Sistema Colaborativo de Apoio à Vigilância do *Aedes aegypti* com uso de YOLOv12 para Detecção em Imagens de *Smartphones*

Gustavo V. Castro[1], **Davidson M. R. Vieira**[1], **Matheus S. F. Costa**[1], **Rafael H. da R. Silva**[1], **Pedro H. T. de Souza**[1], **Felipe A. L. Soares**[1]

[1] Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais – Belo Horizonte, Brasil

academic@gvcastro.com, {davidsonmarra, matheussantosfcosta, sinps.rh, phtsouza}@gmail.com, felipesoares@pucminas.br

Abstract. *Combating the proliferation of the Aedes aegypti mosquito faces barriers such as the difficulty of conducting in-person household inspections. To mitigate this problem, this work proposes a system for detecting potential breeding sites through images captured by residents' own smartphones. This aims to provide a ground-level perspective similar to that of health agents, strengthening community proximity and engagement. The images are processed by the YOLOv12m model and the data are integrated into an interactive map for managers. The solution demonstrated viability as a complementary tool, achieving a precision of 83.93%, F1-score of 70.68%, and recall of 61.04% in identifying potential breeding sites.*

Resumo. *O combate à proliferação do mosquito Aedes aegypti enfrenta barreiras como a dificuldade de inspeção domiciliar presencial. Para mitigar esse problema, este trabalho propõe um sistema de detecção de potenciais criadouros por meio de imagens registradas pelos smartphones dos próprios moradores. Isso visa oferecer uma perspectiva terrestre semelhante à dos agentes de saúde, fortalecendo a proximidade e engajamento da comunidade. As imagens são processadas pelo modelo YOLOv12m e os dados são integrados a um mapa interativo para gestores. A solução demonstrou viabilidade como ferramenta complementar, alcançando precisão de 83,93%, F1-score de 70,68% e recall de 61,04% na identificação de potenciais criadouros.*

1. Introdução

O mosquito *Aedes aegypti* é agente transmissor da dengue, zika e chikungunya, que são arboviroses que representam um grave problema de saúde pública. Apenas em 2024, foram reportados 187.719 casos de hospitalização [DATASUS 2024] e 6.264 mortes causadas pela dengue no Brasil [Ministério da Saúde do Brasil 2025]. A proliferação dessas doenças está diretamente associada à presença de água parada, pois é este o local onde as fêmeas depositam seus ovos [Wilke et al. 2020]. Portanto, é de grande importância propor métodos eficazes de identificação dos criadouros para eliminá-los e, assim, reduzir o impacto causado pelas doenças mencionadas.

Em ambientes urbanos, a inspeção depende majoritariamente de visitas domiciliares realizadas por agentes de saúde, um processo que, além de demandar tempo e recursos

humanos significativos, apresenta limitações em sua abrangência, como a dificuldade de acesso a todas as residências e a necessidade de repetidas visitas [Zara et al. 2016]. Além disso, segundo Passos et al. 2022, mesmo com inspeções, a ausência de métodos automatizados para identificar e mapear áreas com alta concentração de focos compromete a eficiência das estratégias de combate à dengue, dificultando a alocação de recursos e a tomada de decisões por parte dos gestores de saúde pública. Fatores como a variação na qualidade das imagens capturadas, condições de iluminação e a necessidade de segmentar potenciais criadouros representam desafios técnicos adicionais.

Portanto, diante desse contexto, este trabalho tem como objetivo investigar como um modelo baseado em redes convolucionais pode detectar potenciais criadouros de mosquito e contribuir para o mapeamento de áreas de maior risco de proliferação, oferecendo suporte para ações preventivas mais eficazes[1]. O modelo foi implementado como parte de um sistema que cumpre as funções de capturar imagens do ambiente externo das residências, detectar possíveis criadouros de mosquito e disponibilizar os resultados para os agentes de saúde. O sistema tem a funcionalidade de receber as imagens registradas por moradores, que fotografam essas áreas e terrenos próximos.

O restante do artigo está organizado da seguinte forma: na Seção 2 são apresentados os trabalhos correlatos, enquanto a Seção 3 descreve a metodologia adotada para o desenvolvimento do sistema incluindo as tecnologias e parâmetros utilizados acompanhados pela justificativa da sua escolha. Além disso, nessa mesma seção são apresentados os resultados do treinamento do modelo de detecção e o fluxo de funcionamento do serviço de detecção de imagens implementado no projeto. Na Seção 4, são apresentados os resultados quantitativos do modelo de detecção e o desenvolvimento da aplicação *mobile* e portal web, respectivamente. Por fim, a Seção 5 contém as conclusões e sugestões para trabalhos futuros.

2. Trabalhos Correlatos

O uso de visão computacional aplicada ao monitoramento do mosquito *Aedes aegypti*, seja na identificação de potenciais criadouros, seja no estudo de seu comportamento, tem sido amplamente estudado nos últimos anos. Esta seção apresenta uma revisão de trabalhos correlatos, selecionados por apresentarem alguma semelhança nos objetivos, diferenças nas abordagens para solucionar o problema ou por proporem soluções próximas ao contexto deste trabalho.

No trabalho de Cunha et al. 2021, a abordagem da detecção de possíveis criadouros de mosquito tem enfoque em caixas d'água e em piscinas. Foram usadas redes neurais convolucionais (modelo Faster R-CNN com MobileNetV2) para a detecção automática dos objetos. Já no artigo de Lima et al. 2021, foi proposto um sistema de visão computacional baseado em YOLOv4 para identificar automaticamente possíveis focos do vetor da dengue em imagens aéreas capturadas por drones.

Em outro artigo semelhante, Passos et al. 2022 também utiliza drones para alimentar uma base de dados composta pelos vídeos gerados que serviram como base para o treinamento do modelo. O sistema proposto utiliza uma CNN com arquitetura Faster

[1]Códigos desenvolvidos e as bases de imagens estão disponíveis publicamente no repositório da pesquisa: https://github.com/cart-pucminas/deteccao-criadouros-mosquito.

R-CNN com backbone ResNet-50-FPN para a detecção dos objetos em cada *frame* das filmagens. Também é implementado o método *spatio-temporal consistency* (em português, consistência espaço-temporal) que registra a trajetória dos objetos através dos frames, diminuindo a quantidade de possíveis falsos positivos e negativos. Os resultados obtidos dentro da métrica F1-score foram de 65% para pneus e 77% para caixa d'água.

No artigo de Javed et al. 2023, o objetivo foi desenvolver o EggCountAI, que é um software gratuito baseado em *Deep Learning* (em português, Aprendizado Profundo) para contar automaticamente os ovos de *Aedes aegypti*. Esse sistema apresentou acurácia de 98,88% para imagens micro e 96,06% para imagens macro.

O estudo de Laranjeira et al. 2023 propõe o uso do modelo YOLOv7 para detectar e rastrear criadouros do mosquito transmissor de arboviroses em vídeos capturados por drones. A detecção é realizada quadro a quadro, seguida de uma agregação baseada em medidas de Interseção sobre União (IoU, do inglês *Intersection over Union*) e distâncias espaço-temporais para manter a consistência no rastreamento.

Por fim, Ong et al. 2024 propõe o sistema *AMOSS* (Automatic Mosquito Oviposition Study System), voltado ao estudo automatizado do comportamento de oviposição de mosquitos Aedes. O sistema utiliza um microcomputador Raspberry Pi com câmera infravermelha (NoIR) para capturar vídeos em time-lapse, e emprega o modelo SSD-MobileNetV2 para detectar automaticamente as atividades de postura de ovos, eliminando os vieses associados à presença do observador. O modelo alcançou precisão de 75,34% e recall de 72,06% na detecção das atividades de oviposição, com mAP de 70,68%.

A Tabela 1 mostra que nos trabalhos correlatos apresentados foram utilizadas três formas de adquirir as imagens: drones com câmeras acopladas, câmeras macroscópicas ou infravermelhas em ambiente controlado. Vale ressaltar que, apesar das métricas deste trabalho serem comparativamente mais baixas, o contexto em que as imagens foram registradas é mais abrangente, portanto, as imagens são mais diversas. Sabendo disso, este trabalho traz uma nova perspectiva para captura das imagens: as imagens são registradas pelos *smartphones* dos moradores, oferecendo um ponto de vista adicional, mais semelhante ao que os agentes de saúde estão habituados em suas visitas. Isso reforça o intuito de permitir que o usuário (cidadão) colabore com o monitoramento dos criadouros de mosquito. As métricas não apresentadas pelos trabalhos foram marcadas com um traço (–).

Tabela 1. Comparação entre os trabalhos correlatos.

Artigo	Precisão	Recall	F1-score	Modelo	Aquisição
Lima et al. 2021	89,73%	–	–	YOLOv4	Drone
Cunha et al. 2021	91%	–	–	Faster R-CNN	Drone
Passos et al. 2022	67%	90%	76,82%	Faster R-CNN	Drone
Javed et al. 2023	$\approx$ 97%	$\approx$ 90%	93,37%	EggCountAI	Câmera macro
Laranjeira et al. 2023	$\approx$ 85%	90%	87,43%	YOLOv7	Drone
Ong et al. 2024	75,34%	72,06%	73,66%	SSD-MobileNetV2	Câmera NoIR
Este trabalho	83,93%	61,04%	70,68%	YOLOv12	Registro por smartphone

3. Metodologia

Esta seção apresenta primeiramente as ferramentas escolhidas para o desenvolvimento do sistema bem como a justificativa para seu uso. Em seguida, para descrever o método, adotou-se uma abordagem dividida em três partes: a primeira refere-se ao processo de treinamento do modelo de detecção; a segunda apresenta as funcionalidades disponíveis para os clientes (ou seja, moradores e gestores de saúde) e a infraestrutura necessária para implantação dessas funções, incluindo o aplicativo móvel, portal web e serviços *backend* em um ambiente de nuvem. Por fim, a terceira parte ilustra o fluxo do *pipeline* de detecção de objetos além de apresentar curvas que descrevem a evolução do treinamento do modelo de detecção.

3.1. Materiais

A arquitetura convolucional escolhida para treinar o modelo foi a rede YOLO em sua décima segunda versão, que introduz uma arquitetura centrada em mecanismos de atenção e traz ganhos de desempenho em relação às versões precedentes [Tian et al. 2025]. Sua baixa latência de detecção e requisitos moderados de *hardware* foram pontos que levaram à sua escolha, considerando a infraestrutura disponível para demonstração do sistema. A saída do modelo de detecção é representada pela imagem original sobreposta por caixas retangulares que delimitam o objeto detectado, além de metadados com a lista de objetos, suas classes e a quantidade deles.

A base de dados escolhida para treinar o modelo foi a *MosquitoFusion Dataset* [Sayeedi et al. 2024]. Foram removidas imagens que não eram de interesse como picadas de mosquito e o restante foi incluído dentro de uma única classe, além de *background* (fundo ou área não marcada); as imagens dessa base têm dimensão de 640×640. Além dessa base de dados, os autores do artigo registraram 55 imagens adicionais; essas imagens foram redimensionadas para que seu menor lado contenha 640 pixels. Em relação à sua divisão, adotou-se uma proporção maior de treino (60%) dado o volume reduzido de amostras, preservando representatividade nos conjuntos de validação e teste.

A Tabela 2 também indica o uso de imagens geradas pelo *LLM* (*Large Language Model*, em português, Grande Modelo de Linguagem) da *Google* da classe de modelos *Gemini* (na versão Gemini 3 Pro) que é capaz de gerar imagens fotorrealistas. No contexto deste trabalho, foram geradas imagens de quintais e terrenos baldios contendo uma vasta gama de objetos com potencial para acúmulo de água como potes, baldes, pneus, garrafas etc. A composição detalhada das bases, discriminando as fontes originais, sintéticas e o volume final após o processo de aumento, encontra-se sumarizada na Tabela 2.

Tabela 2. Composição das bases de dados.

Dataset	Total de imagens	Treino	Validação	Testes
MosquitoFusion	200	100	60	40
Imagens dos autores	55	33	11	11
Gerado por Gemini 3 Pro	293	293	0	0
Total	**548**	**426**	**71**	**51**
Total aumentado	**974**	**852**	**71**	**51**

Em relação às imagens geradas artificialmente, cabe ressaltar que estas não foram

incluídas nos conjuntos de validação e teste do modelo. Essa decisão metodológica visa evitar que imagens sintéticas mascarem o real desempenho do modelo durante o monitoramento do treinamento e comprometam a confiabilidade da avaliação de generalização no conjunto de teste.

Para ampliar a variabilidade das amostras e fortalecer a capacidade de generalização do modelo frente a diferentes cenários, foram aplicadas técnicas de aumento de dados (*data augmentation*) sobre o conjunto de treinamento. Essa estratégia visa mitigar a escassez de dados reais e prevenir o *overfitting*. Foram realizados os seguintes aumentos sobre a base de dados: espelhamento horizontal, rotação, transformação afim, ajuste de brilho e contraste e ajuste de saturação.

Quanto às tecnologias implementadas no sistema, a aplicação móvel foi desenvolvida utilizando o *framework* multiplataforma React Native, sendo a versão atual destinada à plataforma Android. O *backend* foi desenvolvido com o *framework* FastAPI, tecnologia também utilizada pelo serviço de detecção para servir as predições do modelo. A infraestrutura necessária para hospedar e demonstrar os resultados do trabalho foi implantada na *Google Cloud Platform* (GCP), onde foi utilizado o serviço *Cloud SQL* para banco de dados PostgreSQL e *Cloud Storage* para armazenamento de objetos.

Informações adicionais sobre a divisão dos conjuntos de dados, as técnicas de *data augmentation* aplicadas e o processo de geração das imagens sintéticas, bem como instruções para reprodução do sistema desenvolvido, estão disponíveis no repositório da pesquisa referenciado na Seção 1. Além disso, em relação à implementação, é importante ressaltar que a infraestrutura para demonstração dos resultados não foi disponibilizada publicamente.

3.2. Método

A metodologia de treinamento, dividida em cinco etapas, está ilustrada na Figura 1.

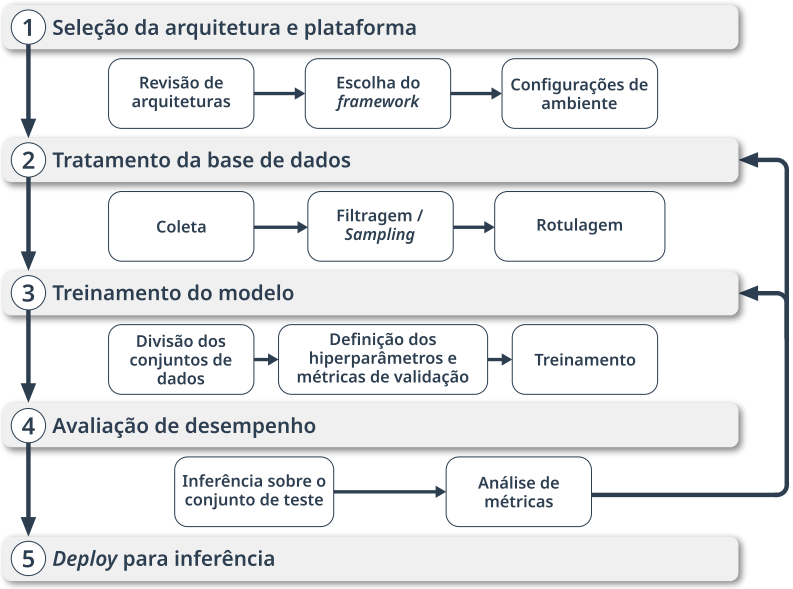

Figura 1. Metodologia de treinamento do modelo de detecção.

Sucintamente, a primeira etapa consiste no processo de seleção da arquitetura de detecção e tecnologias de treinamento levando em consideração a natureza do problema;

é na segunda etapa em que as bases de dados foram selecionadas, filtradas e rotuladas de acordo com a arquitetura YOLO, que foi a escolhida. Na terceira etapa, a base de dados foi distribuída entre os conjuntos de treinamento, validação e testes, também sendo realizado o aumento artificial de dados; em seguida, foram definidas as configurações para treinamento como o ajuste de hiperparâmetros e foi enfim realizado o treinamento do modelo de detecção.

Por fim, a partir do modelo treinado, foi realizada a detecção dos objetos sobre as imagens das bases de teste com a finalidade de avaliar seu desempenho. Conforme indicado pela seta de retorno na Figura 1, as etapas 3 e 4 se repetiram até que se alcançasse um resultado satisfatório de acordo com a análise quantitativa (baseando-se em métricas de desempenho) e qualitativa dos resultados (baseando-se em inspeção visual). Finalmente, o modelo treinado foi implementado no *pipeline* de detecção do *backend*.

O conjunto de treinamento é utilizado para que o modelo aprenda os padrões e características presentes nas imagens, ajustando seus pesos internos com base no erro obtido. O conjunto de validação é usado durante a etapa de treinamento para calcular métricas que permitem acompanhar a evolução do aprendizado além de permitir *early stopping* (conclusão antecipada do treinamento). Esse processo é fundamental para detectar problemas como *overfitting*, quando o modelo passa a memorizar os dados de treinamento ao invés de generalizar. Por fim, o conjunto de teste é utilizado apenas após o término do treinamento, com o objetivo de avaliar a detecção em dados completamente novos, simulando a aplicação real em produção.

Concomitante ao treinamento do modelo de detecção, foi desenvolvido o sistema do qual este faz parte. Conceitualmente, como ilustra a Figura 2, o sistema é dividido entre clientes (usuários do aplicativo *mobile* e do portal) e o serviço de *backend*. Os dois contextos são intermediados por uma API web (do inglês, *Application Programming Interface*) que interage com o banco de dados relacional, sistema de armazenamento de objetos e o serviço de detecção.

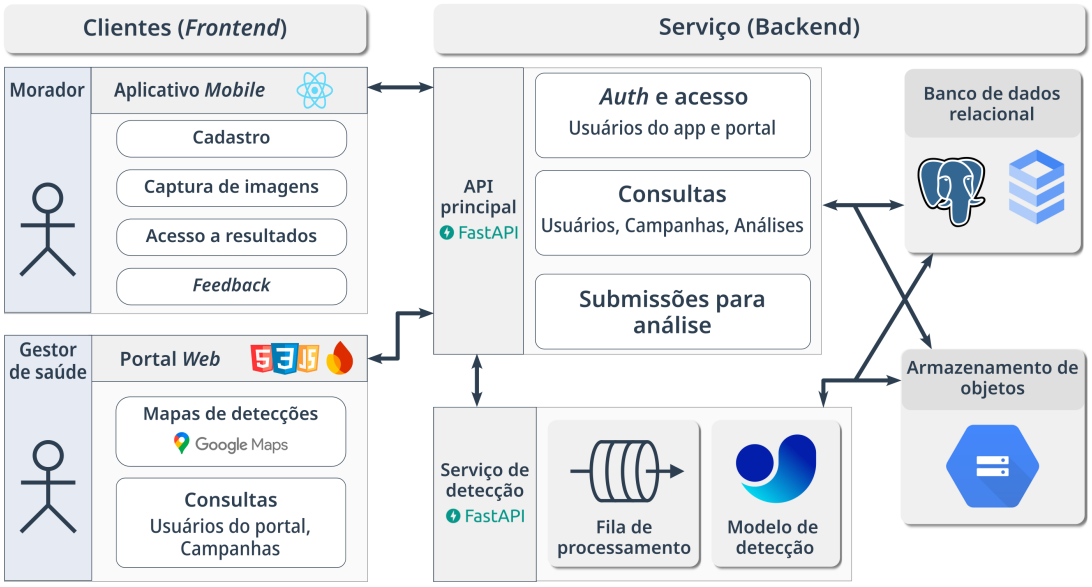

Figura 2. Diagrama de *design* do sistema.

Um dos principais componentes do *backend* é o serviço de detecção. Ele é

constituído por um conjunto de etapas de processamento de imagem (referido anteriormente como *pipeline* de detecção), que resultam na imagem original sobreposta pelas demarcações dos potenciais criadouros. As etapas do *pipeline* ocorrem na seguinte ordem: a imagem enviada pelo morador é armazenada em um diretório no sistema de armazenamento de objetos; uma tarefa é então adicionada à fila de processamento que redimensiona a imagem e aguarda disponibilidade do sistema; a rede treinada detecta os objetos na imagem; por fim, a imagem com as demarcações (também referidas como *bounding boxes*) é armazenada e os atributos da detecção, como quantidade de objetos detectados, são disponibilizados para consulta.

A aplicação móvel que foi desenvolvida permite ao usuário capturar e submeter imagens da sua residência ou de terrenos próximos. Cada imagem corresponde a uma análise e pode ou não estar associada a uma campanha sazonal criada pelos gestores de saúde do município. Busca-se detectar nessas imagens locais propensos ao acúmulo de água como vasos, calhas e recipientes diversos. Após conclusão dessa análise, o resultado é disponibilizado para o usuário de forma que ele possa verificar as detecções. Considerando isso, foi implementada a funcionalidade de *feedback*, permitindo ao morador deixar comentários sobre os resultados obtidos, potencialmente contribuindo para o aprimoramento de futuras iterações do sistema.

Já o portal web permite que gestores de saúde municipal criem e acompanhem o andamento de campanhas de eliminação dos focos de mosquito. A forma de visualização principal dos dados processados ocorre através de um mapa indicando localizações onde os potenciais criadouros foram detectados. Isso só é possível porque a aplicação móvel envia imagens georreferenciadas; na ausência desse dado, é referenciado o local da residência. Portanto, a finalidade do portal é auxiliar no aprimoramento da alocação de recursos e implementação de medidas preventivas nas áreas de risco.

3.3. Implementação

A Figura 3 apresenta o fluxo de chamadas no *pipeline* de detecção após envio da imagem pelo aplicativo.

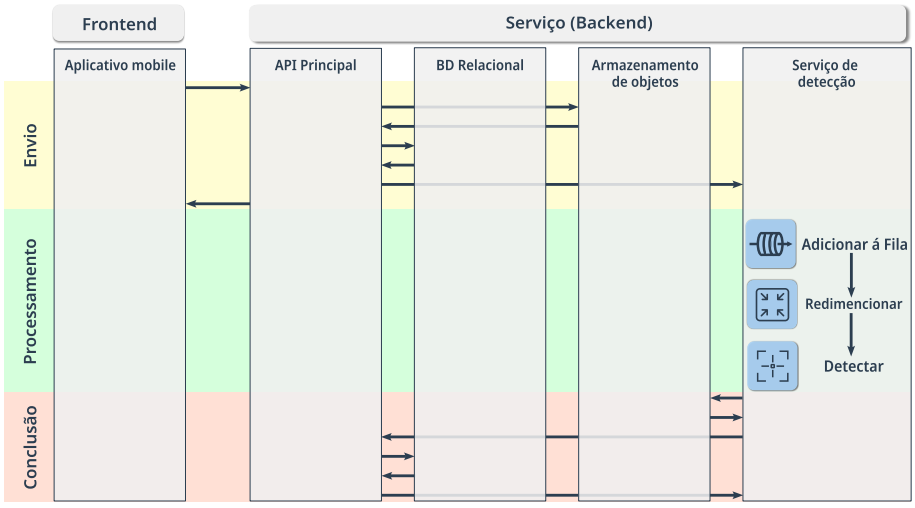

Figura 3. Diagrama do *pipeline* de processamento do sistema implementado.

Descrevendo este fluxo em mais detalhes, observa-se que, nas etapas de envio

e conclusão do processamento, são feitas requisições sequenciais tanto ao banco de dados relacional quanto ao armazenamento de objetos. Destaca-se que, neste *pipeline*, o armazenamento de objetos é destinado a armazenar as imagens originais e processadas e disponibilizá-las por meio de URLs públicas, enquanto o banco de dados relacional é responsável por agregar os metadados e registros associados às análises e ao restante da aplicação e disponibilizá-los para consulta pela API principal. O tempo total de processamento de uma imagem na implementação foi da ordem de 10 segundos.

Quanto ao treinamento da rede de detecção, os parâmetros foram definidos de forma empírica, ou seja, variando atributos como tamanho de *batch*, número de épocas e resolução das imagens, selecionou-se a combinação que obteve os melhores resultados na avaliação com a base de testes. O treinamento foi conduzido considerando um problema de classificação binária (classes: potencial criadouro e *background*), treinado por 230 épocas, com 32 imagens por *batch* e cada imagem foi redimensionada para 640×640 pixels. O *hardware* utilizado foi a GPU Nvidia A100 (80 GB) na plataforma Google Colab. Como modelo base, utilizou-se a versão YOLOv12m, composta por 292 camadas e 20.138.529 pesos e foram adotadas as configurações padrões de treinamento para parâmetros não mencionados.

A Figura 4 apresenta a evolução durante o treinamento do modelo de acordo com as métricas precisão, recall e mAP (do inglês, *Mean Average Precision*). Além disso, ela apresenta as perdas calculadas para o posicionamento das *bounding boxes* inferidas em relação à referência (*ground truth*). Essas métricas foram calculadas a partir das predições sobre o conjunto de validação. Observa-se que o treinamento é concluído quando o platô é atingido nas curvas de precisão e recall.

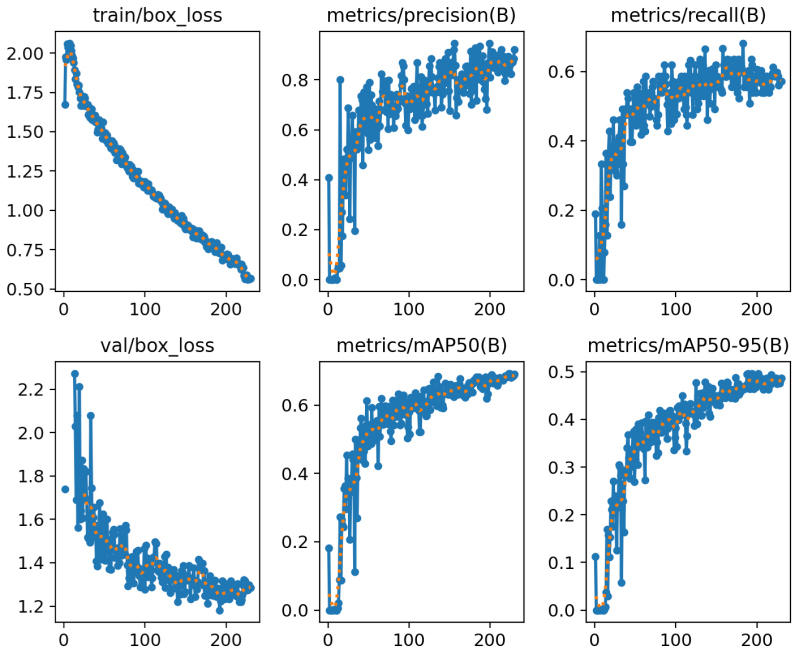

Figura 4. Evolução do treinamento do modelo baseado na rede YOLOv12m.

4. Experimentos e resultados

A Figura 5 apresenta um mosaico de 6 imagens que foram processadas pela versão final do modelo. Essas imagens pertencem ao conjunto de testes.

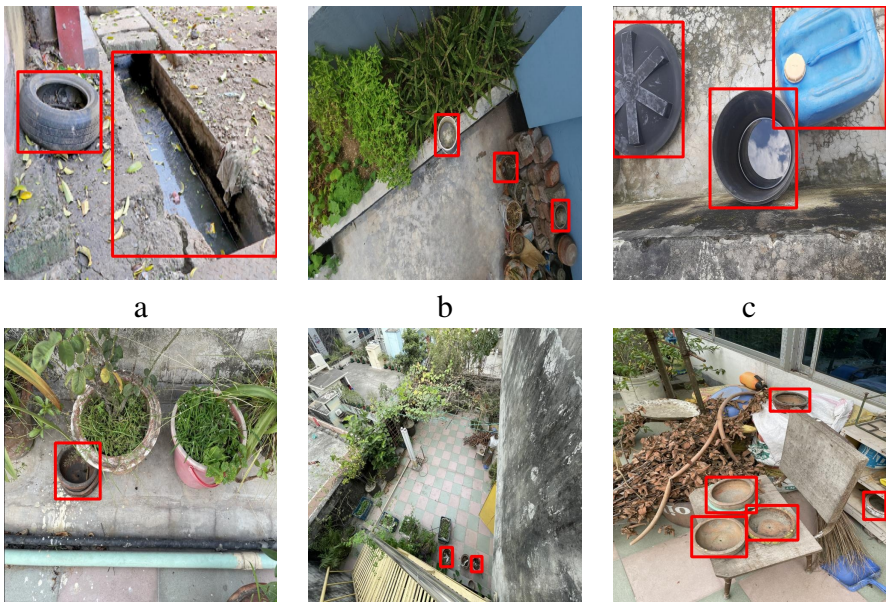

Figura 5. Resultados de detecções sobre a base de testes.

Para configurar uma detecção, foi adotado como critério considerar a confiança (conf.) $\geq 20\%$ e $IoU \geq 50\%$. No mosaico, as imagens **a**, **d** e **f** mostram imagens com detecções bem-sucedidas. Neste caso, elas identificaram corretamente valas, pneus (**a**) e compartimentos destampados (**d** e **f**). Nota-se que vasos contendo plantas não configuram detecções nestas imagens, porque a terra absorve líquidos. No entanto, as imagens **b**, **c** e **e** demonstram algumas limitações da rede nesta versão, deixando por exemplo, de detectar objetos pequenos (**e**) ou com baixo contraste (**b**), também detectando falsos positivos como tampas com concavidade para baixo e recipientes fechados (**c**).

A Tabela 3 apresenta as métricas de desempenho do modelo inferindo sobre a base de testes, considerando os limiares de detecção mencionados anteriormente.

Tabela 3. Desempenho do modelo YOLOv12m inferindo sobre a base de testes.

Métrica	Resultado
Precisão	83,93%
Recall	61,04%
F1-score	70,68%
mAP50	74,80%
mAP75	66,44%
mAP50–95	57,78%

Considerando que no conjunto de testes há 77 instâncias correspondentes a potenciais criadouros, o modelo detectou 47 dessas instâncias corretamente. Entretanto, a rede deixou de detectar 30 objetos desse conjunto (falsos negativos) e 9 detecções adicionais não correspondem a nenhum objeto rotulado originalmente (falsos positivos).

4.1. Aplicativo Móvel

No aplicativo, após o cadastro e autenticação, o usuário pode submeter imagens de duas formas distintas: fazendo *upload* de fotografias armazenadas na galeria do dispositivo

ou capturando as imagens pela câmera do *smartphone* diretamente pelo aplicativo. Essas imagens podem ser enviadas no contexto de uma campanha vigente ou podem ser submetidas de forma avulsa para análise.

Independentemente do método escolhido, após o envio da imagem, o sistema processa a fotografia através do *pipeline* de detecção descrito anteriormente. Quando o processamento é concluído, o aplicativo apresenta ao usuário os resultados da análise, exibindo a imagem com marcações visuais. Cada detecção é apresentada com um retângulo delimitador, permitindo que o morador identifique claramente quais objetos foram classificados como potenciais criadouros. A Figura 6 apresenta este procedimento.

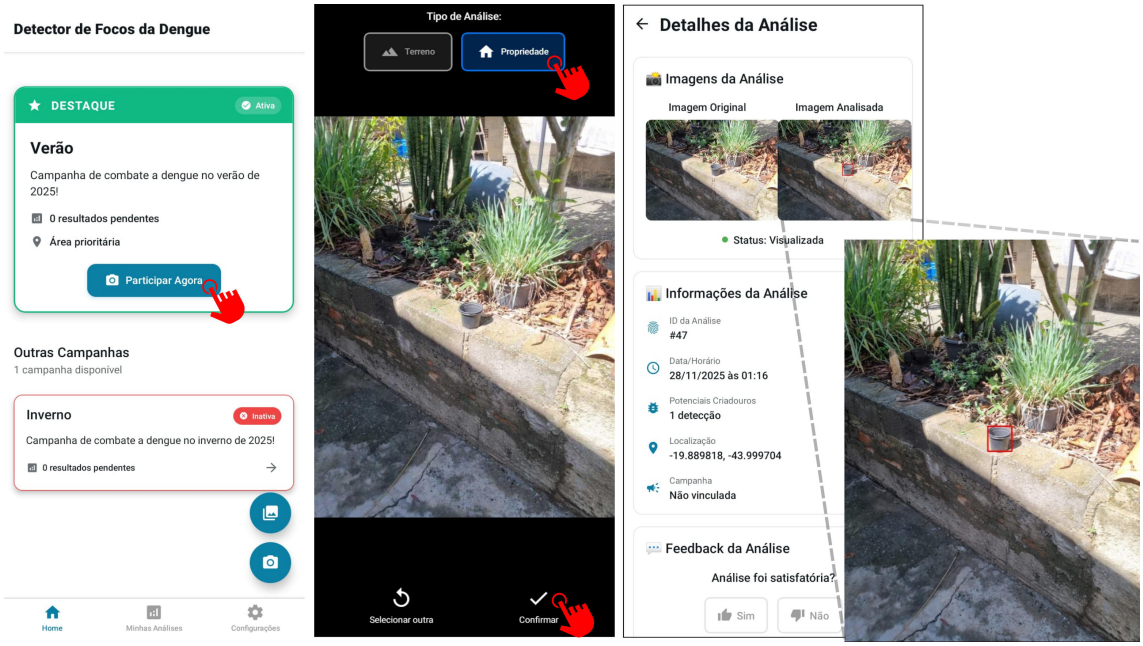

Figura 6. Fluxo de envio de imagem.

4.2. Portal Web

O portal web foi desenvolvido para atender às necessidades dos gestores de saúde municipal, fornecendo uma interface centralizada para o acompanhamento e análise dos dados coletados através do aplicativo móvel. A plataforma permite que os administradores criem e gerenciem campanhas de combate às arboviroses, definindo parâmetros como período de duração, área de abrangência geográfica e objetivos específicos de cada iniciativa. O código para reprodução tanto da aplicação *mobile* quanto do portal estão disponíveis no repositório da pesquisa apresentado na Seção 1.

A Figura 7 mostra o mapa de detecções com regiões do município de Belo Horizonte divididas em quadrantes e locais de onde moradores registraram imagens. Os locais podem se classificar como propriedades (quadrados azuis) ou terrenos (quadrados laranjas), conforme indica a legenda na parte inferior da figura. Cada marcação do mapa se expande com informações adicionais sobre as detecções naquele local ou quadrante.

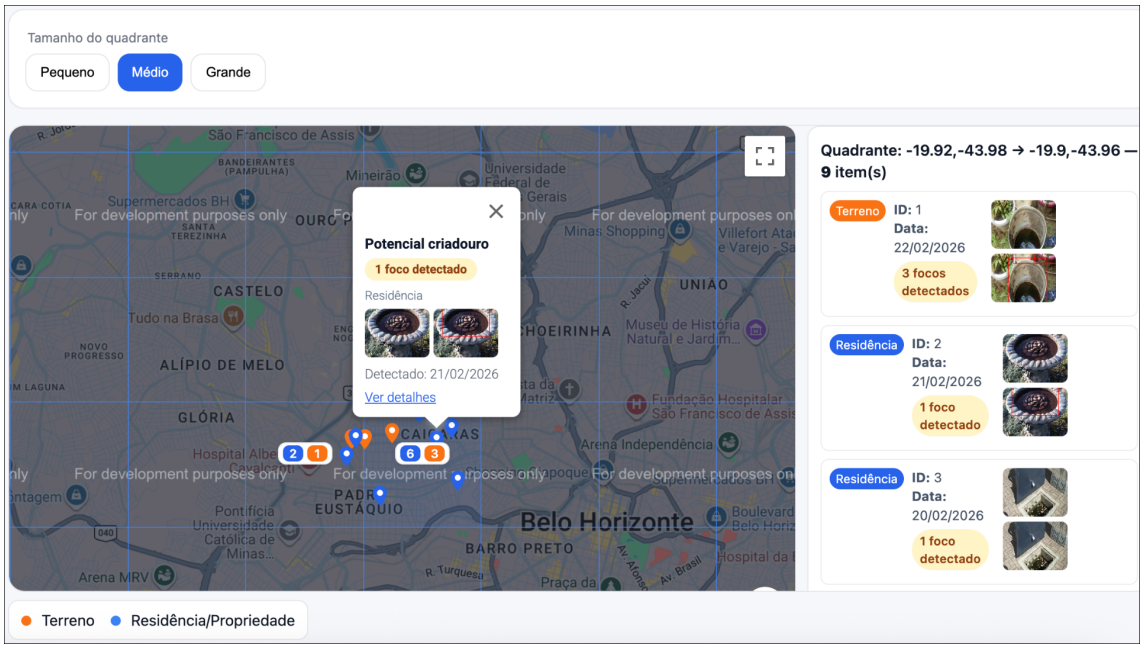

Figura 7. Mapa de detecções para a cidade de Belo Horizonte.

5. Conclusão

Este estudo demonstrou a viabilidade da visão computacional como apoio à vigilância do *Aedes aegypti*. O modelo YOLOv12 obteve resultados promissores na detecção de criadouros, com precisão de 83,93% e F1-score de 70,68%. Embora apresente um recall mais baixo (61,04%), o sistema destaca-se como uma ferramenta complementar estratégica para contornar barreiras da inspeção manual, como a escassez de agentes e a restrição de acesso a domicílios.

Além do algoritmo, a solução consolida um fluxo ágil de dados georreferenciados. O engajamento da população no envio de imagens possibilita a geração de mapas em tempo real para os gestores de saúde, o que acelera a identificação de focos e otimiza a alocação de recursos preventivos.

Como trabalhos futuros, sugere-se a adoção de uma rotulagem multiclasse que, apesar de demandar maior esforço de anotação e balanceamento, permitirá uma avaliação granular da sensibilidade e precisão do modelo para tipos específicos de recipientes. A consolidação dessa base de dados poderá estabelecer um benchmark padronizado, viabilizando a validação e a comparação direta com novas pesquisas na área. Adicionalmente, recomenda-se investigar outras arquiteturas de detecção de objetos, além da família YOLO, visando encontrar o melhor equilíbrio entre o desempenho preditivo e o custo computacional.

Agradecimentos

O presente trabalho foi realizado com o apoio da Pontifícia Universidade Católica de Minas Gerais.

Referências

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):e0258681.

DATASUS (2024). Casos prováveis por ano 1° sintoma(s) período: 2024–2024. Acesso em: 21 mar. 2025.

Javed, N., López-Denman, A. J., Paradkar, P. N., and Bhatti, A. (2023). Eggcountai: a convolutional neural network-based software for counting of aedes aegypti mosquito eggs. *Parasites & Vectors*, 16(1):341.

Laranjeira, C., Andrade, D., and dos Santos, J. A. (2023). Yolov7 for mosquito breeding grounds detection and tracking.

Lima, G., Cotrin, R., Belan, P., and Araújo, S. (2021). Sistema de visão computacional para identificação automática de potenciais focos do mosquito aedes aegypti usando drones. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, pages 93–109.

Ministério da Saúde do Brasil (2025). Monitoramento das arboviroses. Acesso em: 26 abr. 2025.

Ong, S.-Q., Isawasan, P., Nair, G., and Salleh, K. A. (2024). A novel method for studying mosquito oviposition behaviour using computer vision and deep learning algorithm. In *2024 5th International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, pages 1–6. IEEE.

Passos, W. L., Araujo, G. M., de Lima, A. A., Netto, S. L., and da Silva, E. A. (2022). Automatic detection of aedes aegypti breeding grounds based on deep networks with spatio-temporal consistency. *Computers, Environment and Urban Systems*, 93:101754.

Sayeedi, M. F. A., Hafiz, F., and Rahman, M. A. (2024). Mosquitofusion: A multiclass dataset for real-time detection of mosquitoes, swarms, and breeding sites using deep learning. In *The Second Tiny Papers Track at ICLR 2024*.

Tian, Y., Ye, Q., and Doermann, D. (2025). Yolov12: Attention-centric real-time object detectors.

Wilke, A. B. B., Vasquez, C., Carvajal, A., Medina, J., Chase, C., Cardenas, G., Mutebi, J.-P., Petrie, W. D., and Beier, J. C. (2020). Proliferation of aedes aegypti in urban environments mediated by the availability of key aquatic habitats. *Scientific Reports*, 10(1):12925.

Zara, A. L. d. S. A., Santos, S. M. d., Fernandes-Oliveira, E. S., Carvalho, R. G., and Coelho, G. E. (2016). Estratégias de controle do Aedes aegypti: uma revisão. *Epidemiologia e Serviços de Saúde*, 25(2):391–404.