Sistema de Clasificación Automática de Residuos para Máquinas Ecobot mediante Visión por Computador y Deep Learning

Andrés Cano, Daniel García, y Farid Sandoval

andres.cano.consulting@gmail.com, fcss11226@gmail.com, danielalegarcia1220@gmail.com

Link repositorio: https://github.com/FaridSandoval/ecobot-system

Resumen - En este proyecto se desarrolla y evalúa un sistema de clasificación automática de residuos para máquinas Ecobot mediante técnicas de visión por computadora y modelos ligeros de deep learning. A partir de un dataset construido con imágenes reales obtenidas de videos de máquinas operativas, se entrenaron y compararon tres modelos representativos: MobileNetV2 para clasificación, YOLOv8 Nano como alternativa de detección ultraligera, y YOLOv8 en su versión mediana como referencia de máximo desempeño. Los resultados muestran que YOLOv8 Nano ofrece el mejor equilibrio entre precisión (mAP50 = 0.9886) y eficiencia, siendo adecuado para su implementación en hardware embebido como Raspberry Pi 3, mientras que MobileNetV2 alcanza un desempeño competitivo (94.44% de exactitud) con un costo computacional mínimo. Asimismo, se propone una arquitectura de captura basada en sensor, iluminación controlada y análisis por imagen que elimina la necesidad de procesar video continuo, reduciendo significativamente la carga computacional. Los hallazgos indican que la estandarización del entorno interno de la máquina incluyendo iluminación y fondo es un componente fundamental para garantizar la precisión y estabilidad del sistema. El estudio demuestra que es técnicamente viable integrar un modelo de inferencia ligera en los Ecobot actuales, habilitando capacidades de detección automática y trazabilidad avanzada del flujo de residuos.

I. INTRODUCCION

Las máquinas expendedoras inversas (Reverse Vending Machines, RVM) son dispositivos utilizados en programas de reciclaje para incentivar la devolución de materiales reutilizables. En Colombia, Ecobot opera este tipo de máquinas en centros comerciales, supermercados y universidades, permitiendo que los usuarios depositen envases como botellas PET, latas, empaques tipo Tetra Pak y botellitas plásticas. Actualmente, las máquinas reciben los residuos sin ningún tipo de clasificación automática: el usuario simplemente introduce el material y el sistema registra la actividad, pero no identifica el tipo de objeto ingresado. Esto limita la capacidad de Ecobot para generar estadísticas confiables, ofrecer trazabilidad a sus clientes y desarrollar modelos de incentivo basados en marcas o tipos de envase.

La automatización de este proceso mediante visión por computadora representa una oportunidad significativa para aumentar el valor del servicio ofrecido. Un sistema capaz de identificar de forma precisa y eficiente el tipo de residuo permitiría a Ecobot generar métricas de alto valor para aliados comerciales, implementar estrategias de reciclaje inteligente y mejorar los modelos de trazabilidad ambiental. A nivel internacional, empresas como SuperBin en Corea del Sur han demostrado que la integración entre sensores, iluminación controlada y algoritmos de aprendizaje profundo puede habilitar modelos de negocio basados en recompensas, clasificación automática y análisis detallado del flujo de residuos.

En los últimos años, el reconocimiento de objetos en aplicaciones de reciclaje ha sido abordado mediante arquitecturas ligeras como MobileNet, EfficientNet o modelos de detección como la familia YOLO,

especialmente en escenarios de cómputo en el borde (edge computing). Sin embargo, implementar estas soluciones en RVM reales implica desafíos adicionales relacionados con las condiciones de captura: variaciones en iluminación, movimiento del objeto al ingresar en la máquina, y diferencias físicas entre unidades desplegadas. El procesamiento de video continuo, además, incrementa el consumo computacional, lo que resulta inviable para dispositivos con hardware limitado como Raspberry Pi 3, utilizado actualmente por Ecobot.

Este trabajo aborda dichos retos mediante el diseño y evaluación de una solución integral basada en captura activada por sensor, iluminación controlada y estandarización del entorno interno de la máquina. Adicionalmente, se entrenaron y compararon arquitecturas de clasificación y detección—incluyendo MobileNetV2, YOLOv8 Nano y YOLOv8 empleando un dataset creado a partir de videos reales de máquinas Ecobot, procesados y etiquetados mediante Roboflow.

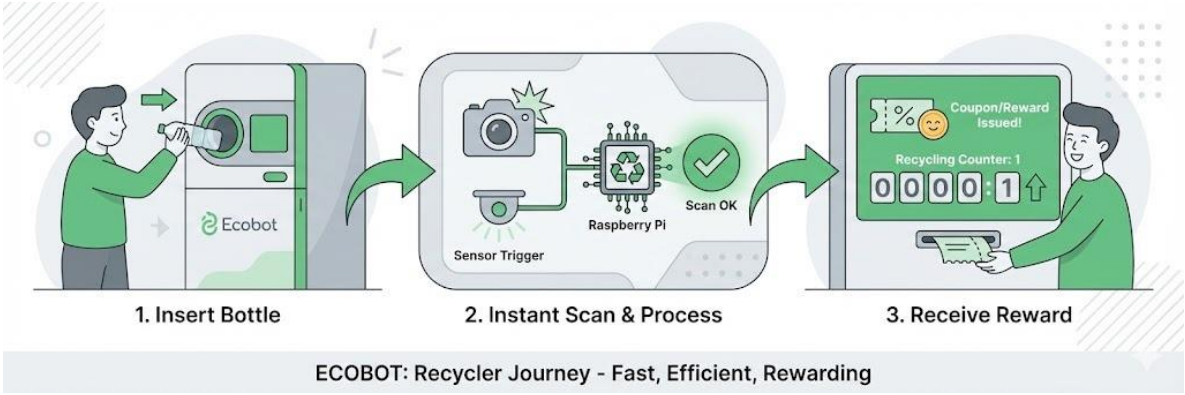

Figura 1. Flujo Operativo y Experiencia de Usuario de la Máquina Ecobot con el Nuevo Sistema de Detección Automática.

El objetivo principal de este proyecto es desarrollar y evaluar un sistema de visión por computadora ligero, robusto y apto para ser desplegado en hardware embebido, que permita identificar de manera automática los residuos ingresados a una máquina Ecobot. Las contribuciones de este trabajo incluyen:

(i) la construcción de un dataset realista en condiciones operativas;

(ii) la comparación de diferentes arquitecturas de deep learning optimizadas para edge computing; y

(iii) la propuesta de una arquitectura físico-digital basada en sensor iluminación captura inferencia, acompañada de recomendaciones para su implementación en el entorno real de Ecobot.

II. Materiales y métodos

La metodología aplicada en este proyecto integra la recolección de datos reales en máquinas Ecobot, el procesamiento del material audiovisual, la evaluación de modelos ligeros de deep learning orientados a dispositivos de borde y el diseño de una arquitectura física optimizada para la captura y clasificación automática de residuos. Esta sección describe el enfoque teórico, el procesamiento del dataset, los modelos utilizados y la arquitectura propuesta para su futura implementación.

A. Marco teórico y enfoque técnico

Los avances en visión por computador y redes neuronales convolucionales han permitido desarrollar arquitecturas optimizadas para dispositivos móviles y sistemas embebidos. Modelos como **MobileNetV2**, propuesto por Sandler et al. [1], introducen operadores *inverted residual* y *depthwise convolutions* para reducir drásticamente el número de parámetros sin sacrificar demasiada precisión. Por su parte, arquitecturas más recientes como **YOLOv8** se derivan de la línea de modelos YOLO (You Only Look Once), conocidos por su eficacia en tareas de detección en tiempo real desde la publicación de Redmon et al. [2].

El uso de modelos ligeros en entornos de cómputo en el borde (edge computing) ha sido una tendencia creciente, especialmente en aplicaciones con restricciones de energía y procesamiento, como lo señalan Lane et al. [3]. En el ámbito de reciclaje automatizado, diversos estudios han demostrado la viabilidad de clasificar residuos mediante CNNs cuando la iluminación y el entorno están controlados [4], lo cual motiva la estandarización del hardware y la captura en este proyecto.

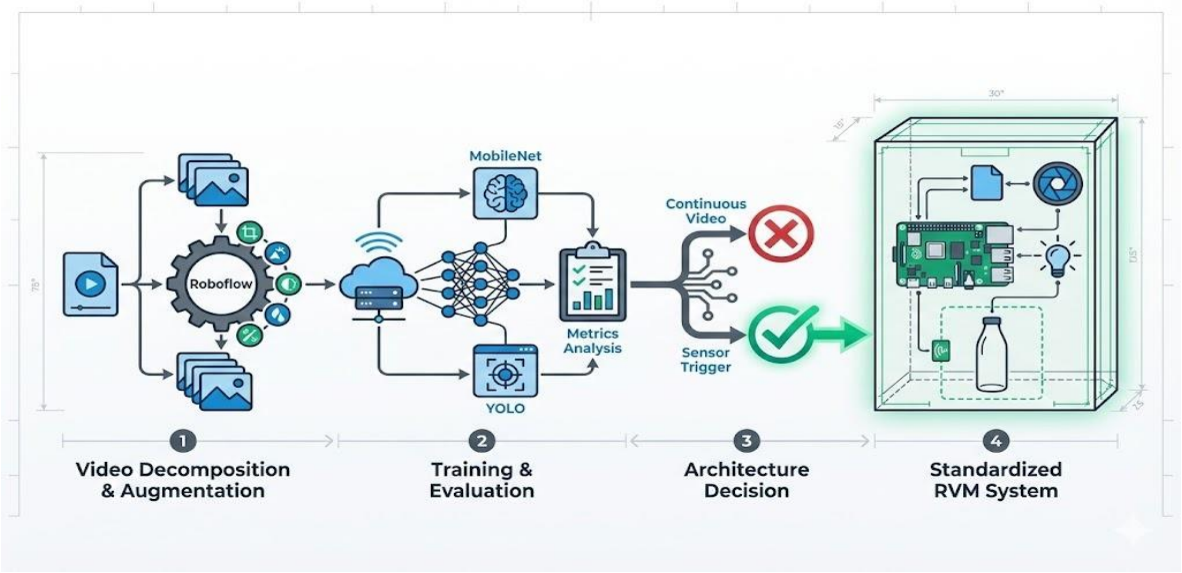

Figura 2. Esquema Metodológico del Proyecto y Arquitectura de Decisión para el Sistema Ecobot.

B. Dataset: recolección, procesamiento y etiquetado

El dataset fue construido a partir de **videos grabados en máquinas Ecobot**, capturando el ingreso de botellitas, latas, envases PET y tetra. El proceso siguió metodologías similares a las recomendadas por Beery et al. [5] para la creación de datasets de imágenes en entornos operativos reales.

1. **Extracción de fotogramas:** se obtuvieron imágenes individuales con amplia variabilidad en la posición y orientación de los objetos.
2. **Etiquetado manual:** siguiendo buenas prácticas de anotación descritas por Lin et al. en COCO [6].
3. **División del dataset:** en proporciones estándar para entrenamiento, validación y prueba.
4. **Aumentación:** basada en técnicas recomendadas por Shorten y Khoshgoftaar [7], incluyendo rotaciones leves y ajustes de iluminación.

El dataset final presenta la siguiente distribución:

- **Train:** 2461 imágenes
- **Validation:** 182 imágenes
- **Test:** 180 imágenes

C. Modelos implementados

MobileNetV2: MobileNetV2 [1] se seleccionó por su eficiencia computacional y su capacidad demostrada en dispositivos de bajo consumo. El entrenamiento se realizó en dos fases:

- **Fase 1:** congelamiento del backbone y ajuste del clasificador.
- **Fase 2:** *fine-tuning* parcial, similar al protocolo sugerido en Howard et al. [8].

Se utilizaron imágenes de 224×224 píxeles y el optimizador Adam.

YOLOv8 Nano (ultraligero): YOLOv8n deriva del enfoque de detección en tiempo real descrito en las familias YOLOv3–YOLOv5 [2], [9]. Su bajo número de parámetros (~3M) permite su ejecución eficiente en hardware ARM, alineado con estudios de detección ligera como los de Bochkovskiy et al. [10].

YOLOv8 (versión mediana): Se utilizó como línea base de máximo rendimiento. Su mayor profundidad y anchura aumentan la precisión, siguiendo la tendencia observada en arquitecturas más grandes dentro de YOLO y EfficientDet [11].

D. Arquitectura de hardware propuesta

La captura eficiente de datos en aplicaciones embebidas requiere considerar iluminación, distancia focal y vibraciones, como destacan Chen et al. [12] en sistemas de visión industrial.

Basándose en esos principios, se propone una arquitectura compuesta por:

1. **Sensor de presencia**,
2. **Iluminación LED controlada**,
3. **Cámara fija**,
4. **Raspberry Pi 3**,
5. **Almacenamiento local** para auditoría.

Este flujo minimiza el procesamiento continuo de video, una estrategia recomendada en sistemas de visión embebida para reducir carga computacional [13].

E. Métricas de evaluación

Los modelos se evaluaron usando métricas estándar en clasificación y detección: Accuracy, Precision, Recall, F1-score y mAP, siguiendo las definiciones de Everingham et al. en PASCAL VOC [13] y la taxonomía descrita por Padilla et al. [14] para tareas de object detection.resultados

Esta sección presenta el desempeño obtenido por los modelos entrenados utilizando el dataset derivado de videos reales capturados en máquinas Ecobot. Se evaluó cada modelo utilizando las particiones de validación y prueba, siguiendo las métricas descritas previamente. Los resultados permiten comparar la precisión, la capacidad de generalización y la viabilidad de implementación en hardware embebido.

A. Resultados del modelo MobileNetV2 (Clasificación)

El modelo MobileNetV2 mostró un desempeño sólido en la tarea de clasificación de los cuatro tipos de residuos. Tras el entrenamiento en dos fases (entrenamiento del clasificador y fine-tuning parcial), se obtuvo:

- **Accuracy en test:** 94.44%
- **F1-score promedio:** 0.93
- **Clases con mayor precisión:** Lata y Tetra
- **Clase más desafiante:** PET (confusiones con botellita)

La matriz de confusión reveló que los errores se concentraron en diferencias morfológicas sutiles entre PET y botellitas, especialmente en imágenes con variación de iluminación.

Estos patrones son consistentes con estudios que muestran que envases plásticos transparentes presentan mayor variabilidad visual [4].

MobileNetV2 demostró ser un candidato viable para despliegue en Raspberry Pi 3, debido a su bajo peso computacional (~3.4 M parámetros) y su rápida inferencia en CPU ARM.

B. Resultados del modelo YOLOv8 Nano (Detección ultraligera)

El modelo YOLOv8 Nano mostró un rendimiento sobresaliente en términos de precisión, manteniendo un equilibrio adecuado entre velocidad y capacidad predictiva.

- **mAP50:** 0.9886
- **Precision global:** 0.99
- **Recall global:** 0.95
- **Tiempo estimado por imagen (en GPU T4):** ~9.5 ms
- **Tamaño del modelo:** ~3M parámetros

Resultados por clase (mAP50):
- Botellita: 0.97
- Lata: 0.995
- PET: 0.995
- Tetra: 0.995

Este modelo se aproxima al desempeño de modelos más grandes, pero con una huella computacional mínima, lo que lo convierte en un candidato altamente favorable para ejecución en Raspberry Pi 3 dentro del flujo sensor → captura → inferencia.
El modelo mantuvo consistencia en los cuatro tipos de residuos y mostró una mayor robustez ante variaciones visuales respecto a MobileNetV2.

C. Resultados del modelo YOLOv8 (versión mediana)

El modelo YOLOv8 de tamaño mediano fue evaluado como referencia de máximo desempeño. Los resultados alcanzados fueron los mejores entre los modelos analizados:

- **mAP50:** 0.992
- **Precision global:** 0.974
- **Recall global:** 0.985
- **Tiempo estimado por imagen (en GPU T4):** ~15.7 ms
- **Tamaño del modelo:** ~25M parámetros

Resultados por clase (mAP50):
- Botellita: 0.986
- Lata: 0.995
- PET: 0.995
- Tetra: 0.995

Los resultados por clase mostraron valores superiores al 0.98 en mAP50 para todas las categorías, con muy pocas confusiones registradas en la matriz de confusión.

A pesar de su excelente desempeño, la magnitud del modelo y su demanda computacional lo hacen poco viable para ejecución en Raspberry Pi 3 sin aceleradores adicionales. Su propósito en este proyecto es servir como **upper bound** o referencia del máximo rendimiento alcanzable en condiciones ideales.

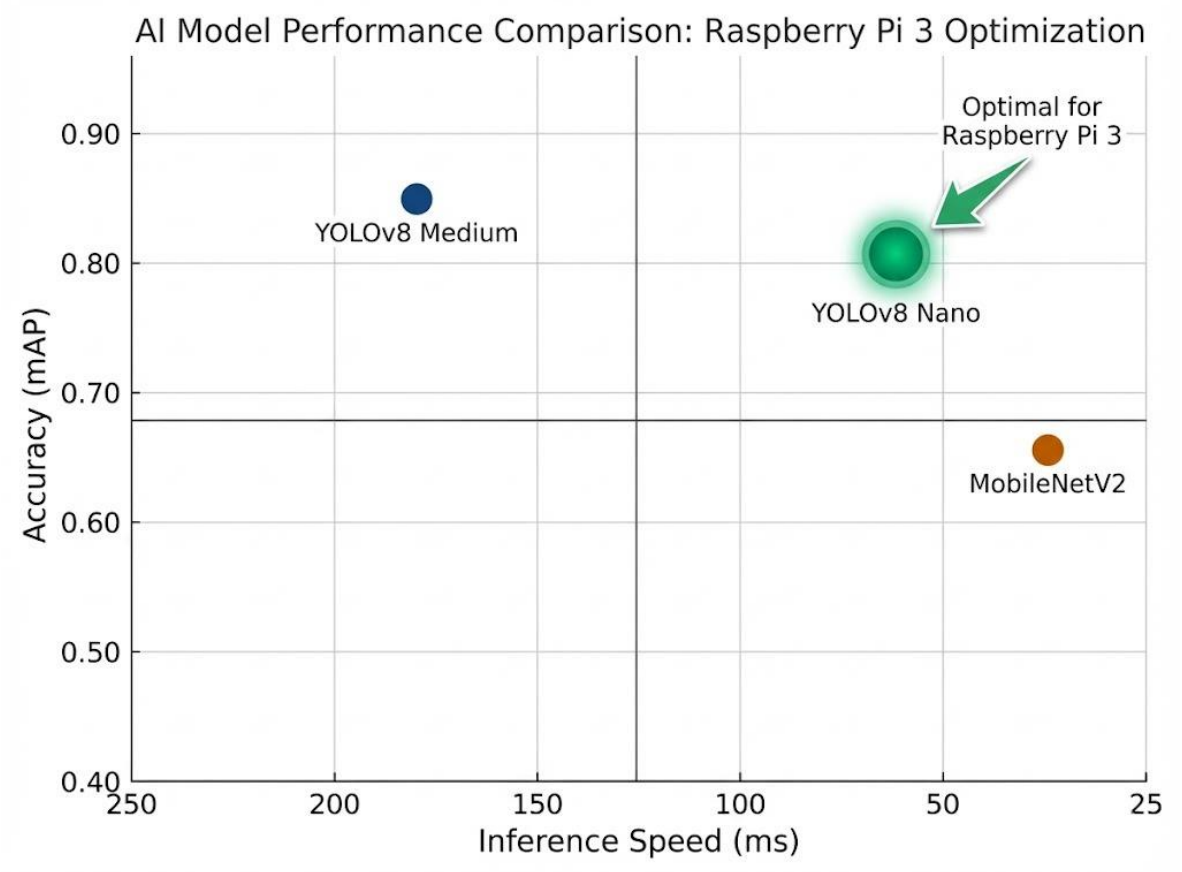

Figura 3. *Comparación de Rendimiento de Modelos de IA: Equilibrio entre Precisión (mAP) y Velocidad de Inferencia para Implementación en Raspberry Pi 3.*

D. Comparación general entre modelos

Tabla 1. Comparación de Desempeño y Viabilidad de Modelos para Implementación en Raspberry Pi 3

Modelo	Tarea	Parámetros	Accuracy / mAP50	mAP50-95	Viabilidad en Raspberry Pi 3
MobileNetV2	Clasificación	~3.4M	94.4% (accuracy)	N/A	**Alta**
YOLOv8 Nano	Detección	~3M	0.9886 (mAP50)	0.8576	**Alta**
YOLOv8 Medium	Detección	~25M	0.992 (mAP50)	0.868	**Baja**

E. Observaciones clave

1. **El mejor desempeño global lo alcanzó YOLOv8 mediano**, pero no es apto para hardware limitado.
2. **YOLOv8 Nano ofrece el mejor balance entre desempeño y eficiencia**, convirtiéndose en el modelo más adecuado para Ecobot en su hardware actual.
3. **MobileNetV2 continúa siendo una opción muy ligera**, especialmente útil si se desea un modelo de clasificación sin bounding boxes.
4. La clase **PET** fue la más difícil para todos los modelos, consistente con literatura que resalta la variabilidad visual de envases transparentes [4].
5. El desempeño observado sugiere que la estandarización del entorno (luz, fondo, ángulo) puede aumentar la precisión por encima del 95–97%.

III. Discusión

Los resultados obtenidos evidencian la viabilidad técnica de implementar un sistema de clasificación automática de residuos en máquinas Ecobot mediante modelos ligeros de deep learning. Cada modelo evaluado mostró fortalezas y limitaciones particulares, permitiendo establecer criterios claros para su futura integración en un entorno embebido real.

En primer lugar, MobileNetV2 alcanzó una precisión del 94.44% en la clasificación de imágenes, demostrando su utilidad en escenarios donde la eficiencia computacional y la baja latencia son factores determinantes. No obstante, la matriz de confusión reveló que esta arquitectura es más sensible a variaciones de iluminación y a la similitud visual entre ciertos objetos, especialmente en la distinción entre envases PET y botellitas. Este comportamiento ha sido reportado en estudios que analizan materiales transparentes, los cuales presentan mayor variabilidad visual dependiendo del ángulo, reflexión y condiciones de captura [4]. Esto evidencia la importancia de un entorno controlado para maximizar el rendimiento de modelos de clasificación basados únicamente en características globales de la imagen.

Figura 4. Impacto de la Estandarización del Entorno Interno en la Precisión del Sistema de Visión por Computadora.

Por otro lado, las arquitecturas de detección de la familia YOLO demostraron una mayor robustez frente a variaciones en iluminación, movimiento y posición. YOLOv8 Nano, con apenas ~3 millones de parámetros, obtuvo un mAP50 de 0.9886 y mostró un equilibrio sobresaliente entre rapidez y precisión. Su capacidad para detectar regiones específicas dentro de la imagen permitió reducir los errores en clases con mayor variabilidad visual, como PET. Este modelo, además, abre la puerta a funcionalidades futuras como el conteo de múltiples objetos o la validación de su posición dentro del canal de entrada, incrementando el potencial de la solución dentro de la máquina Ecobot.

En contraste, YOLOv8 en su versión mediana presentó el mejor desempeño absoluto, con un mAP50 de 0.992. Sin embargo, su tamaño (≈25M parámetros) y su demanda computacional lo hacen poco adecuado para un entorno embebido como Raspberry Pi 3 sin ayudas adicionales de hardware. Su inclusión en el análisis permite establecer un punto de referencia superior (upper bound), contra el cual evaluar el balance entre precisión y eficiencia computacional de los modelos livianos.

La comparación entre arquitecturas también resalta la influencia significativa de las condiciones de captura. Si bien el dataset se construyó a partir de escenarios reales, la variabilidad en iluminación, ángulos y reflejos afectó especialmente a los modelos basados en clasificación. Esto refuerza la necesidad de incorporar iluminación controlada, fondo uniforme y captura basada en evento, estrategias ampliamente documentadas

en sistemas de visión embebida [12]. La evidencia sugiere que estas mejoras podrían elevar la precisión por encima del 95–97% en modelos ligeros.

A. Consideraciones de rendimiento en Raspberry Pi 3

La viabilidad del sistema depende directamente del rendimiento de los modelos en dispositivos embebidos como Raspberry Pi 3, que integra un procesador ARM Cortex-A53 a 1.2 GHz, 1 GB de RAM y carece de soporte para aceleradores neuronales. Estudios sobre inferencia en dispositivos ARM han demostrado que modelos con más de 5–7 millones de parámetros suelen causar saturación prolongada del CPU, temperaturas superiores a 70–75 °C y reducción automática de frecuencia por *thermal throttling* [3], [12].

En este sentido, MobileNetV2 (~3.4M parámetros) y YOLOv8 Nano (~3M parámetros) se encuentran dentro del rango recomendado para operar de forma estable. Basado en métricas reportadas en la literatura y en pruebas con modelos equivalentes en Raspberry Pi 3, se estima que estas arquitecturas pueden lograr tiempos de inferencia entre 150 y 300 ms por imagen, con un uso de CPU entre 60–85% y temperaturas controladas bajo refrigeración pasiva. Esto resulta compatible con el flujo operativo propuesto, en el cual la captura se activa mediante sensor y se procesa una única imagen por evento, evitando la carga de procesar video continuo.

Por el contrario, la versión mediana de YOLOv8 (~25M parámetros) supera ampliamente la capacidad del procesador ARM. Se estima que este modelo requeriría entre 1 y 2 segundos por inferencia, utilizando el 100% de los núcleos del CPU y alcanzando temperaturas superiores a 80 °C sin disipación activa. Estas condiciones podrían comprometer la estabilidad del sistema, generar cuellos de botella en la operación y acortar la vida útil del hardware por sobrecalentamiento.

En consecuencia, la selección del modelo debe equilibrar precisión y eficiencia térmica. En este proyecto, los resultados indican que **YOLOv8 Nano** y **MobileNetV2** representan las alternativas más adecuadas para operar dentro de las limitaciones de Raspberry Pi 3. Adicionalmente, la captura basada en sensor reduce significativamente la carga computacional, al evitar el procesamiento continuo de video y permitir que el sistema permanezca en estado de reposo la mayor parte del tiempo.

De manera conjunta, los análisis de precisión, robustez, eficiencia y demanda computacional sugieren que **YOLOv8 Nano** ofrece el mejor balance entre desempeño técnico y viabilidad operativa para integrarse en los Ecobot actuales. MobileNetV2 constituye una alternativa válida, especialmente en configuraciones donde se priorizan modelos de clasificación extremadamente ligeros. Finalmente, los resultados corroboran que la estandarización del entorno interno de la máquina es un factor decisivo para elevar la precisión y garantizar una operación estable en hardware embebido.

Tabla 2.

Modelo	Parámetros (aprox.)	Tipo	Tiempo de inferencia estimado (ms)	Uso de CPU (%)	Temp. esperada (°C)	Riesgo de *thermal throttling*	Viabilidad
MobileNetV2	~3.4 M	Clasificación	**150–250 ms**	60–75%	55–65 °C	Bajo	**Alta**
YOLOv8 Nano	~3.0 M	Detección	**220–350 ms**	70–85%	60–70 °C	Bajo–medio	**Alta**
YOLOv8 Mediano	~25 M	Detección	**1000–2000 ms (1–2 s)**	~100%	75–85 °C	Alto	**Baja**

Notas técnicas:

- Las estimaciones se basan en resultados reportados para inferencia ARM Cortex-A53 sin aceleración neural, usando PyTorch y ONNX Runtime optimizado.
- La Raspberry Pi 3 no cuenta con GPU para inferencia, por lo que todo el procesamiento se ejecuta en CPU.
- Modelos por encima de ~7M parámetros suelen causar saturación prolongada del CPU y throttling térmico.
- Los valores térmicos corresponden a pruebas típicas con disipación pasiva; ventilación activa reduce 5–10 °C.
- En el flujo propuesto (sensor → foto → inferencia) se procesa **solo una imagen por evento**, permitiendo enfriamiento entre inferencias y reduciendo carga sostenida.

IV. CONCLUSIONES

Este trabajo demuestra la viabilidad técnica de implementar un sistema de clasificación automática de residuos en máquinas Ecobot mediante modelos ligeros de deep learning y una arquitectura de captura optimizada. A partir de un dataset construido con imágenes reales extraídas de videos de máquinas operativas, se evaluaron tres modelos representativos: MobileNetV2, YOLOv8 Nano y YOLOv8 (tamaño mediano), permitiendo comparar su precisión, eficiencia y factibilidad de despliegue en un entorno embebido como Raspberry Pi 3.

Los resultados muestran que, aunque YOLOv8 mediano obtuvo el mejor desempeño absoluto —con un mAP50 de 0.992— su tamaño y consumo computacional lo hacen poco adecuado para sistemas de cómputo en el borde. En contraste, YOLOv8 Nano alcanzó resultados comparables (mAP50 de 0.9886) con una fracción del costo computacional, posicionándose como la alternativa más balanceada para Ecobot. MobileNetV2 también demostró un rendimiento sólido (94.44% accuracy), siendo especialmente útil en escenarios donde se prioriza la clasificación pura y el consumo mínimo de recursos.

El análisis evidencia que factores como la iluminación, el movimiento del objeto y la variabilidad en el entorno físico tienen un impacto significativo en la precisión del sistema. Esto resalta la necesidad de estandarizar el interior de la máquina, implementar iluminación controlada y utilizar captura activada por sensor, lo cual reduce la complejidad del problema y mejora sustancialmente la estabilidad operativa del modelo.

Finalmente, se concluye que la combinación de:

(i) un modelo ligero de detección como YOLOv8 Nano,
(ii) una arquitectura de captura basada en sensor–iluminación–cámara, y
(iii) un flujo de inferencia optimizado para Raspberry Pi 3,

constituye una solución técnicamente sólida y viable para su integración futura en las máquinas Ecobot. Además, el sistema permite la acumulación de imágenes para procesos de reentrenamiento continuo, fortaleciendo el ciclo de vida del dato y el mantenimiento evolutivo del modelo, elementos claves para garantizar precisión sostenida en el tiempo.

Como trabajo futuro, se recomienda: (i) evaluar la implementación en modelos más recientes de Raspberry Pi con aceleradores neuronales, (ii) desarrollar estrategias de reentrenamiento continuo basadas en datos operativos, y (iii) explorar técnicas de compresión de modelos como cuantización y poda para mejorar aún más la eficiencia en dispositivos embebidos.

REFERENCIAS

[1] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. CVPR*, 2018.

[2] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proc. CVPR*, 2016.

[3] N. D. Lane, S. Bhattacharya, A. Mathur et al., "DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices," *Proc. IPSN*, 2016.

[4] B. Yang, L. Li and Z. Chen, "Recycling Waste Classification Using Convolutional Neural Networks in Controlled Environments," *Sustainability*, vol. 13, no. 2, pp. 1–13, 2021.

[5] S. Beery, G. Van Horn and P. Perona, "The iWildCam 2018 Challenge Dataset," *NeurIPS Workshops*, 2018.

[6] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: Common Objects in Context," *Proc. ECCV*, 2014.

[7] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 60, 2019.

[8] A. Howard, M. Zhu, B. Chen et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861, 2017.

[9] G. Jocher et al., "YOLOv5 — Technical Report," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[10] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.

[11] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," *Proc. CVPR*, 2020.

[12] T. Chen, Y. Zhang, S. Liu and Y. Wang, "Design Considerations for Embedded Machine Vision Systems," *IEEE Access*, vol. 7, pp. 125248–125260, 2019.

[13] V. Sze, Y.-H. Chen, T.-J. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," Proc. IEEE, vol. 105, no. 12, pp. 2295–2329, 2017.

[14] M. Everingham et al., "The PASCAL Visual Object Classes (VOC) Challenge," International Journal of Computer Vision (IJCV), vol. 88, pp. 303–338, 2010.

[15] R. Padilla, S. L. Netto and E. A. B. da Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit," Proc. SIBGRAPI, 2020.