	0.3106

Tabela 8. Melhores resultados por modelo e otimizador quarto experimento.

Modelo	Otimizador	Lote	mAP50	mAP50-95
yolov8l	SGD	8	0.5644	0.2903
yolov8x	SGD	8	0.2073	0.1085



(a) Experimento 2 (YOLOv8x, MBG-v2)

(b) Experimento 3 (YOLOv8x, com aumento)

(c) Experimento 4 (YOLOv8l, classes menores)

Figura 3. Matrizes de confusão normalizadas para os experimentos 2, 3 e 4.

tank, utilizando como modelo base o Faster R-CNN com ResNet-50-FPN como *backbone*. Para possibilitar a comparação dos resultados, o modelo escolhido nesta pesquisa foi o YOLOv8x com otimizador SGD e lote 8. A escolha da arquitetura YOLO ao inves da Faster R-CNN, se deu devido a YOLO apresentar melhores resultados na base de dados

MBG e por reportar individualmente os resultados das classes nos testes, facilitando assim a comparação. A Tabela 9 apresenta os resultados do trabalho de [Passos et al. 2021] e os resultados desta pesquisa para a base MBG. Com base nas métricas reportadas, observa-se que, para a MBG, o modelo YOLOv8x apresentou desempenho superior na localização das classes *tire* e *water tank*, indicando maior eficácia na identificação dos casos verdadeiros com poucos erros. Além de obter resultados superiores para as classes específicas, o modelo YOLOv8x/SGD com tamanho de lote 8, é treinado para seis classes de objetos, o que torna o problema consideravelmente mais difícil do que quando se considera apenas duas classes, como em [Passos et al. 2021]. O resultado superior se deve também a uma extensa experimentação com de modelos YOLO considerando tamanho de rede, otimizador e tamanho de lote. Contudo, cabe também destacar que apesar da base de dados deste trabalho ser extraída da mesma base de dados MBG, elas não são exatamente iguais, principalmente pelo falo de [Passos et al. 2021] considerar a predição em vídeos, enquanto este trabalho considera a predição em imagens isoladas.

Tabela 9. Comparação de resultadados.

Modelo	**Otimizador**	**F1-score (*tire*)**	**F1-score(*water tank*)**
YOLOv8x	SGD	0.89	0.94
Faster R-CNN	SGD	0.65	0.77

Em uma análise geral, nota-se que a arquitetura YOLOv8 com o otimizador SGD superou consistentemente os modelos SSD e Faster R-CNN nos cenários avaliados. O aumento de dados contribuiu para elevar os valores de mAP50-95 e reduzir a ocorrência de falsos negativos. Neste contexto, a redução desse tipo de erro é particularmente importante, pois um falso negativo corresponde a um potencial foco de água parada que não foi identificado pelo sistema e, consequentemente, pode deixar de ser inspecionado pelos agentes de saúde. No entanto, a detecção de objetos pequenos em imagens capturadas a 40 metros de altitude permanece como o principal desafio. Esses resultados fornecem subsídios para o desenvolvimento de sistemas automatizados de apoio aos agentes de saúde na localização de potenciais criadouros do *Aedes aegypti* em áreas urbanas.

5. Conclusões

Este trabalho avaliou arquiteturas CNN (YOLO, SSD, Faster R-CNN) para detectar focos de água parada em imagens de drone, investigando o impacto de otimizadores e tamanhos de lote. Os resultados demonstraram a eficácia da arquitetura YOLO na detecção de potenciais focos de água parada, especialmente nas versões *Large* e *Extra Large* associadas ao otimizador SGD. No primeiro experimento, o modelo YOLOv8x com SGD e lote 8 alcançou o melhor desempenho geral, com mAP50-95 de 0.764, demonstrando a eficácia dessa combinação para a tarefa proposta. A transição para a MBG-v2 (10 classes, objetos menores a 40 m, área urbana) reduziu o mAP50-95, evidenciando maior desafio.

A aplicação de técnicas de aumento de dados às imagens que continham as classes minoritárias *bin*, *bottle*, *bucket* e *tire* mostrou-se benéfica, elevando os valores de mAP50-95. No caso do YOLOv8x, por exemplo, essa métrica passou de 0,39 para 0,4966. A análise das matrizes de confusão indicou uma redução dos falsos negativos e uma melhora no reconhecimento dessas classes, demonstrando a eficácia do aumento de dados para lidar com classes que possuem poucas amostras.

O quarto experimento, focado exclusivamente nas classes de objetos menores *bottle*, *bucket*, *plastic bag* e *tire*, evidenciou a principal limitação do estudo. A queda no valor de mAP50-95, de 0,39 para 0,2903 no YOLOv8l e para 0,1085 no YOLOv8x, demonstra que a detecção de objetos pequenos a 40 metros de altitude, em ambientes urbanos densos, permanece como o maior desafio a ser superado.

Por fim, com base nessas análises, as CNNs mostraram-se promissoras para a detecção de potenciais focos de água parada, com destaque para a arquitetura YOLO associada ao otimizador SGD. O aumento de dados e a escolha adequada dos hiperparâmetros de treinamento mostraram-se determinantes para a obtenção de um bom desempenho. Como trabalhos futuros, sugere-se avaliar versões mais recentes da família YOLO, como a YOLO26, e diferentes *backbones* para as arquiteturas Faster R-CNN e SSD. Outra frente de investigação consiste na realização de novas coletas de dados, contemplando maior diversidade de cenários, a fim de construir conjuntos mais robustos e representativos. Por fim, propõe-se o desenvolvimento de uma ferramenta integrada a drones para a detecção automatizada dos objetos de interesse. Para aproximar a avaliação das condições reais de operação, essa ferramenta deverá ser testada por meio de inferências em vídeos capturados durante os voos. Essas investigações podem contribuir para melhorar o desempenho dos modelos e ampliar a aplicabilidade prática da abordagem proposta.

Referências

(2024). Using uav images and deep learning in investigating potential breeding sites of aedes albopictus. *Acta Tropica*, 255:107234.

Carrasco-Escobar, G., Moreno, M., Fornace, K., Herrera-Varela, M., Manrique, E., and Conn, J. E. (2022). The use of drones for mosquito surveillance and control. *Parasites & Vectors*, 15(1):473.

Centers for Disease Control and Prevention (2025). Mosquitoes: About Mosquitoes. Acesso em: 21 fev. 2025.

Cunha, H. S., Sclauser, B. S., Wildemberg, P. F., Fernandes, E. A. M., dos Santos, J. A., Lage, M. d. O., Lorenz, C., Barbosa, G. L., Quintanilha, J. A., and Chiaravalloti-Neto, F. (2021). Water tank and swimming pool detection based on remote sensing and deep learning: Relationship with socioeconomic level and applications in dengue control. *PLOS ONE*, 16(12):1–24.

Jocher, G., Qiu, J., Liu, M., Lyu, S., Akyon, F. C., and Kalfaoglu, M. E. (2026). Ultralytics yolo26: Unified real-time end-to-end vision models. *arXiv preprint arXiv:2606.03748*.

Laboratório de TeleMídia - UFRJ (2026). Mosquito database. `https://www02.smt.ufrj.br/~tvdigital/database/mosquito/index.html`. Acessado em 22 jun. 2026.

Laranjeira, C., Pereira, M., Oliveira, R., Barbosa, G., Fernandes, C., Bermudi, P., Resende, E., Fernandes, E., Nogueira, K., Andrade, V., Quintanilha, J. A., dos Santos, J. A., and Chiaravalloti-Neto, F. (2024). Automatic mapping of high-risk urban areas for aedes aegypti infestation based on building facade image analysis. *PLOS Neglected Tropical Diseases*, 18(6):1–29.

Ministério da Saúde (2025). Monitoramento das Arboviroses. Acesso em: 21 fev. 2025.

Moran-Landa, D., del Rosario Damian, M., Mendoza, P. M. P., and Sotomayor-Beltran, C. (2022). A drone system with an object identification algorithm for tracking dengue disease. *International Journal of Advanced Computer Science and Applications*, 13(10).

Passos, W. L., Araujo, G. M., de Lima, A. A., Netto, S. L., and da Silva, E. A. B. (2021). Automatic detection of aedes aegypti breeding grounds based on deep networks with spatio-temporal consistency.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Roboflow (2025). Roboflow: Computer Vision Dataset and Model Management. Acesso em: 21 fev. 2025.

Roboflow Universe (2026a). mgbv2augmented dataset. `https://universe.roboflow.com/testedengue/mbgv1`. Acessado em: 23 ago. 2026.

Roboflow Universe (2026b). mgbv2augmented dataset. `https://universe.roboflow.com/testedengue/mbgv2`. Acessado em: 23 ago. 2026.

Roboflow Universe (2026c). mgbv2augmented dataset. `https://universe.roboflow.com/testedengue/mgbv2augmented`. Acessado em: 23 ago. 2026.

Roboflow Universe (2026d). mgbv2augmented dataset. `https://universe.roboflow.com/testedengue/mgbv2smallobjetcs`. Acessado em: 23 ago. 2026.

Signals, Multimedia, and Telecommunications Laboratory (COPPE/Poli/UFRJ) (2026). Mosquito breeding grounds (mbg) video database. `https://www02.smt.ufrj.br/~tvdigital/database/mosquito/index.html`. acessado em 3 jan. 2026.

Trujillano, F., Garay, G. J., Salas, H. A., Byrne, I., del Prado, M. N., Chan, K., Manrique, E., Johnson, E., Apollinaire, N., Kouakou, P. K., Oumbouke, W. A., Tiono, A. B., Guelbeogo, M. W., Lines, J., Carrasco-Escobar, G., and Fornace, K. M. (2023). Mapping malaria vector habitats in west africa: Drone imagery and deep learning analysis for targeted vector surveillance. *Remote sensing*, 15:2775 – 2775.

World Health Organization (2020). Vector-borne diseases. Acesso em: 21 fev. 2025.