International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(Research Article)

Waste Detection on Mobile Devices: Model Performance and Efficiency Comparison

Eva Urankar *

University of Ljubljana, Faculty of Electrical Engineering, Slovenia.

International Journal of Science and Research Archive, 2025, 15(01), 722-731

Publication history: Received on 03 March 2025; revised on 08 April 2025; accepted on 11 April 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.1.1052

Abstract

This study evaluates object detection models for mobile deployment by comparing YOLOv11 and EfficientDet-Lite using a waste classification dataset. EfficientDet-Lite0 demonstrated higher speed (13 FPS), YOLOv11n was the most power-efficient (125,000 µAh in 590 seconds), and YOLOv11m achieved the highest accuracy (mAP@50: 0.694). The deployment of these models on an Android application highlights their trade-offs: EfficientDet-Lite0 suits speed-critical tasks, YOLOv11n excels in power-sensitive scenarios, and YOLOv11m and YOLOv11s perform best in accuracy-driven applications. These findings inform the selection of optimal models for efficient and accurate waste sorting in mobile and edge computing environments.

Keywords: YOLO; Efficient Det; Waste Detection; Mobile AI; Edge Computing

1. Introduction

The growing global emphasis on waste management has increased interest in leveraging artificial intelligence (AI) to promote environmental sustainability. Object detection, a core application of AI, plays a crucial role in automating waste classification by accurately identifying recyclable materials. This capability is particularly valuable on mobile devices, which are widely available and enable individuals to directly contribute to sustainability efforts. However, deploying AI models on mobile platforms presents unique challenges, including limited computational resources, memory constraints, and power efficiency requirements.

Mobile devices differ from traditional computing systems in terms of hardware architecture. Unlike conventional computers with separate RAM for computation and VRAM on GPUs for graphical and AI tasks, mobile devices use unified memory. This shared resource must accommodate both running AI algorithms and storing intermediate data, requiring careful optimization. Additionally, mobile devices often rely on general-purpose processors with limited computational capacity; however, some devices now feature AI accelerators to enhance machine learning performance. Running AI models, especially for extended periods, can quickly deplete battery reserves, underscoring the need for models that balance computational demand, memory usage, and energy efficiency.

State-of-the-art object detection models, such as YOLO (You Only Look Once) and EfficientDet, have demonstrated high performance in various domains. While these models have been deployed on mobile devices for general object detection tasks, their effectiveness in waste detection and evaluation of their hardware efficiency metrics, such as detection time, Frames Per Second (FPS), memory usage, and battery consumption, remain underexplored. Addressing these aspects is critical for developing practical and efficient mobile AI solutions tailored for resource-constrained environments.

This study assesses the YOLOv11 and EfficientDet-Lite models for mobile-based waste detection, focusing on their hardware efficiency metrics. By providing comprehensive measurements on an Android-based platform, this work

* Corresponding author: Eva Urankar

demonstrates the feasibility of real-time waste classification and highlights the potential of edge AI for implementing accessible and scalable waste management solutions.

The remainder of this paper is structured as follows: Section II reviews related work, emphasizing mobile object detection and waste classification studies. Section III outlines the methodology, including dataset preparation, model evaluation, and deployment. Section IV presents experimental results and comparative analyses. Finally, Section V concludes with key findings and future research directions.

2. Related work

Common object detection models used on mobile devices include YOLO, SSD (Single Shot Detector) combined with MobileNet, and EfficientDet. These are single-stage algorithms that perform detection in a single pass through the input image, making them fast and suitable for real-time detection (1).

Evaluating the performance of these models typically involves "accuracy", which in object detection is commonly evaluated using the mean Average Precision (mAP) score. The metric is a standard measure of a model's effectiveness in identifying and localizing objects across multiple classes in images. The mAP score ranges from 0 to 1, with a value of 1 (100%) representing the highest achievable accuracy. Additionally, Frames Per Second (FPS) quantifies the number of frames a model can process per second, which serves as a key indicator of computational efficiency.

2.1. Model Overview

YOLO divides the input image into a grid of cells, predicting bounding boxes and class probabilities for each cell simultaneously, resulting in quick object detection (1). For low-resource devices, YOLO models are adapted using quantization (FP16 or INT8) to reduce the model size and computation without significant accuracy loss. Versions such as YOLOv4-tiny are optimized for low-power devices. They offer faster responses at a cost of accuracy (2).

The MobileNet neural network model is designed specifically for mobile devices and embedded systems (3). The combined MobileNet-YOLO model leverages the strengths of both architectures to realize efficient and accurate object detection, with MobileNet serving as the backbone for feature extraction.

SSD is an efficient object detection model that uses multiple feature maps at different resolutions to efficiently detect objects of various sizes (1). SSD is often combined with MobileNet for mobile devices (3).

EfficientDet is based on the EfficientNet family of convolutional neural networks. It is designed for high efficiency by systematically balancing network depth, width, and resolution (4).

2.2. Model Performance Comparison in Prior Studies

Due to rapid advancements in object detection technologies, our review was limited to studies conducted in 2023. However, research on YOLOv11 is sparse because it was released only shortly before this study, and EfficientDet-Lite has received limited attention in the literature.

Luo et al. (5) compares models for real-time road damage detection and found that YOLOv4 stood out for its accuracy, while YOLOv4-tiny and YOLOv7-tiny excelled in processing speed. EfficientDet-D0 provided balanced performance, combining reasonable accuracy with compact model size.

In another study, Priyadi and Suharjito (6) compared YOLOv8s and EfficientDet-Lite4 for detecting the ripeness of palm oil fruit bunches. YOLOv8s demonstrated higher accuracy and speed, whereas EfficientDet-Lite4 excelled in bounding box precision.

As summarized in Tables 1 and 2, it can be concluded that YOLO models offer superior accuracy and speed, and EfficientDet models achieve a balance between accuracy, resource efficiency, and precision. However, limited research examines the hardware efficiency metrics (e.g., memory usage, power consumption) of these models on mobile devices.

Table 1 Summary of Model Performance from Luo et al. (5)

Model	mAP	FPS	Model Size (MB)
YOLOv4	0.554	35.09	244.34
YOLOv3	0.522	39.48	235.02
SSD	0.508	41.56	91.62
EfficientDet-D0	0.507	28.44	14.95
YOLOv4-tiny	0.474	135.31	22.47
YOLOv7-tiny	0.513	102.34	23.09

Table 2 Summary of Model Performance from Priyadi and Suharjito (6)

Model	mAP	FPS
YOLOv8s	0.97	10–15
EfficientDet-Lite4	0.94	1–1.5

2.3. Selected Algorithms

This study evaluates six models: YOLOv11n, YOLOv11s, YOLOv11m, EfficientDet-Lite0, EfficientDet-Lite2, and EfficientDet-Lite4.

2.3.1. YOLOv11

The YOLOv11 architecture, like its predecessors, includes three main components: the backbone, the neck, and the head. The backbone serves as the primary neural network for feature extraction from the input image, the neck acts as a bridge between the backbone and the head for additional processing and feature fusion, and the head handles the final stages of object detection and classification (7).

YOLOv11 introduces a customized backbone featuring the C3k2 block, which is an improved version of the Cross Stage Partial (CSP) block. This modification employs a smaller kernel size to perform feature extraction more efficiently. The backbone's initial layers downsample the input image, reducing spatial dimensions while expanding the channels for detailed feature representation. YOLOv11 also introduces a C2PSA (Cross Stage Partial with Spatial Attention) block. This spatial attention mechanism allows YOLOv11 to focus on important regions within the image, thereby enhancing its ability to detect objects of various sizes with improved precision.

The neck combines feature maps of different scales and feeds them to the head.

In the head, YOLOv11 refines and processes feature maps across different scales, allowing for more detailed and accurate predictions. YOLOv11's head also includes several CBS (Convolution-BatchNorm-SiLU) layers that help stabilize and normalize feature maps, making them suitable for final predictions. Each detection branch ends with Conv2D layers that produce bounding boxes, object scores, and class probabilities for the detected objects. Finally, the head uses the Detect layer to consolidate these predictions.

To minimize overlapping predictions, YOLOv11 employs Non-Maximum Suppression (NMS), which filters out redundant bounding boxes and retains only the most reliable ones.

2.3.2. EfficientDet

The EfficientDet architecture also includes three main components: the backbone, neck, and head.

The EfficientDet backbone employs EfficientNet, which is designed for scalability and high performance. The EfficientNet classification network introduced the compound scaling method, which uniformly increases the network's depth, width, and resolution using a set of fixed scaling coefficients (4).

Instead of the traditional Feature Pyramid Network (FPN), the neck in EfficientDet uses an improved weighted bidirectional FPN (BiFPN). BiFPN enables features from different levels of the backbone to be fused, improving detection accuracy and efficiency through enhancements such as weighted feature fusion (8).

The EfficientDet head includes classification and bounding box regression modules, which use the features provided by BiFPN to accurately predict the position and category of objects within the image.

EfficientDet-Lite is a simpler version of EfficientDet that utilizes smaller and less complex versions of the EfficientNet backbone and a simplified version of BiFPN, which reduces system resource requirements but results in slightly lower prediction accuracy.

2.3.3. Model Performance Comparison based on Documentation

According to the performance data provided in the models' documentation (Table 3), YOLOv11 models consistently demonstrate higher accuracy than the EfficientDet-Lite models on the COCO dataset. Although the accuracy trends are clear, a direct comparison of speed between YOLOv11 and EfficientDet-Lite models is not included in the documentation due to the absence of consistent measurements on identical hardware.

Table 3 Performance Comparison (mAP) for EfficientDet-Lite and YOLOv11 models on the COCO Dataset (9,10)

Model	**mAP**	**FPS**	**Model Size (MB)**
YOLOv4	0.554	35.09	244.34
YOLOv3	0.522	39.48	235.02
SSD	0.508	41.56	91.62
EfficientDet-D0	0.507	28.44	14.95
YOLOv4-tiny	0.474	135.31	22.47
YOLOv7-tiny	0.513	102.34	23.09

2.4. Waste Detection

Reviewed studies contribute significantly to waste detection research, each addressing a distinct challenge. Huang et al. (11) explore vision transformers (ViTs), a model architecture that uses self-attention mechanisms to capture global contextual information, for waste classification, achieving superior accuracy compared to traditional methods. This demonstrates that ViTs can effectively capture the global context by deploying models via cloud integration for portable devices. However, it focuses on controlled datasets with limited waste types and does not extend to include critical real-life metrics, such as device-specific performance or energy efficiency.

Jain et al. (12) investigate underwater trash detection using YOLOv8 and Mask R-CNN models. It finds YOLOv8 excels in speed and detection precision, while Mask R-CNN performs well in complex underwater scenes. Despite these achievements, the current focus remains confined to underwater environments without addressing broader waste detection scenarios or mobile-specific constraints.

Although previous studies offer significant insights into waste detection, they do not focus on essential hardware efficiency metrics and the scalability of object detection models for mobile platforms. This study addresses these gaps by systematically evaluating energy consumption, processing speed, and memory usage, providing a comprehensive analysis tailored to waste detection on resource-constrained devices.

3. Methods

3.1. Dataset

To compare model performance, YOLOv11n, YOLOv11s, YOLOv11m, EfficientDet-Lite0, EfficientDet-Lite2, and EfficientDet-Lite4 were trained.

Out of all available datasets for waste classification, Garbage Classification 3 (13) was chosen because it is the largest dataset in its domain: 10,464 images across six different classes of waste. It is highly documented and well-rated on

Roboflow (14). It is important to note that the classes are not equally represented (biodegradable: 2,220; plastic: 1,162; cardboard: 1,439; glass: 2,649; paper: 1,746; metal: 1,248), due to greater diversity within certain classes. Examples of the dataset's images and their bounding boxes are provided in Figure 1.

To ensure compatibility with the input resolution required by each model, the dataset images, which originally varied in size, were resized accordingly. Additionally, data augmentation techniques were employed to enhance variability, including a 50% probability of horizontal and vertical flips and an equal likelihood of applying one of the following 90-degree rotations: none, clockwise, counterclockwise, or upside down.

The annotated dataset was exported in YOLOv11 format and Pascal VOC XML format to facilitate compatibility with the training and evaluation pipelines of YOLO and EfficientDet models, respectively. The data were split into training data (70%), validation data (20%), and test data (10%).

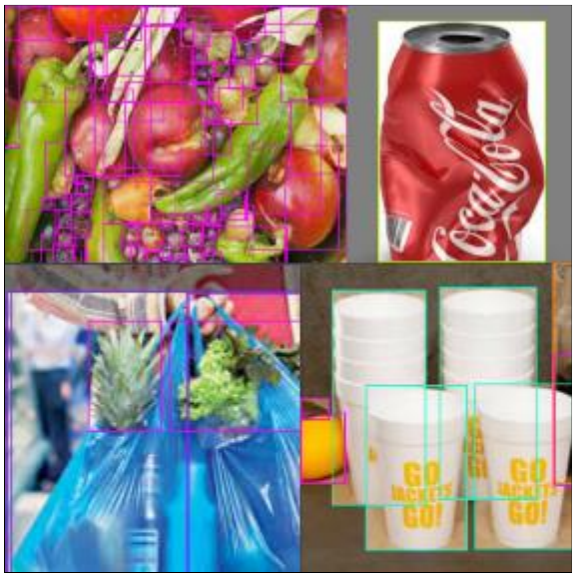


Figure 1 Four Image Examples from the Garbage Classification 3 Dataset

3.2. Model Training

The training and evaluation of the models were conducted in a high-performance computing (HPC) environment equipped with an NVIDIA H100 PCIe GPU with 81,110 MiB of VRAM. To train the YOLOv11 models, we used pretrained weights from the Ultralytics library (15), which facilitates easy training and evaluation. Google's TensorFlow Lite Model Maker (16) was used to train, evaluate, and export EfficientDet-Lite models with pretrained weights. This approach improved the process efficiency and optimized the models for edge deployment.

In our training setup, most models used a batch size of 16, except for EfficientDet-Lite0 and EfficientDet-Lite2, which employed a batch size of 32. By default, YOLO and EfficientDet-Lite4 models process images at a 640-pixel resolution, whereas EfficientDet-Lite0 uses 320x320 pixels and EfficientDet-Lite2 uses 448x448 pixels. The YOLO and EfficientDet models were trained for 150 and 85 epochs, respectively. The Ultralytics library automatically exports the best-performing model during training, which allows us to set a higher number of epochs to ensure optimal model selection.

3.3. Model Deployment and Resource Efficiency Metrics

To comprehensively evaluate model performance and assess their behavior on an edge device, we developed an Android application using Android Studio. This application facilitates testing and comparing the YOLOv11 and EfficientDet-Lite models directly on a mobile platform. The models were exported in TensorFlow Lite (TFLite) format with 8-bit integer (INT8) quantization, optimizing them for mobile deployment by reducing size and computational requirements. Custom classes were implemented within the application to handle model loading, input image preprocessing, and object detection with bounding box predictions. Additionally, the application leverages TensorFlow Lite's APIs for efficient inference, incorporating GPU delegation to accelerate processing.

The application was deployed on a Xiaomi phone equipped with 8 GB RAM, a MediaTek Dimensity 1080 processor with an octa-core CPU, and a 5000mAh battery. In addition to displaying data on the screen, the application generates a CSV file while running the models, recording various resource efficiency metrics to evaluate model efficiency.

The measurements were conducted while running the Android application to assess model performance and resource use. Detection time was determined by calculating the duration of each inference using system clock timestamps. CPU usage was monitored using Android debugging tools to record the thread's active processing time, and Frames Per Second (FPS) was computed as the reciprocal of the average frame processing time. Memory utilization was evaluated using the ActivityManager API (17), which provides real-time data on available and used system memory. Battery-related metrics, including instantaneous current, average current, remaining charge capacity, and voltage, were collected using the BatteryManager API (18). These metrics leverage the device's hardware sensors and battery management software to deliver a combination of direct measurements and reliable estimates.

We performed the measurements for each model over exactly 590 seconds, using the mentioned device under consistent conditions. The device was fully charged, with no other processes running, the brightness was set to maximum, and battery-saver mode was disabled to ensure reliable and comparable results across all tests.

4. Experiments and results

Training durations were as follows: EfficientDet-Lite0 completed in 2.36 hours, EfficientDet-Lite2 in 4.39 hours, and EfficientDet-Lite4, with a smaller batch size, extended to 14.46 hours. YOLOv11 models trained more rapidly, with YOLOv11n finishing in 2.08 hours, YOLOv11s in 2.17 hours, and YOLOv11m in 2.25 hours.

4.1. Model Performance Metrics

After training and evaluating the models, we processed the data and analyzed the following performance metrics on the test set.

Precision indicates the proportion of correctly identified positive detections out of all detections made by the model; Recall measures the proportion of correctly identified positive detections out of all actual positives in the dataset; Intersection over Union (IoU) is a metric that measures the overlap between a model's predicted bounding box and the ground truth box. In this study, detection was classified as true or false based on whether the IoU exceeded the threshold of 0.5; mAP@50 (mean Average Precision at 50% IoU) represents the model's accuracy in detecting objects where the predicted bounding boxes overlap the ground truth bounding boxes by at least 50%; mAP@50-95 extends this accuracy measurement across multiple IoU thresholds, specifically 10 thresholds ranging from 0.50 to 0.95 in increments of 0.05; The F1 Score tells us how well a model balances Precision and Recall–essentially showing the model's ability to detect all relevant objects (high Recall) while keeping false detections low (high Precision).

The results presented in Table 4 demonstrate that the performance metrics for EfficientDet-Lite and YOLOv11 models are closely aligned, with minimal variation across the evaluated parameters. YOLOv11 models demonstrate marginally higher precision and mAP@50, indicating a slight advantage in accuracy, while EfficientDet-Lite models achieve superior recall, reflecting their strength in identifying a higher proportion of relevant objects.

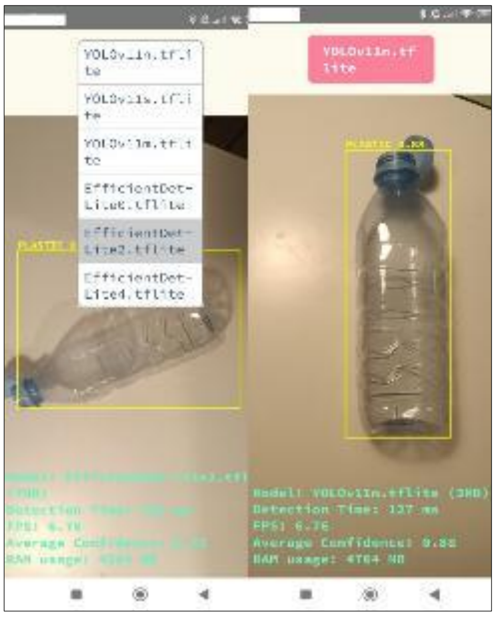
Table 4 Performance Comparison of EfficientDet-Lite and YOLOv11 Models

	Precision	**Recall**	**mAP@50**	**mAP@50-95**	**F1 Score**
EfficientDet-Lite0	0.742	0.601	0.579	0.379	0.662
EfficientDet-Lite2	0.743	**0.619**	0.633	0.430	0.676
EfficientDet-Lite4	0.762	0.613	0.648	0.458	0.679
YOLOv11n	0.754	0.583	0.669	0.483	0.656
YOLOv11s	0.771	0.607	0.693	0.511	0.680
YOLOv11m	**0.773**	0.609	**0.694**	**0.517**	**0.681**

4.2. Resource Efficiency Metrics

As shown in Figure 2, our garbage detection application allows users to select between the YOLOv11 and EfficientDet-Lite models, view device hardware specifications, and observe model performance. The application can be used to identify the appropriate waste bin for a recyclable item and to compare the processing performance of each model on a mobile device.

To evaluate the performance of the models on our device with a total available RAM capacity of 5,535 MB, we conducted a detailed analysis. Tables 5, 6, and 7 summarize the mean values of most resource efficiency metrics for the models. The bottom rows show the battery capacity drop, which indicates the difference between the minimum and maximum values rather than the average.


YOLOv11n.tflite
YOLOv11s.tflite
YOLOv11m.tflite
EfficientDet-Lite0.tflite
EfficientDet-Lite2.tflite
EfficientDet-Lite4.tflite
YOLOv11n.tflite

```
Model: YOLOv11n.tflite (3HD)
Detection Time: 137 ms
FPS: 6.76
Average Confidence: 0.88
RAM usage: 4764 MB
```

Figure 2 Garbage Detection Application Interface for Detecting Recyclable Items and Measuring Model Performance on an Android Device

4.2.1. Memory

The memory (RAM) usage of each model reflects its computational demand and is influenced by architecture and optimization rather than storage size alone. For example, YOLOv11 models generally required less RAM despite varying storage sizes, while EfficientDet-Lite2 exhibited the highest RAM usage among all models. A detailed comparison of RAM usage and model sizes is provided in Table 5.

Table 5 Memory Usage and Model Sizes for YOLOv11 and EfficientDet-Lite Models

Model	Used Memory (MB)	Model Size (MB)
YOLOv11n	965.33	2.89
YOLOv11s	810.25	9.63
YOLOv11m	870.38	20.03
EfficientDet-Lite0	934.39	4.35
EfficientDet-Lite2	**1,034.32**	7.23
EfficientDet-Lite4	916.94	**20.08**

4.2.2. Power Consumption

The YOLOv11 models demonstrated better power efficiency than the EfficientDet-Lite models, particularly when comparing models of similar sizes. For example, YOLOv11 models exhibited lower battery capacity drops and lower instantaneous current draw during the measurement period. This trend indicates that YOLOv11 models are more suitable for applications requiring efficient energy usage, whereas EfficientDet-Lite models generally consume more power. Detailed metrics on battery capacity drop and instantaneous current are summarized in Table 6.

Table 6 Power Consumption and Battery Impact for YOLOv11 and EfficientDet-Lite Models

Model	Instantaneous Current (µA)	Battery Capacity Drop (µAh)	Battery Capacity Drop (%)
YOLOv11n	773,708	125,000	2.9
YOLOv11s	796,338	131,000	3.0
YOLOv11m	966,540	178,000	4.1
EfficientDet-Lite0	835,531	136,000	3.1
EfficientDet-Lite2	933,663	166,000	3.8
EfficientDet-Lite4	1,087,949	185,000	4.3

4.2.3. Detection Time

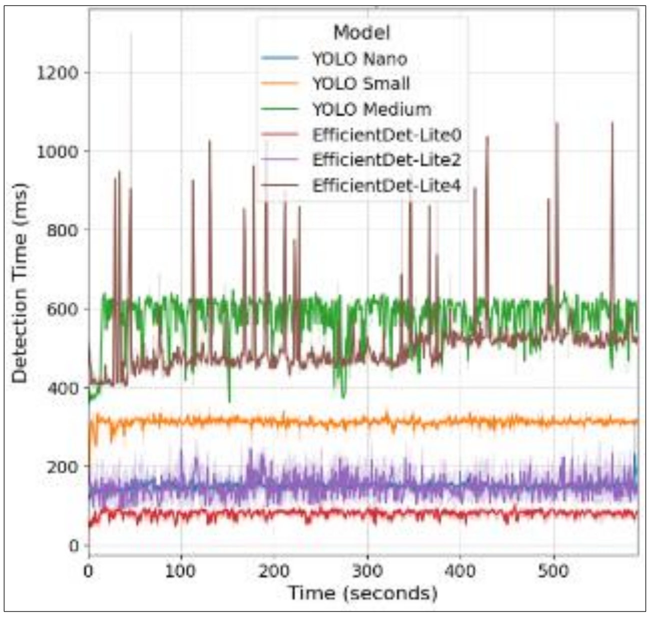


Figure 3 Detection Time (ms) over Time (s) for EfficientDet-Lite Models (EfficientDet-Lite0, EfficientDet-Lite2, EfficientDet-Lite4)

Table 7 Detection Times, CPU Times, and FPS for YOLOv11 and EfficientDet-Lite Models

Model	Detection Time (ms)	CPU Time (ms)	FPS
YOLOv11n	149.24	49.91	6.80
YOLOv11s	311.02	62.63	3.24
YOLOv11m	572.39	62.16	1.81
EfficientDet-Lite0	79.26	77.71	13.03
EfficientDet-Lite2	144.73	143.09	7.03
EfficientDet-Lite4	501.69	497.76	2.08

As shown in Figure 3, EfficientDet-Lite0 was the fastest model, achieving the lowest detection time and highest FPS, while YOLOv11n provided a good balance between speed and performance. EfficientDet-Lite2 and YOLOv11s exhibited moderate speeds, with EfficientDet-Lite2 performing slightly better in terms of FPS. The slowest models were YOLOv11m and EfficientDet-Lite4, which demonstrated the highest detection times and lowest FPS. The detection time, CPU time, and FPS metrics are summarized Table 7. Notably, EfficientDet-Lite4 occasionally experienced latency spikes, which suggests potential system resource constraints.

5. Conclusion

This study addresses a gap in the literature on mobile AI by evaluating YOLOv11 and EfficientDet-Lite models for waste detection on mobile platforms. We focused on critical performance metrics, including accuracy, detection speed, memory usage, and power efficiency. The results demonstrate notable trade-offs: EfficientDet-Lite0 achieved the fastest detection time (79 ms) and highest FPS (13), making it ideal for real-time applications. YOLOv11n exhibited superior energy efficiency, with a minimal battery capacity drop of 2.9% over 590 seconds, while YOLOv11m and YOLOv11s achieved the highest accuracy (mAP@50 of 0.694 and 0.693, respectively), making them well-suited for precision-critical tasks.

By balancing computational demand and resource efficiency, these models enable the deployment of accurate waste detection on widely available mobile devices. This enables individuals to actively contribute to sustainable waste management efforts through accessible and practical AI-driven solutions deployed on mobile devices. The study further highlights the potential of edge AI to deliver scalable solutions for global sustainability challenges.

Future research could expand this work by evaluating models across a broader range of devices, refining resource utilization, and exploring diverse application scenarios in resource-constrained environments.

Compliance with ethical standards

Acknowledgments

This study was supported by the Slovenian National Supercomputing Network, which provides access to high-performance computing resources.

Disclosure of conflict of interest

There is no conflict of interest to be disclosed.

References

[1] Cherapanamjeri J, Rao BNK. Neural Networks based Object Detection Techniques in Computer Vision. In: 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA) [Internet]. Coimbatore, India: IEEE; 2022 [cited 2024 Oct 19]. p. 1092–9. Available from: https://ieeexplore.ieee.org/document/9985581/

[2] Terven J, Cordova-Esparza D. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Mach Learn Knowl Extr. 2023 Nov 20;5(4):1680–716.

[3] Palwankar T, Kothari K. Real Time Object Detection using SSD and MobileNet. Int J Res Appl Sci Eng Technol. 2022 Mar 31;10(3):831–4.

[4] Tan M, Le QV. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Internet]. arXiv; 2020 [cited 2024 Oct 19]. Available from: http://arxiv.org/abs/1905.11946

[5] Luo H, Li C, Wu M, Cai L. An Enhanced Lightweight Network for Road Damage Detection Based on Deep Learning. Electronics. 2023 Jun 8;12(12):2583.

[6] Priyadi MRA, Suharjito. Comparison of YOLOv8 and EfficientDet4 Algorithms in Detecting the Ripeness of Oil Palm Fruit Bunch. In: 2023 10th International Conference on ICT for Smart Society (ICISS) [Internet]. Bandung, Indonesia: IEEE; 2023 [cited 2024 Oct 19]. p. 1–7. Available from: https://ieeexplore.ieee.org/document/10291928/

[7] Khanam R, Hussain M. YOLOv11: An Overview of the Key Architectural Enhancements [Internet]. arXiv; 2024 [cited 2024 Nov 10]. Available from: http://arxiv.org/abs/2410.17725

[8] Tan M, Pang R, Le QV. EfficientDet: Scalable and Efficient Object Detection [Internet]. arXiv; 2020 [cited 2024 Oct 19]. Available from: http://arxiv.org/abs/1911.09070

[9] Jocher G, Qiu J. Ultralytics YOLOv11 [Internet]. 2024 [cited 2024 Nov 10]. Available from: https://github.com/ultralytics/ultralytics

[10] EfficientDet [Internet]. Google; 2024 [cited 2024 Sep 1]. Available from: https://github.com/google/automl/tree/master/efficientdet

[11] Huang K, Lei H, Jiao Z, Zhong Z. Recycling Waste Classification Using Vision Transformer on Portable Device. Sustainability. 2021 Oct 20;13(21):11572.

[12] Jain R, Zaware S, Kacholia N, Bhalala H, Jagtap O. Advancing Underwater Trash Detection: Harnessing Mask R-CNN, YOLOv8, EfficientDet-D0 and YOLACT. In: 2024 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS) [Internet]. Coimbatore, India: IEEE; 2024 [cited 2024 Nov 30]. p. 1314–25. Available from: https://ieeexplore.ieee.org/document/10625362/

[13] Material I. GARBAGE CLASSIFICATION 3 Dataset [Internet]. https://universe.roboflow.com/material-identification/garbage-classification-3: Roboflow Universe; 2022 [cited 2024 Nov 11]. Available from: https://universe.roboflow.com/material-identification/garbage-classification-3

[14] Dwyer B, Nelson J, Hansen T. Roboflow [Internet]. 2024 [cited 2024 Jul 13]. Available from: https://roboflow.com/

[15] Ultralytics Documentation [Internet]. [cited 2024 Jul 2]. Available from: https://docs.ultralytics.com/

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Internet]. 2015. Available from: https://www.tensorflow.org/

[17] ActivityManager Class [Internet]. Google; 2024 [cited 2024 Nov 20]. (Android Developer Reference). Available from: https://developer.android.com/reference/android/app/ActivityManager

[18] BatteryManager Class [Internet]. Google; 2024 [cited 2024 Nov 20]. (Android Developer Reference). Available from: https://developer.android.com/reference/android/os/BatteryManager