LitterLens: An Automated Identification and Quantification of Waterways Macrolitter via Machine Learning

A Case Study Proposal Presented to the

College of Computer Studies of

Pamantasan ng Lungsod ng Pasig

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Computer Science

by:

Alentajan, Jervie D.

Florida, Emanuel C.

Serapion, Pauline L.

May 2025

Introduction

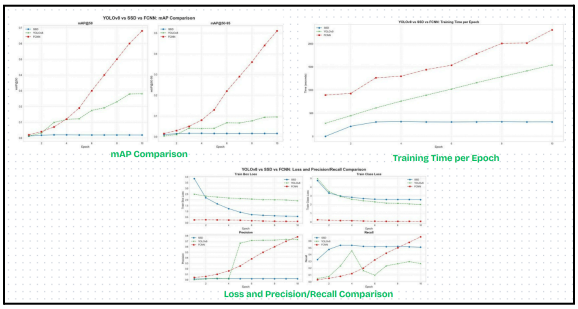
Water litter has become a rising global concern, threatening ecosystems and biodiversity. Common water pollutants are based on land-based and ocean-based sources. These sources are namely the improper disposal of wastes, runoff from agricultural and industrial areas, and illegal dumping. This issue narrows down to the densely populated cities of Metro Manila. Pasig City, known as the Green City, is one of the greatest contributors of waste pollution. This also goes down to the arteries of the communities like canals, dikes, and creeks. This research develops a web-based system using advanced machine learning models to quantify and identify surface water litter in Pasig City's waterways. The paper analyzes these models.

Purpose

This study develops an automated system, using machine learning, to identify and quantify surface water macrolitter. It implements and compares three models—YOLOv8, SSD, and Faster R-CNN—for litter detection. The objectives of the study are as follows:

I. To evaluate the performance of selected algorithms based on specific criteria such as time complexity, model performance, and efficiency.
II. To compare and contrast multiple algorithms to determine their strengths and weaknesses in solving a particular problem.
III. To identify the most suitable algorithm for a given application or dataset.

Comparison of Machine



This figure provides a comprehensive comparison of YOLOv8, SSD, and FCNN across various object detection metrics. It includes mean Average Precision (mAP) at two thresholds, training time per epoch, and specific losses for both bounding box and class prediction. Additionally, the figure presents Precision and Recall charts, illustrating each model's effectiveness in identifying objects while minimizing false alarms.

Discussion

In choosing the most suitable object detection algorithm, we carried out a detailed comparison of SSD, FCNN, and YOLOv8, focusing on key performance metrics like mean Average Precision (mAP), loss reduction, precision, recall, and training efficiency. The results clearly pointed to YOLOv8 as the top performer, offering an excellent balance between accuracy and computational practicality. It consistently outperformed the others in both mAP at 50 percent and the more challenging mAP at 50 to 95 percent, while SSD lagged behind with relatively stagnant accuracy. FCNN showed strong mAP scores as well, but its high accuracy came at a steep cost—training time per epoch increased dramatically, making it less feasible for large-scale or time-sensitive projects.

In contrast, YOLOv8 showed a much more efficient learning curve, with quicker and more consistent reductions in both box and class loss during training—something SSD could not match. It also struck a solid balance between precision and recall, doing a better job of minimizing false positives while reliably identifying true objects. When it came to training time, YOLOv8 kept things manageable with a steady, linear increase, whereas FCNN's time demands increased rapidly and SSD, although faster, never quite delivered the accuracy needed.

Overall, YOLOv8 offered the best trade-off between performance and efficiency. Its consistent accuracy, fast convergence, and reasonable computational requirements make it an ideal fit for our needs, effectively balancing strong detection capabilities with practical usability

References

Juras, E. (2025, January 3). Train YOLO Models in Google Colab [Jupyter notebook]. EJ TechnologyConsultants.*https://colab.research.google.com/github/EdjeElectronics/Train-and-Deploy-YOLO-Models/blob/main/Train_YOLO_Models.ipyn*

Young, J. Balanca, P. (n.d.). SSD-Tensorflow [Source code]. GitHub. *https://github.com/balancap/SSD-Tensorflow*

Girshick, R. (2018, January 23). *fast-rcnn* [Source code]. GitHub. *https://github.com/rbgirshick/fast-rcnn*