Performance Evaluation of Parallel Inference Pipeline for Multi-Model Processing on Edge Devices

So-Yeon Lee[1], Tae-Jun Yoon[2] and Dae-Young Kim[2, *]

[1] Department of Software Convergence, Soonchunhyang University, Asan 31538, Korea
lsy8647@sch.ac.kr
[2] Department of Computer Software Engineering Soonchunhyang University, Asan 31538, Korea
{20214004, dyoung.kim}@sch.ac.kr

Abstract. Efficient recycling of transparent PET bottles necessitates a multi-model AI system capable of concurrently performing material classification and component detection. However, executing multiple models sequentially on resource-constrained edge devices results in cumulative inference times. In this paper, we address this challenge by implementing and evaluating a GStreamer-based parallel inference pipeline on a Raspberry Pi CM 5 equipped with a Hailo-8 AI accelerator. The implemented parallel architecture replicates input frames into two independent branches, enabling simultaneous execution of a PET/CAN classification model and a cap/ring/label detection model. Experimental results demonstrate that the parallel pipeline reduced the average frame processing time by 11.3% (from 29.50ms to 26.17ms) compared to sequential processing, thereby improving overall processing efficiency. This finding suggests that the parallel processing architecture mitigates inefficient idle times during multi-model execution, enabling stable real-time multi-task processing even in resource-constrained embedded environments.

Keywords: Edge AI, Hailo-8, Multi-Model System, Parallel Inference, Real-time Object Detection, Waste Sorting

1 Introduction

As environmental protection and resource circulation gain global importance, the development of accurate automated waste-sorting systems has emerged as a key policy priority. Transparent PET bottles, in particular, represent a high-value recyclable resource; however, contamination and misclassification during collection and sorting processes continue to reduce recycling efficiency [1]. To establish an effective recycling workflow, it is essential not only to detect PET bottles themselves but also to identify fine-grained components—such as caps, rings, and labels—whose presence directly affects the quality of recycling. This requires an intelligent system capable of performing multiple tasks simultaneously, including multi-object detection, material classification, and component-level recognition.

For such a multi-model AI model to be practically deployed in real recycling facilities, real-time operation on embedded hardware is indispensable. In a previous study

[2], our research team demonstrated that combining a Hailo-8 AI inference accelerator with a Raspberry Pi 5 can improve the inference speed of a YOLOv8n object detection model to an average of 27.18 FPS, thereby validating the feasibility of real-time processing in resource-constrained embedded environments. However, when the "PET/CAN classification" model and the "component detection" model are executed sequentially in an actual system, the inference time of each model accumulates, resulting in increased end-to-end latency. In real-time video processing systems, even a few milliseconds of additional delay per frame can significantly impact throughput and responsiveness; thus, it is crucial to efficiently integrate multiple models and optimize hardware resource utilization.

This study analyzes the frame-level latency issues inherent in sequential multi-model pipelines and implements a GStreamer-based parallel processing approach to mitigate these limitations. The proposed parallel inference pipeline duplicates the input video frames into two independent branches, delivering them concurrently to the PET/CAN classifier and the component detection model. The Hailo-8 accelerator performs inference on both models in parallel, while the GStreamer pipeline merges their outputs in real time to produce the final result. Through this mechanism, we evaluate the latency reduction achieved compared with the sequential method and demonstrate the potential scalability of the approach toward more complex multi-model scenarios.

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 describes the overall system architecture and the design of the parallel inference pipeline, Section 4 presents the experimental setup and performance evaluation, and Section 5 concludes the paper.

2 Related Work

2.1 The Need for AI System and Multi-Stage Recognition in Transparent Plastic Recycling

In recent years, the waste management sector has accelerated automation and intelligence through integration with artificial intelligence [3-6]. Lubongo et al [7] introduced recent trends in applying ML/AI to plastic recycling for identification and sorting, demonstrating that these technologies can significantly enhance the efficiency of sorting and separation processes. However, in actual recycling sites, AI-based recognition systems still face unresolved technical challenges. Cheng et al [8] developed a YOLOv7-based robotic solution for recycling beverage containers (PET, cans, glass bottles), but explicitly noted limitations in distinguishing unlabeled transparent PET bottles from transparent glass bottles using AI vision alone. Similarly, Rosca et al [9] proposed a recycling AIoT solution operating in distributed edge environments, implementing a dual recognition algorithm (PBIA) combining AI vision and weight sensors.

These prior studies demonstrate the limitations of single AI models in the transparent plastic recycling domain, supporting the need for complex recognition architectures. Building upon this, this study implements a multi-model system integrating material classification (PET/CAN) and component detection (cap, ring, label). Furthermore, it experimentally verifies the performance improvement effects of a parallel processing

approach for efficiently executing this multi-model in resource-constrained edge environments.

2.2 Edge AI Acceleration and Parallel Inference Pipeline

With the increasing demand for real-time AI inference, hardware accelerator technologies and optimization methodologies enabling efficient inference on edge devices are rapidly advancing. Particularly in resource-constrained environments, edge inference requires optimization across various aspects, including model design, model compression, compilation optimization, and collaborative inference [10].

However, when applying complex model architectures to edge environments, one faces the challenge of managing the end-to-end latency of the entire pipeline, beyond simply compressing the model. Previous studies have shown limitations, either relying on high-performance local PCs to solve this problem or accepting bottlenecks by using sequential processing approaches.

Xie et al [11] achieved a fast speed of 62FPS by implementing a single model (Yolov8n) for a plastic bottle sorting robot via local processing. However, this performance was only possible by leveraging high-spec desktop PC resources—an Intel i7 CPU and NVIDIA GeForce GTX 1650 GPU—which fundamentally differs from the low-power/resource-constrained edge environments targeted by this research. Terence et al [12] implemented a sequential pipeline to enhance small object detection performance on a Raspberry Pi 5 + Hailo-8 environment. This involved first increasing image resolution using the ESPCN model, followed by object detection using the YOLOv8n model. This achieved approximately 27FPS, comparable to the single model execution performance on the same hardware in our team's prior work. This demonstrates the structural limitation where latency accumulates when processing multi-models sequentially.

Building upon the latency accumulation issues identified in sequential processing approaches in prior studies, this research implements a multi-model execution method on a Raspberry Pi CM 5 + Hailo-8 environment using parallel processing with GStreamer. It quantitatively evaluates the resulting improvement in latency.

3 System Design and Architecture

3.1 System Overview

This study implements a multi-model parallel inference system for real-time transparent PET bottle recycling in edge environments. The system's hardware is based on the Raspberry Pi CM 5, utilizing the Hailo-8 AI accelerator connected via PCIe interface for AI inference acceleration. The Hailo-8 is a low-power NPU designed for edge devices, delivering high performance of up to 26 TOPS [13]. In prior studies, Hailo-8 demonstrated 12% higher accuracy and 20ms shorter latency compared to NVIDIA Jetson Orin Nano. When integrated with Raspberry Pi 5, it achieved stable real-time performance running the YOLOv8m model at 20FPS, outperforming Google Coral TPU, thereby proving its superior capability [14-15].

The software framework for this research is based on GStreamer. GStreamer is a framework for creating media processing pipelines, optimized for constructing complex video data flows by modularly connecting a series of plugins [16]. In this study, both the sequential pipeline and parallel pipeline used for comparison were implemented using GStreamer, enabling comparative analysis of performance evaluation based on structural differences in the pipelines.

The AI models used were (1) a PET/CAN material classification model and (2) a Cap/Ring/Label component detection model, both based on the YOLOv8n architecture. YOLOv8n is a lightweight model suitable for real-time object recognition on resource-constrained platforms. Its real-time performance of 27.18 FPS was validated in our team's prior research on a Raspberry Pi CM 5 + Hailo-8 environment.

3.2 Sequential Pipeline Architecture

This study experimentally verifies whether the latency accumulation issue caused by sequential execution of dual models in resource-constrained edge environments can be mitigated through parallel processing. First, we analyze the structure and limitations of the existing sequential processing pipeline and compare performance through parallel processing implementation using GStreamer.

Figure 1 depicts the conventional sequential pipeline structure, where video frames are processed sequentially through a single pipeline. Each frame of the input video undergoes preprocessing before being passed to the classification model (Model A). Upon completion of Model A's inference, the frame and inference result are fed as input to the detection model (Model B). Finally, the outputs from both models are merged and output.

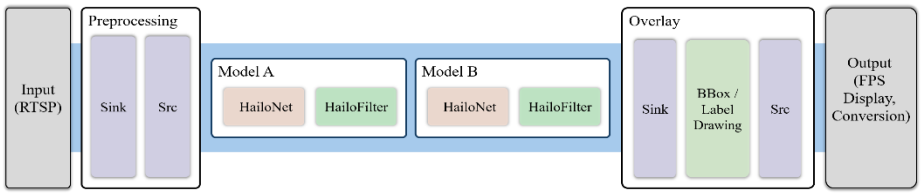

Fig. 1. GStreamer-based sequential pipeline.

The total processing delay of the sequential pipeline is a structure where the inference times of the two models are accumulated:

$$T_{total} \approx T_{modelA} + T_{modelB} + T_{overhead} \quad (1)$$

Where T_{modelA} is the classification inference time, T_{modelB} is the detection inference time, and $T_{overhead}$ includes preprocessing and data transfer overhead.

This approach has the advantage of a simple data flow and intuitive implementation. However, it has the limitation that during data transfer between models, frames repeatedly move between CPU memory and NPU memory, accumulating DMA (Direct Memory Access) transfer delays and memory access overhead. Specifically, since each

frame sequentially passes through Model A and Model B, the total processing delay increases as the sum of the inference times for both models and the overhead. Consequently, the overall system throughput is lower compared to a parallel pipeline structure, and efficiency is reduced in environments requiring real-time processing.

3.3 Parallel Pipeline Architecture

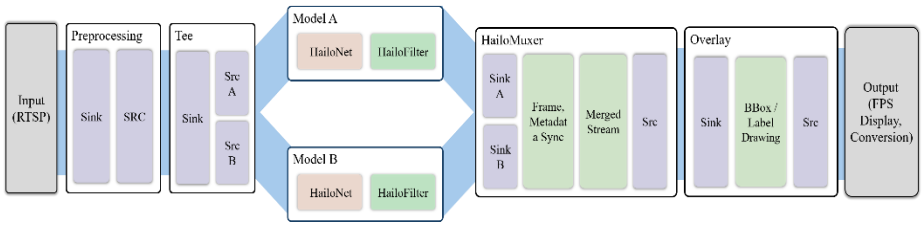

Fig. 2. GStreamer-based parallel pipeline.

To alleviate bottlenecks in the sequential pipeline, a parallel inference pipeline as shown in Figure 2 was implemented. In this architecture, after preprocessing, the input frame is replicated into two independent branches using GStreamer's tee element. The replicated frames are simultaneously fed into the classification model (Model A) and the detection model (Model B), where both models perform inference in parallel from their respective queues. The inference results from both models are merged into a single frame stream via the HailoMuxer element.

$$T_{total} \approx \max(T_{modelA}, T_{modelB}) + T_{overhead} \quad (2)$$

Where $max()$ indicates that the longer inference time dominates, rather than the sum of both models.

This parallel structure mitigates the accumulation of inference time that occurs in sequential structures by temporally overlapping the inference operations and DMA transfers of the two models.

4 Experiments and Results

4.1 Experimental Setup

This study conducted experiments by connecting a Raspberry Pi CM 5 and a Hailo-8 accelerator via a PCIe interface. The input video used a 1280x720 resolution RTSP stream, resized to 640x640 to suit the model input. Both the PET/CAN classification model and the Cap/Ring/Label detection model are based on the YOLOv8n architecture. They were converted to HEF[1] format with 8-bit quantization applied via the Hailo Dataflow Compiler and quantized using the Random Calibration method. The main

[1] HEF (Hailo Executable Format): A compiled and optimized binary format specifically designed for inference execution on Hailo NPUs, generated by the Hailo Dataflow Compiler.

specifications of the experimental environment are shown in Table 1. Performance was compared under identical conditions for both the sequential pipeline and the parallel pipeline.

Table 1. Specifications of the experimental environment.

Category	Component	Specification / Version
Hardware	Processor	Raspberry Pi CM5
	AI Accelerator	Hailo-8 (PCIe interface)
OS	OS	Debian GNU/Linux 12
Software	Gstreamer	1.22.0
	HailoRT	4.20.0
	Python	3.11.2
Input Video	Resolution / Format	1280x720 RTSP Stream
Model	Model A / Model B	Yolov8n (optimized HEF)

In this experiment, the processing speed per frame for sequential and parallel pipelines was measured for the interval from the completion of preprocessing to just before the start of overlay (Inference Latency). To eliminate variance caused by the initial warm-up phase, the first 100 frames were excluded from the measurement. Subsequently, the mean, standard deviation, median, P95 percentile, and P99 percentile were calculated for the remaining 1,000 frames.

4.2 Performance Evaluation

Table 2 compares the inference latency measurement results for the two pipelines. The sequential pipeline had an average frame processing time of 29.50ms, a standard deviation of 1.14ms, and a median of 29.44ms. P95 and P99 were measured at 31.34ms and 33.42ms, respectively. In contrast, the parallel pipeline showed an average of 26.17ms, a standard deviation of 2.85ms, a median of 27.05ms, P95 of 28.93ms, and P99 of 32.26ms.

Table 2. Statistical results for the inference latency of sequential and parallel pipelines.

Pipeline	Mean (ms)	Std (ms)	P50 (ms)	P95 (ms)	P99 (ms)
Sequential	29.50	1.14	29.44	31.34	33.42
Parallel	26.17	2.85	27.05	28.93	32.26

Comparing the two architectures, the parallel pipeline achieved an average frame processing delay reduction of approximately 11.3% compared to the sequential architecture. This is attributed to the overlapping execution of the inference operations for both models and the DMA transfer phase in the parallel architecture.

Furthermore, from a standard deviation perspective, the parallel pipeline exhibited slightly higher variance than the sequential structure. This is interpreted as frame-to-frame variation arising from the asynchronous, cross-execution of the two models' inferences. Nevertheless, the parallel pipeline maintained low average and median delays, confirming its superior overall processing efficiency.

Figure 3 presents a multi-faceted comparison of the two pipelines' performance. (a) Total System Latency and (b) Inference Latency measurements both showed the parallel pipeline improved by approximately 11%. (c) Converting this to FPS, it improved from 34.1 FPS for sequential to 37.9 FPS for parallel, an increase of about 11.1%. (d) Regarding system resource utilization, the parallel pipeline showed an 8.4% increase in CPU usage but only a 0.2% increase in memory usage, maintaining nearly identical levels.

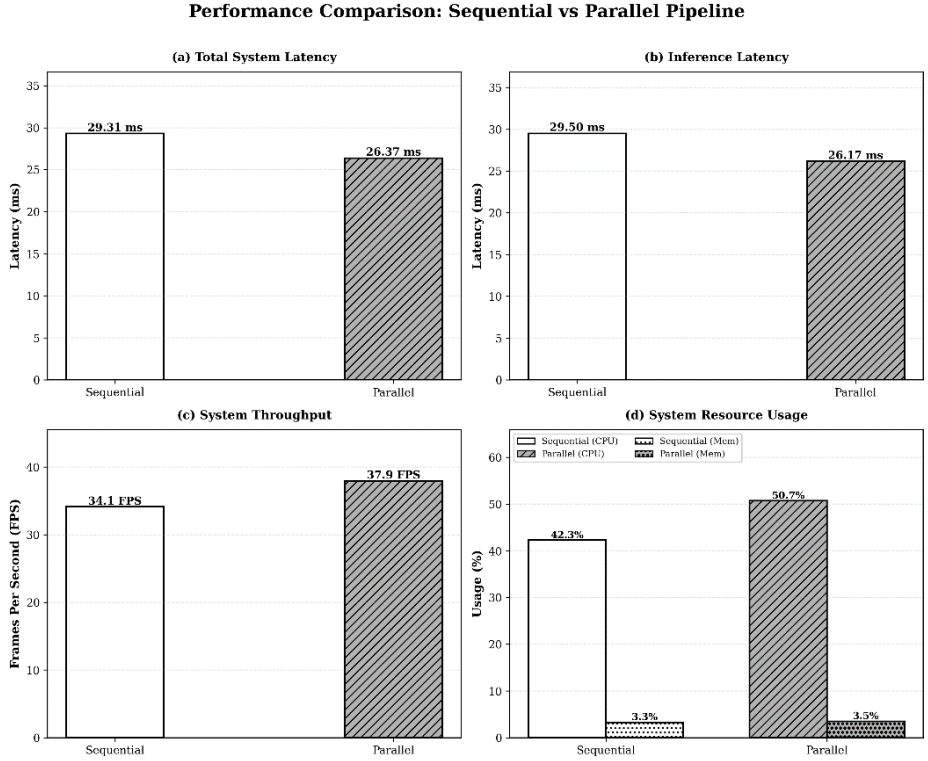

Fig. 3. Comparative analysis of sequential and parallel pipelines across key metrics: (a) Total System Latency, (b) Inference Latency, (c) System Throughput, and (d) System Resource Usage.

5 Conclusion

This study analyzes the cumulative delay problem caused by sequential processing during multi-model inference in resource-constrained edge environments. It experimentally verifies the performance improvement achieved by implementing a GStreamer-based parallel processing pipeline. Experiments conducted on Raspberry Pi CM 5 and Hailo-8 environments showed that the parallel pipeline reduced the average inference latency per frame by 11.3% (3.33ms) compared to sequential processing. This performance is attributed to the parallel overlap of both models' inference operations and DMA transfer processes within the GStreamer pipeline. This structure minimizes idle time of the NPU and related system resources that occurred during sequential processing, maximizing overall processing efficiency.

While this study focused on multi-model parallel inference, it can be extended to multi-model scenarios involving three or more models for application in more complex industrial environments. This research demonstrates the significance of a parallel processing architecture combining GStreamer and hardware accelerators as a key implementation approach capable of simultaneously ensuring real-time performance and efficiency in diverse industrial edge AI systems.

Acknowledgments. This work was funded by BK21 FOUR (Fostering Outstanding Universities for Research) (No. :5199990914048) and This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the National Program for Excellence in SW, supervised by the IITP(Institute of Information & communications Technology Planning & Evaluation) in 2025”(2021-0-01399)

References

1. Basuhi, R., et al.: Evaluating strategies to increase PET bottle recycling in the United States. Journal of Industrial Ecology **28**(4), 916–927 (2024)
2. Yoon, T.J., Jeong, Y.H., Lee, S.Y., Kim, D.: A Hailo-8-Based Lightweight Object Detection System for Transparent PET Bottle Separation. In: Summer Annual Conference of IEIE, pp. 4662-4665. Jeju, Korea (2025)
3. Satheesh, S., et al.: AI-Driven Sustainable Waste Management for Enhancing Quality of Life. In: 2025 International Conference on Emerging Information Technology and Engineering Solutions (EITES). Pp. 166-172. IEEE, Pune, India (2025)
4. Tiwari, A., et al.: Smart waste management system using AR and IoT with AI. In: 2025 International Conference on Artificial Intelligence and Data Engineering (AIDE), pp. 201-206. IEEE, Nitte, India (2025)
5. Kulkarineetham, S., et al.: Leveraging AI Chatbots to Foster Engaged Community Waste Management: A Case Study of Takhian Tia Community. In: 2025 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON). pp. 499-503. IEEE, Nan, Thailand (2025)
6. Suchdeo, E., Rajnish, R.: AI-Driven Waste Classification Model for Recycling Applications using YOLOv11. In: 2025 6th International Conference on Intelligent Communication

Technologies and Virtual Mobile Networks (ICICV), pp. 1003-1008. IEEE, Tirunelveli, India (2025)
7. Lubongo, C., Bin Daej, M.A.A., Alexandridis, P.: Recent developments in technology for sorting plastic for recycling: The emergence of artificial intelligence and the rise of the robots. Recycling **9**(4), 59 (2024)
8. Cheng, T., et al.: Optimizing Waste Sorting for Sustainability: An AI-Powered Robotic Solution for Beverage Container Recycling. Sustainability **16**(23), 10155 (2024)
9. Rosca, C.-M., Stancu, A.: Innovative AIoT Solutions for PET Waste Collection in the Circular Economy Towards a Sustainable Future. Applied Sciences **15**(13), 7353 (2025)
10. Lechner, M., Jantsch, A.: Hardware-Aware Pruning for Efficient Inference on Embedded Devices. IEEE Access **13**, 189358–189371 (2025)
11. Xie, S., et al.: Study on efficient recognition and accurate localization method of waste plastic bottles based on deep learning. Ecological Informatics **86**, 103020 (2025)
12. Terence, E., et al.: Real-Time Object Detection Enhancement Using ESPCN with ISA on Edge Devices. In: 2025 11th International Conference on Electrical Energy Systems (ICEES), pp. 217–222. IEEE, Chennai, India (2025)
13. Pereira, G.P., Chaari, M.Z.: A Look into the Performance of the Raspberry Pi AI HAT+ for Computer Vision Applications. In: 2025 11th International Conference on Communication and Signal Processing (ICCSP). pp. 1620-1625. IEEE, Melmaruvathur, India (2025)
14. Achmadiah, M.N., et al.: Fast Person Detection Using YOLOX With AI Accelerator For Train Station Safety. In: 2024 International Electronics Symposium (IES). pp. 504-509. IEEE, Denpasar, Indonesia (2024)
15. Ponneeshwaran, C., et al.: Integrating Edge AI and IoT: A Deep Learning Approach to Safe Guard Crops from Wildlife Threats. In: 2025 9th International Conference on Inventive Systems and Control (ICISC). pp. 909-916. IEEE, Coimbatore, India (2025)
16. Sanderson, J., et al.: Integrating Gstreamer with Xilinx's ZCU 104 Edge Platform for Real-Time Intelligent Image Enhancement. In: 2023 IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS). pp. 639-643. IEEE, Tempe, AZ, USA (2023)