Journal of Soft Computing Exploration

Homepage: htttps://shmpublisher.com/index.php/joscex

p-ISSN: 2746-7686
e-ISSN: 2746-0991

.

Comparative performance analysis of YOLOv8 small and larger for real-time website-based monitoring

Faiz Noor Adhytia[1], Raka Surya Kusuma[2], Vincentius Raditya Agung Soedomo[3], Yonathan Purbo Santosa[4]

[1,2,3,4]Department of Computer Science, Soegijapranata Catholic University, Indonesia

Article Info

Article history:

Received December 18, 2025
Revised January 8, 2026
Accepted January 28, 2026

Keywords:

Yolov8
Object detection
Real-time monitoring
Web application
Environmental informatics

ABSTRACT

River trash pollution in Indonesia demands efficient real-time monitoring solutions. By using deep learning, while YOLOv8 is a promising model for implementation, it is a family of models with each variant differ in accuracy, speed, and computational demand. Among these, YOLOv8s and YOLOv8l come out as potential variants due to the balance between speed, accuracy, and computational demand. Therefore, to address this gap, this study intends to compare the YOLOv8 small (YOLOv8s) and YOLOv8 large (YOLOv8l) variants for real-time, website-based river trash monitoring systems, aiming to identify the optimal balance between accuracy and inference speed for practical real-time deployment. A combined dataset of 66 images, consisting of 86% images from Kaggle's 2024 dataset and 14% AI-Generated images from Krea AI, was augmented using Roboflow to produce 591 annotated images. Both models were treated equally with the same dataset and method. The evaluation was conducted using Precision, Recall, mAP50, mAP50-95, Inference Speed, and Frame per Second (FPS) speed. Consequently, YOLOv8s exceeded YOLOv8l, achieving approximately 20% higher precision, 30% better detection quality, and 20% higher mAP across IoU threshold, and 150%-340% faster FPS. This finding indicates that YOLOv8s offer better accuracy and speed trade-off for real-time implementation than YOLOv8l. Moreover, successful integration to the website with real-time stream and threshold alert, confirm its feasibility for proactive waste and flood management.

Corresponding Author:

Faiz Noor Adhytia,
Department Computer Science,
Soegijapranata Catholic University,
Pawiyatan Luhur Street IV No. 1, Bendan Duwur, Gajahmungkur District, Semarang City, Central Java.
Email: 22k30006@student.unika.ac.id
https://doi.org/10.52465/joscex.v6i4.634

1. INTRODUCTION

Indonesia has 35.67% or 11.3 million ton of trash that is not well managed. Meanwhile on Indonesia's water areas, the trash pollution is also concerning, in 2022, it was recorded that there were 398 millions tons of waste polluting Indonesia's seas [1], [2]. On the other hand, low public awareness and participation worsen the issue. There only 20% of Indonesians practice proper disposal, while nearly 70% still litter on rivers and beaches [3]. Furthermore, studies show that in several provinces, up to 70% of residents dispose of waste

directly into rivers [4], [5]. This condition could lead to significant contamination of rivers, coastal zones, and groundwater, affecting ecosystem stability and public health, and increase the possibility of flood [6]. Meanwhile, the flood itself is already a pressing environmental challenge for Indonesia especially in urban regions. These floods are usually caused by solid waste and plastic debris blockage on waterways [7]. Although various initiatives have been done to improve Indonesia's waste management system, it remains faces major challenges due to weak infrastructure, low public participation, improper disposal habits, and the absence of real-time surveillance systems hinder waste mitigation efforts, particularly in riverine and coastal environments [7]. These limitations highlight the need for technology-based surveillance solutions to enhance environmental monitoring and support data-driven intervention [8].

Recent advances in Artificial Intelligence (AI) and computer vision offer new opportunities for environmental monitoring. Object detection, a core task in computer vision, has been widely explored for litter detection applications [9], [10]. Satellite-based solutions have been used to observe large-scale detection areas. However, they suffer from limitations such as cloud cover distortion, low temporal resolution, and high operational costs, so its accuration is become a problem [10]. On the other hand, ground based approaches utilizing deep learning have shown promise. Single-Shot Detector (SSD) and Faster R-CNN offer good detection capabilities but often struggle in real-time deployments. The SSD suffers from accuration degradation for small objects while Faster R-CNN, despite its accuration, is computationally intensive and operates slower [11]. Meanwhile, the You Only Look Once (YOLO) family has emerged as a preferred choice for real-time litter monitoring due to its balance of speed and accuracy [12]. Previous research has shown the YOLO capabilities for litter monitoring with river debris detection using an optimized YOLO variant [13], an improved YOLOv5 architecture for floating debris detection [14], and large-scale evaluation of YOLOv8 configurations in agricultural and natural environments [15]. As a result, compared to older YOLO versions, the YOLOv8 is able to perform with faster speed and achieve higher accuracy while still maintaining lightweight complexity [12]. However, the YOLOv8 is not a stand-alone model, but a family model comprising multiple variants with each variant being optimized for different deployment constraints. The YOLOv8n and YOLOv8s models prioritize lightweight computation and fast inference but less accuracy, while YOLOv8m, YOLOv8l, and YOLOv8x provide higher accuracy but require increasingly larger computational budgets. However, a decrease in accuracy when moving from YOLOv8l to YOLOv8x, where the performance gains become small compared to the substantial increases in GPU memory, FLOPs, and inference latency [16]. Meanwhile, comparing the YOLOv8 with more advanced successors such as YOLOv9 up to YOLOv11 integrated features such as PGI, GELAN, and C2PSA to improve small object detection performance, these newer models typically require larger computational resources and dedicated GPU acceleration [12]. Furthermore, their performance improvements are primarily observed in large-scale datasets like COCO and Object365 rather than in smaller, real-time detection systems [12].

Despite these advancements, several research gaps remain. Most of the previous studies rely on static datasets [13], [14] and without a real-time implementation [15], [17]. The promising YOLOv8 also comes with practical considerations it is not a single model, and each variant has its own advantages and limitations. Consequently, YOLOv8l may offer higher accuracy but slower speed and overfitting due to high model complexity and computational, while YOLOv8s may offer higher speed but lower accuracy due to simple model complexity and computational. These considerations highlight the importance of evaluating which variant is more appropriate for resource-constrained, continuous real-time monitoring systems.

Therefore, to address this gap, this study intends to compare the YOLOv8 small (YOLOv8s) and YOLOv8 large (YOLOv8l) variants for real-time, website-based river trash monitoring systems, aiming to identify the optimal balance between accuracy and speed for practical real-time deployment. The best performing model is then implemented into a Flask-based website for continuous river trash detection. This study contributes to applied environmental informatics by presenting a scalable and efficient detection framework that supports proactive waste management and flood prevention.

2. METHOD

The method for carrying out processing in this study is explained in this session. From dataset collection, a series of image processing stages to cleaning data from things that interfere with the data processing process. The stages of this study are visualized in Figure 1. Both YOLOv8s and YOLOv8l were trained and evaluated using identical dataset, preprocessing steps, and hardware settings to ensure fair comparison of accuracy and speed.

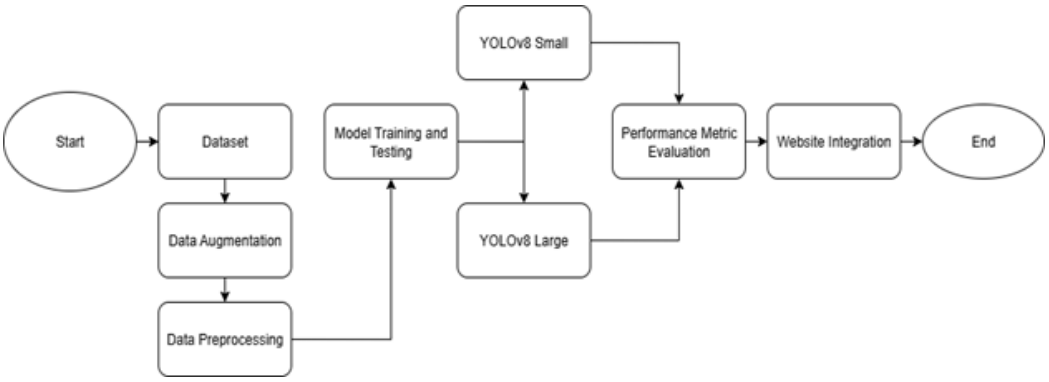


Figure 1. Proposed method's steps

Dataset

This study utilized two types of data, synthetic AI-generated images from Krea AI and Kaggle's 2024 Data. The advantage by including the synthetic dataset is to address data scarcity and enhancing dataset diversity [18]. Environmental monitoring often suffers from limited availability of labeled real-world images due to the difficulty of cost and data collection in dynamic and complex natural environments. AI-generated image datasets able to synthetically create diverse robustness to address these difficulties [19]. Compared to using publicly available datasets such as those from *Kaggle*, existing datasets are fixed in size and diversity, limited by the original data collection scope. Existing dataset may not cover all relevant trash types or environmental variations [20]. Dataset visualization is presented on Figure 2, which also shows the data label.

Figure 2. Labeled dataset visualization

Data Preprocessing

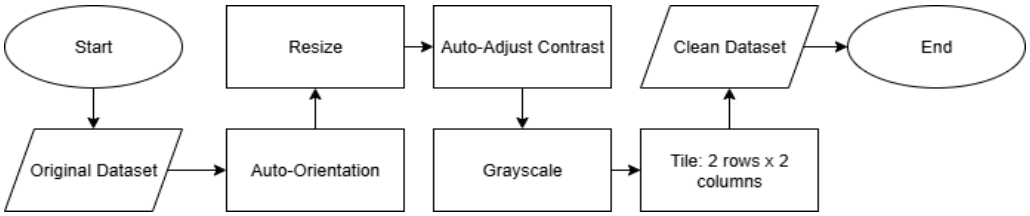


Figure 3. Image preprocessing stage

As shown in Figure 3, images are resized to a standard dimension of 640 x 640 pixels. Auto-Orientation, grayscale conversion, auto-adjusting contrast, and 2 rows x 2 columns for tile were applied to keep the data match the intended orientation, ensuring consistent display across different platforms and tools, prevents annotation errors, and facilitates better integration with real-world data [21]. The collected dataset is then uploaded to *Roboflow* platform for image annotation using a bounding box so that the image has attributes for learning YOLO algorithm. The type of labeling used in this study is based on categories, having two classes: water areas and trashes, these labels were used during the annotation process to distinguish between background water regions and waste objects present in each frame. An example of data is presented in figure 2. further refinement of labels was conducted during model development. The generalized trash class was later expanded into more specific subcategories, such as "sampah_botol", "sampah_plastic", and "sampah_kecil", to enhance model performance in distinguishing waste types. This reclassification was applied during the training stage by manually modifying class definitions and the corresponding YOLO label files. The total data is 57 images from publicly available dataset and 9 from AI-generated images which is 66 in total.

Data Augmentation

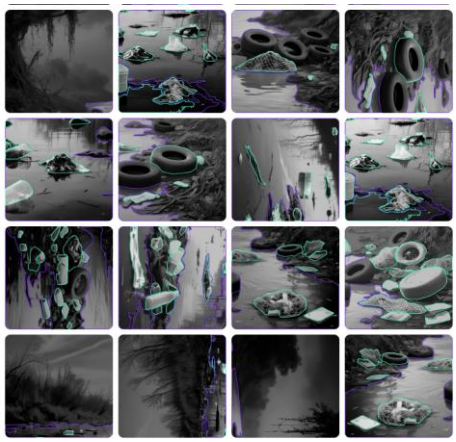


Figure 4. Dataset example after augmentation

Data augmentation was applied due to lack of sufficient training data or an imbalanced class distribution within the dataset, the augmented data example is shown in Figure 4. In this study the type of data augmentation that were used is flipping the image to horizontal and vertical, cropping image with 29% maximum zoom, ±15° horizontal, ±15° vertical shearing, grayscaling on 20% of images, between +24% and -24% on brightness adjustment, and blur the image up to 4.6px. This treatment, based on affine image transformations and color modifications, are fast and easy to implement and have been proven effective for increasing the size and diversity of training datasets [22]. The total dataset after augmentation is increased from 66 to 591 images. The dataset then split into train:valid:test (0.85 : 0.10 : 0.5).

Model Training and Testing

This study utilizes two YOLOv8 variants which are YOLOv8s and YOLOv8l, with identical training and testing configurations. The labeled and augmented dataset was then used to train the YOLOv8 model using 100 epochs, with input images resized to 800 x 800 pixels. This resolution improves detection performance by preserving spatial detail, especially for small objects [23]. Training was performed using AdamW optimizer, batch size of 16, and Automatic Mixed Precision (AMP). Better performance on metrics can be achieved by using AdamW optimizer with YOLO architecture [24], while AMP can significantly speed up the training process of deep neural networks, including convolutional neural networks used in object detection [25]. The Precision, Recall, and mean Average Precision (mAP) were then conducted to evaluate the models.

YOLOv8-Small

The YOLOv8s is a light and efficient object detection model designed to balance accuracy and inference speed [17]. The YOLOv8s employ advanced architectural components. The CSPDarknet backbone and Path Aggregation Network (PANet) were used to efficiently extract multiscale features for detecting small object. This architecture allows waste to be detected efficiently on river environments [26]. The model architecture of YOLOv8s is shown in Figure 5:

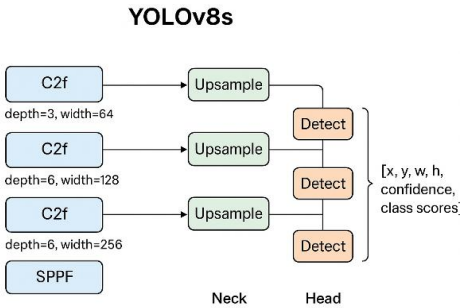


Figure 5. YOLOv8 small architecture

1) Backbone (feature extraction):
   I. Conv → C2f (depth = 3, width=64)
   II. Conv → C2f (depth = 3, width=128)
   III. Conv → C2f (depth = 3, width=256)
   IV. Conv → SPPF (Spatial Pyramid Pooling Fast)
2) Neck (feature aggregation):
   I. Multi scale feature fusion using a combination of PANet with upsampling and concatenation.
   II. Applied C2f layers to enhance semantic and spatial information accross scales
3) Head (detection and prediction):
   I. Detect layers at 3 scales (P3, P4, P5).
   II. Anchor free predictions: [x, y, w, h, confidence, class scores].

Fast inferences (Approximately 33.0 milliseconds) make the YOLOv8s faster than larger variants while maintaining competitive accuracy, this happened due to its 9 million parameter [15]. Furthermore, YOLOv8s has respectable accuracy, despite it falling short of superior accuracy and robustness, particularly in complex scenarios [27]. Therefore, YOLOv8s offers a balance between speed, accuracy and low computational demand, making it well suited for real-time application.

YOLOv8-Large

High accuracy for object detection by leveraging deeper and wider network architecture with approximately 55 million parameters, is the definition of YOLOv8l, complex and powerful variants of YOLOv8 series [15], [16]. Furthemore, it is shown to provide improved detection precision and robustness, which is particularly beneficial for complex and cluttered aquatic environments, where debris is found to vary greatly in size and appearance. Despite its size and high computational demand, result in slower inferences times (53.6 milliseconds), finer spatial details can be captured and false negatives can be reduced by YOLOv8l, which is crucial for comprehensive environmental monitoring. This architecture is integrated with advanced features such as the CSPDarknet backbone, Path Aggregation Network (PANet), and anchorless bounding box prediction, enabling effective multi-scale feature extraction and accurate object localization [15]. The model architecture breakdown of YOLOv8l is shown in Figure 6:

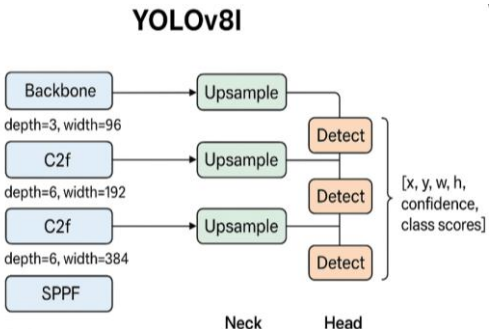


Figure 6. YOLOv8 large architecture

1) Backbone
   I. Stem Conv (input → 128 channels)
   II. Conv → C2f (depth = 3, width = 128)
   III. Conv → C2f (depth = 6, width = 256)
   IV. Conv → C2f (depth = 6, width = 512)
   V. Conv → C2f (depth = 3, width = 512)
   VI. SPPF (Spatial Pyramid Pooling Fast)
2) Neck
   I. Upsample + C2f Layers combining features from different backbones stages
   II. PAN-FPN (Path Aggregation + Feature Pyramid Network) used for multiscale feature fusion
3) Head
   I. Detect layers at 3 scale (P3, P4, P5) → handles small, medium, and large objects
   II. Anchor free predictions:
      a. Outputs: [x, y, w, h, confidence, class scores]
      b. Directly predicts object bounding boxes without anchor boxes

This model has been shown to have superior precision and recall, with subtle and hidden facial features being handled more effectively due to its deeper and broader architecture. However, this increased accuracy is offset

by slower inference times, higher hardware requirements, and bigger dataset than YOLOv8s so not all deployment scenarios are suitable for this model [28].

Evaluation Metrics

Evaluation metrics were regarded as key tools for the evaluation of accuracy and efficiency of object detection models. Such insights were crucial for the evaluation and enhancement of the model's performance [29]. The evaluation was conducted using multiple performance metrics including training loss, validation loss, Precision, Recall, mean Average Precision (mAP50), and mean Average Precision at IoU 0.5-0.95 (mAP50-95), these metrics are directly tied to the same outcomes that determine accuracy for object detection model since TN cannot be defined in object detection tasks due to the effectively infinite number of background locations where no object is present [30]. The classification result is defined using True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). Precision, recall, and F1-Score are described in Equation (1)-(4):

$$Precision\ (P) = \frac{TP}{TP + FP} \tag{1}$$

$$Recall\ (R) = \frac{TP}{TP + FN} \tag{2}$$

$$AP_{0.50} = \sum_{n=1}^{N} (R_n - R_{n-1})\ P_n \tag{3}$$

$$mAP_{50-95} = \frac{1}{10} \sum_{t=0.50}^{0.95} AP_t \tag{4}$$

The evaluation also considers the model's computational efficiency. For real-time implementation, the systems must not only be accurate but also maintain sufficient processing speed. While inference speed represent the raw processing speed of single frame, the Frame per Second (FPS) emphasize how many frames the system can handle continously, making it more reliable indicator for real-time deployment [31], which is defined in Equation (5):

$$FPS = \frac{1000}{t_{\text{inference (ms)}}} \tag{5}$$

Meanwhile the YOLO loss function combines localization, confidence, and classification error as given in Equation (6) and the expanded version of it is on the Equation (7):

$$L_{YOLOv8} = L_{box} + L_{cls} + L_{dfl} \tag{6}$$

a) L_{box} measures how close the predicted bounding box is to the ground-truth box using *CIoU.*
b) L_{cls} evaluates the accuracy of object classification using Binary Cross-Entropy.
c) L_{dfl} refines bounding box prediction by modeling coordinate distributions with Distribution Focal Loss.

Expanded:

$$L_{YOLOv8} = Localization\ \left(1 - CIoU\left(b, \hat{b}\right)\right) + Classification\ BCE(p_c, \hat{p_c}) + Distribution\ Focal\ Loss\ DFL\left(d, \hat{d}\right) \tag{7}$$

This formulation directly corresponds to the training metrics: *box_loss, cls_loss, and dfl_loss* observed during model evaluation.

Website Based Integration

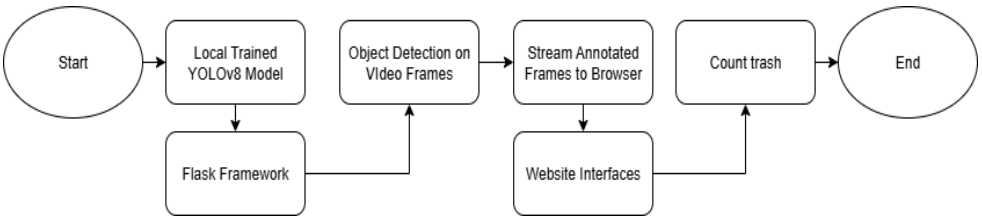


Figure 7. Website based integration diagram

As shown in the Figure 7, the best trained YOLOv8 model then get an integration into website-based application, to enable real-time monitoring with user interface by using Flask framework. Flask offers a straightforward and comprehensible development process, enabling developers to tailor endpoints, routes, threshold, and streaming logic. Compared to other frameworks such as Dash or CherryPy, Flask allows smooth integration to YOLOv8 API, OpenCV Video Pipelines, and dynamic features like notification threshold [32]. This allowed the model to perform object detection on both pre-recorded videos and live camera streams through a browser interface. The model inference code leverages OpenCV for video and image handling, while predictions are made using the Ultralytics YOLOv8 API. Detected objects are identified by class labels and bounding boxes are drawn on the video frames. For the web interface, the Flask application streams annotated video frames to the front-end via the /video_feed endpoint using a multipart JPEG stream (multipart/x-mixed-replace).

A threshold ≥25 cumulative detections is set as the threshold or a bin approaching 80-85% capacity. This approach is adopted with reference to the concept of an IoT based smart waste management system [33]. Furthermore, the detected object then classifies into specific classes (e.g., “bottle_waste”, “plastic_waste”, and “small_waste”) that are tracked by the system. A configurable threshold-based notification is implemented on /trash_detection endpoint, when the total detected object reaches the predetermined threshold, alert will be sent to encourage immediate cleanup. This integration is demonstrated as an implementation of a deep learning model for real-time inference in a lightweight and interactive web environment.

3. RESULTS AND DISCUSSIONS

Result

The study compares YOLOv8s and YOLOv8l models for real-time river trash detection using datasets from Kaggle 2024 and AI-generated images from Krea AI. The findings show how each model performs under varying image conditions and highlight the trade-off between accuracy and inference speed. All speed measurement including FPS testing on website and inference speed were conducted on a system powered by an AMD Ryzen™ 7 5800U processor with integrated AMD Radeon™ Graphics. Furthermore, the integration of the optimal model into a Flask-based web system with a threshold alert feature demonstrates its practical value for proactive waste management and flood prevention.

Accuration Evaluation Metric of YOLOv8 Small

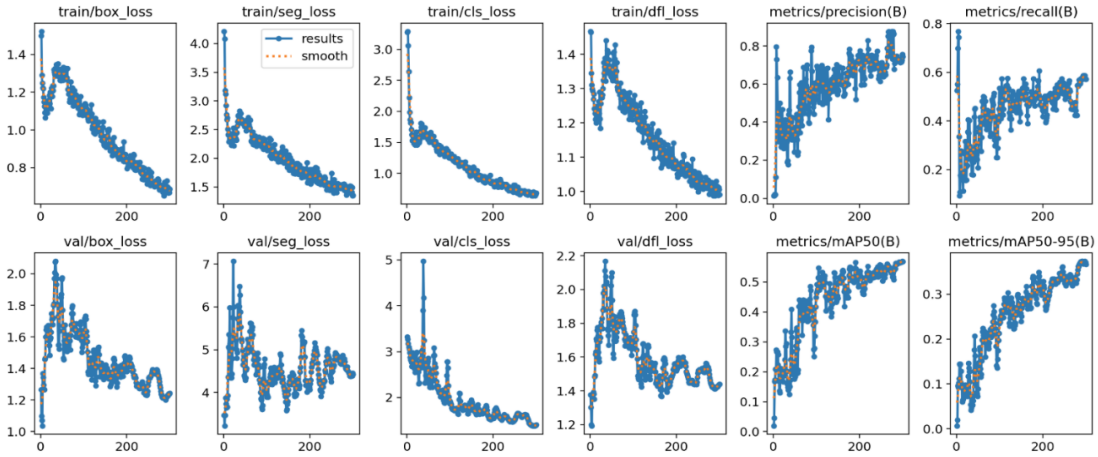


Figure 8. YOLOv8s performance metrics results

YOLOv8s model shows a stable convergence throughout the training process as shown on Figure 8, the continuous decrease in training loss section as well as occasional spikes indicating stable convergence interrupted by transient instabilities [34]. Precision scored above 0.8, recall was achieved within the range of 0.0-0.8, mAP50 scored in range of 0.0-0.5, and mAP50-95 achieved within the range up to more than 0.3, indicating strong accuration with efficient optimizer yet the model still misses objects intermittently.

Accuration Evaluation Metric of YOLOv8 Large

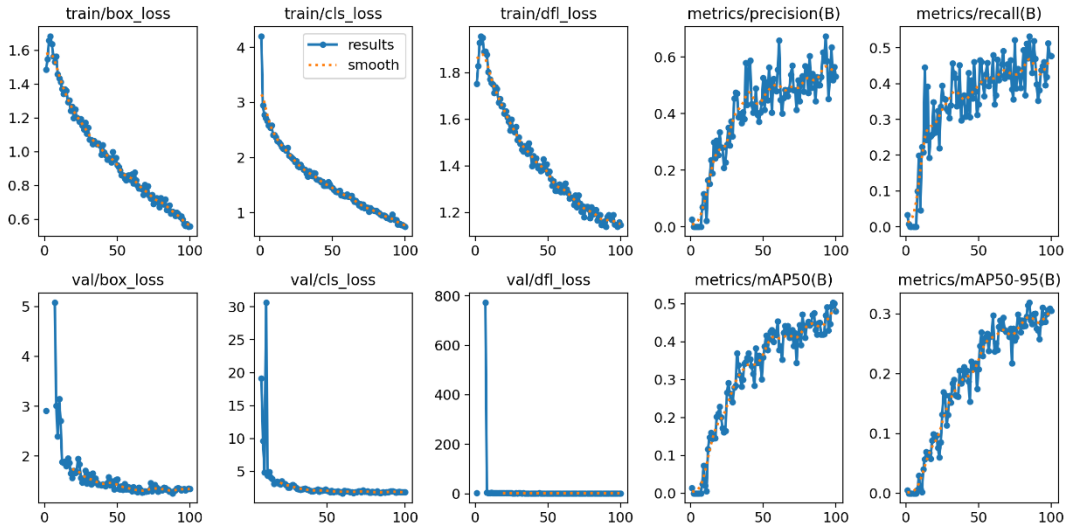


Figure 9. YOLOv8l performance metrics results

The YOLOv8l model shows a fluctuating convergences, precision scored above 0.6, recall was achieved within the range of 0.0-0.5, mAP50 scored in range of 0.0-0.5, and mAP50-95 achieved within the range up to more than 0.3, indicating high precision with the model is learning reliably and being careful, as shown on Figure 9.

Speed Comparison of YOLOv8s and YOLOv8l

(a) FPS Speed of YOLOv8s (b) FPS Speed of YOLOv8l

Figure 10. Speed comparison on FPS (a) YOLOv8s, (b) YOLOv8l

As shown in Figure 10(a), the YOLOv8s achieved higher speed compared to the YOLOv8l on Figure 10(b). YOLOv8s achieved higher FPS within range of 16-22 fps, with inference in range from 31.9-33.0 ms, despite its slower inference, the higher FPS indicating a model practical capability for real-time implementation [31]. On the other hand, the YOLOv8l achieved an FPS in range of 5.6-6.2 fps, and inference in range from 2.4-7.3 ms, despite its fast inference, the low FPS indicating less suitable application for real-time deployment.

Comparative Analysis of YOLOv8s and YOLOv8l

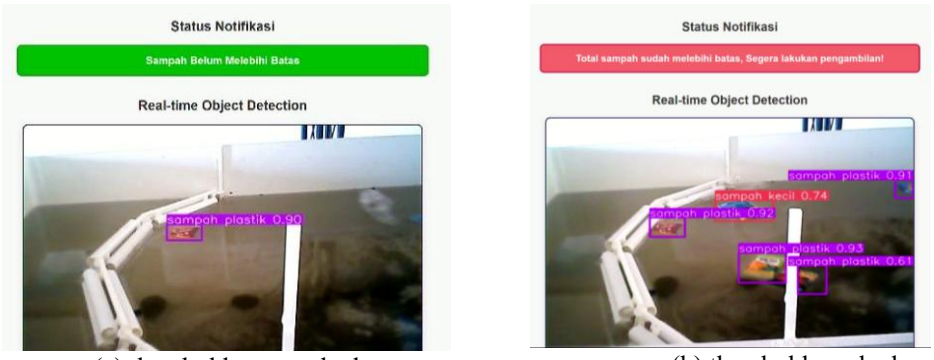
The speed and accuracy trade-off comparison between the two models is evaluated to determine the most suitable model for real-time website-based floating trash detection system.

Table 1. Performance comparison of YOLOv8s and YOLOv8l after website integration

Metric	YOLOv8s (Small)	YOLOv8l (Large)
Train/Val *box_loss*	1.4±0.5 – 0.8±0.5 / 1.0±0.5 – 1.2±0.5	1.4±0.5 – 0.6 / 5 – 2.5±0.5
Train/Val *cls_loss*	3.0±0.5 – 1.0±0.5 / 5 – 2.0±0.5	4±0.5 – 1±0.5 / 10 - 5±0.5
Precision	Above 0.8	Above 0.6
Recall	Approx 0.7	Approx 0.5
mAP50	Approx 0.8	Approx 0.5
mAP50-95	Approx 0.5	Approx 0.3
Inferences	31.8 – 32 ms	2.4 - 7.3 ms
Frame per Second	16 - 22 fps	5 – 6 fps

As shown in Table 1, overall, YOLOv8s is outperforming the YOLOv8l. Training and validation on *box_loss* component, which is responsible for how well the model predicts the correct class for each detected object, on YOLOv8s were started high, approximately on 1.4 and decrease to 0.8, this *box_loss* indicating that YOLOv8s done a stable learning of bounding boxes. Meanwhile, the YOLOv8l the training decreased nicely from approximately 1.4 to 0.6 which is better than YOLOv8s, but the validation despite its impairment the validation were high and fluctuating from 5 to approximately 2.5, indicating unstable convergence or complex model overfitting. On the *cls_loss* component, which is responsible for how accurately the model predicts object location and size, the impairment on YOLOv8s both training and validation *cls_loss* component shows reasonable classification learning, while the YOLOv8l indicates a slower convergence. The accuracy on model represented by Precision, Recall, mAP50, and mAP50-95, showing that YOLOv8s correctly identify object 20% more precisely than YOLOv8l, 30% better detection quality than YOLOv8l, 20% better performance over stricter IoU thresholds than YOLOv8l. For real-time deployment, Precision and mAP metrics ensure reliable detection. In terms of speed, YOLOv8s surpassed the YOLOv8l on real-time implementation, according to FPS measurement. While YOLOv8l achieved a minimum 77% faster inferences and 92% faster inference at maximum, however on FPS speed comparison, YOLOv8s achieved 150% faster FPS on minimum and 340% faster FPS at maximum compared to YOLOv8l.

Model Implementation



(a) threshold not reached (b) threshold reached

Figure 11. Website implementation (a) threshold not reached, (b) threshold reached

The YOLOv8s model was then integrated to the website-based system developed by using a Flask framework to enable real-time river floating trash detection with live visualization through the browser interface, as shown in Figure 11. The test was conducted in a controlled environment using a 720p webcam extension camera to capture livestream video. The system performance goes smoothly with an average processing speed of 16-22 fps. The threshold alert notification is shown on Figure 11(b) once the number of floating trash exceeds 25, while on Figure 11(a) is when the threshold is not reached yet.

4. CONCLUSION

Stable convergence, precise detection accuracy, with faster FPS speed was achieved by the YOLOv8s model, confirming its reliable capability for real-time deployment. In contrast, Despite its good Precision and faster inference, YOLOv8l shows an unstable validation losses especially on *cls_loss* and mAP, indicating less accuracy than YOLOv8s, likely due to model complexity relative to the limited dataset size. Based on comparative evaluation, the YOLOv8s provides better trade-off between accuracy and speed capability for

real-time web-based applications where live decision making is important since the data stream continuously. YOLOv8s were also require lower computational demand than YOLOv8l, making it a better suit for wider practical feasibility without worrying about computational demand that can cost higher price on device. Furthermore, the integration into a Flask-based website demonstrated practical feasibility for real-time monitoring, with a threshold alert system (≥ 25 detections) enabling early response for better waste management system.

REFERENCES

[1] BRIN, "11.3 Juta Ton Sampah di Indonesia Tidak Terkelola dengan Baik." 2025.
[2] Databoks Katadata, "Data Sampah Laut di Indonesia 2018--2022." 2025.
[3] M. Rahman, A. Bakti, and T. Hemansyah, "Communication about Waste Intervention Perspective Study on the Pandawara Community," *J. Dakwah dan Komun.*, vol. 9, 2024.
[4] L. Salsabila, K. T. Lodan, and E. Khairina, "Public Engagement Impact on Sustainable Waste Management in Indonesia," *JAP*, vol. 13, no. 2, pp. 158–178, 2023.
[5] P. Suwarno and N. Nurhayati, "Traditional Views and Attitude Toward Waste and Rivers in Indonesia," in *E3S Web of Conferences*, 2021, vol. 317, p. 1002.
[6] S. Zhao, H. Zhou, and H. Yang, "Smart Monitoring Method for Land-Based Sources of Marine Outfalls Based on an Improved YOLOv8 Model," *Water*, vol. 16, no. 22, p. 3285, 2024.
[7] Y. R. Marbun, Artiawati, and R. Azaria, "Pro-Environmental Behavior among Urban Millennial Workers," *JEHCP*, pp. 214–233, 2025.
[8] G. E. Bautista, P. A. D. Gargar, and R. G. Garcia, "Real-Time Object Detection in Tap Water Utilizing YOLOv8," in *IEEE Eurasia Conference on IoT, Communication and Engineering*, 2025.
[9] H. Agustina, H. Herdiansyah, and H. Adinegoro, "Analysis of Waste Management Processes Based on Peer Interaction," *JPPIPA*, vol. 9, no. 10, pp. 8963–8973, 2023.
[10] C. et al. Kruse, "Satellite monitoring of terrestrial plastic waste," *PLoS One*, vol. 18, no. 1, p. e0278997, 2023.
[11] D. D. Aboyomi and C. Daniel, "A Comparative Analysis of Modern Object Detection Algorithms," *ITEJ*, vol. 8, no. 2, pp. 96–106, 2023.
[12] M. L. Ali and Z. Zhang, "The YOLO Framework: A Comprehensive Review," *Computers*, vol. 13, no. 12, p. 336, 2024.
[13] N. A. Zailan, M. M. Azizan, K. Hasikin, A. S. M. Khairuddin, and U. Khairuddin, "An automated solid waste detection using the optimized YOLO model," *Front. Public Heal.*, vol. 10, p. 907280, 2022.
[14] X. et al. Yang, "Detection of River Floating Garbage Based on Improved YOLOv5," *Mathematics*, vol. 10, no. 22, p. 4366, 2022.
[15] A.-R. A. Gamani, I. Arhin, and A. K. Asamoah, "Performance Evaluation of YOLOv8 Model Configurations." 2024.
[16] M. Yaseen, "What is YOLOv8." 2024.
[17] N. A. Megantara and E. Utami, "Object Detection using YOLOv8: A Systematic Review," *SISTEMASI*, vol. 14, no. 3, p. 1186, 2025.
[18] H. B. et al. Abdalla, "The Future of Artificial Intelligence in the Face of Data Scarcity," *Tech Sci. Press*, 2025.
[19] K. Man and J. Chahl, "A Review of Synthetic Image Data and Its Use in Computer Vision," *J. Imaging*, vol. 8, no. 11, p. 310, 2022.
[20] J. Song, H. Chen, and N. Yokoya, "SyntheWorld: A Large-Scale Synthetic Dataset." 2023.
[21] N. M. Alahdal, F. Abukhodair, L. H. Meftah, and A. Cherif, "Real-time Object Detection in Autonomous Vehicles with YOLO," *Procedia Comput. Sci.*, vol. 246, pp. 2792–2801, 2024.
[22] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 2022.
[23] O. G. Ajayi, P. O. Ibrahim, and O. S. Adegboyega, "Effect of Hyperparameter Tuning on YOLOv8," *Appl. Sci.*, vol. 14, no. 13, p. 5708, 2024.
[24] A. M. Moraes, L. F. Pugliese, R. F. D. Santos, G. B. Vitor, R. A. D. S. Braga, and F. R. D. Silva, "Effectiveness of YOLO Architectures in Tree Detection," *Standards*, vol. 5, no. 1, p. 9, 2025.
[25] S. Nokhwal, P. Chilakalapudi, P. Donekal, S. Nokhwal, S. Pahune, and A. Chaudhary, "Accelerating Neural Network Training." 2023.
[26] J. Di, K. Xi, and Y. Yang, "An enhanced YOLOv8 model for accurate detection of solid floating waste," *Sci. Rep.*, vol. 15, p. 25015, 2025.
[27] E. Panja, H. Hendry, and C. Dewi, "YOLOv8 Analysis for Vehicle Classification," *SJI*, vol. 11, no. 1, pp. 127–138, 2024.
[28] D. Tresnawati, S. Nurhidayanti, and N. Lestari, "Comparison of YOLOv8 Series Performance in Student Facial Expression Detection," *JOIN*, vol. 10, no. 1, pp. 93–104, 2025.
[29] Ultralytics, "YOLO Performance Metrics." 2025.
[30] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Sci. Rep.*, vol. 14, p. 6086, 2024.
[31] S. Oh, Y. Kwon, and J. Lee, "Optimizing Real-Time Object Detection in a Multi-Neural Processing Unit System," *Sensors*, vol. 25, no. 5, p. 1376, 2025.
[32] W. O. Braganca and I. E. Kho, "Comparative Analysis of Python Microframeworks." 2023.
[33] T. Kanjanapruthipong, S. Ampawong, U. Thaenkham, K. Tuentam, and D. Watthanakulpanich, "Survival of immature pre-adult Gnathostoma spinigerum in humans," *PLoS One*, vol. 17, no. 3, p. e0264766, 2022.
[34] D. Jepisah, "YOLOv12 Model Optimization for Monitoring Occupational Health and Safety," *J. Appl. Data Sci.*, vol. 6, no. 4, pp. 2666–2681, 2025.