



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Maestría en Ciencias de la Ingeniería

**PROTOTIPO DE UN CONTENEDOR CLASIFICADOR DE
RESIDUOS SÓLIDOS URBANOS CON REDES NEURONALES
CONVOLUCIONALES**

TESIS

**PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA INGENIERÍA**

Presenta

Dominic Brian Román Padilla

Directora

Dra. Mirna Castro Bello
INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Codirector

Dr. Cornelio Morales Morales
INSTITUTO TECNOLÓGICO DE SAN JUAN DEL RÍO

Revisor 1

Mtro. Diego Esteban Gutiérrez Valencia
INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Revisor 2

Dra. Yanet Evangelista Alcocer
INSTITUTO TECNOLÓGICO DE CHILPANCINGO

Chilpancingo de los Bravo, Guerrero, 5 de junio del 2025

Agradecimientos

Quiero iniciar este apartado expresando mi más profunda gratitud a quienes hicieron posible la culminación de este proyecto de tesis, un camino lleno de aprendizajes y desafíos.

A Dios, por brindarme la fortaleza, la fe y la guía necesaria en cada paso de este proceso. Su presencia fue un faro en los momentos de duda y un impulso constante para seguir adelante.

A mi querida abuelita, Juana Padilla Salgado por su amor incondicional, sabios consejos y ser siempre una fuente de inspiración y ternura en mi vida. Sus oraciones y palabras me motivaron a seguir adelante.

A mis tías: Giselle Janet Cabrera Padilla y Jimena Patricia Cabrera Padilla, gracias por su apoyo incondicional, sus ánimos y su cariño. A mi tío Juvenal Juan Cabrera Padilla gracias por tus consejos, tu ayuda en momentos puntuales y palabras de aliento. Vuestro respaldo ha sido muy valioso.

Al resto de mi familia, por su comprensión, paciencia y el apoyo brindado durante todo este tiempo. Saber que contaba con ustedes hizo esta travesía mucho más llevadera.

En el ámbito académico, deseo expresar mi más sincero agradecimiento a mi directora de tesis, la Dra. Mirna Castro Bello, del Instituto Tecnológico de Chilpancingo. Su invaluable orientación, constante apoyo, paciencia y experta dirección fueron fundamentales a lo largo de esta investigación. Sus acertadas observaciones, la disposición para resolver mis dudas, confianza en mi trabajo me motivaron a seguir adelante y fueron cruciales para la culminación de este trabajo. Toda la guía durante estos años me ayudó enormemente a poder lograr cada uno de los objetivos, le agradezco de todo corazón.

Asimismo, extendo un especial agradecimiento a mi codirector, el Dr. Cornelio Morales Morales pilar fundamental en este trabajo. Su perspectiva en conocimiento en el área de electrónica y su disposición para colaborar fueron de gran valor para el desarrollo y la solidez de esta tesis, su invaluable guía, paciencia y apoyo constante me ayudó a alcanzar las metas propuestas.

A la Dra. Yanet Evangelista Alcocer, Dr. Carlos Virgilio Marmolejo Vega por motivarme a seguir adelante cada día y al Mtro. Sergio Ricardo Zagal Barrera por su apoyo en este instituto.

A mi amigo Víctor Manuel Romero Juárez, el cual tengo el gusto de conocer y trabajar juntos desde ya hace 7 años, siempre está para escucharme y brindarme palabras de apoyo.

A mi amiga Lizbeth Gómez Muñoz por su apoyo durante estos años de trabajo en los momentos donde no entendía que realizar o cuando no sabía que dirección tomar.

A mis amigos de Posgrado les agradezco a cada uno por apoyarme a concluir este estudio.

Al Instituto Tecnológico de Chilpancingo y a sus autoridades, por brindarme la formación académica, las facilidades y los recursos necesarios para llevar a cabo esta investigación.

Finalmente, agradezco a la Secretaría de Educación, Ciencia, Tecnología e Innovación (SECIHTI) por el apoyo económico otorgado a través de la beca, el cual fue un soporte fundamental para dedicarme de lleno a este proyecto.

A todos ustedes, ¡muchas gracias!



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Chilpancingo
División de Estudios de Posgrado e Investigación

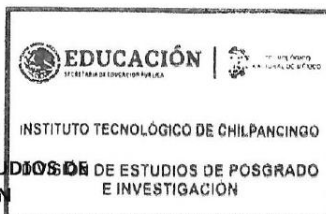
Chilpancingo, Guerrero **13/mayo/2025**
Oficio No. 329/2025

Dominic Brian Román Padilla
Candidato a Maestro en Ciencias de la Ingeniería
PRESENTE

Por este conducto me es grato comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **“Prototipo de un contenedor clasificador de residuos sólidos urbanos con redes neuronales convolucionales”**, ha informado a esta División de estudios de Posgrado e Investigación que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

ATENTAMENTE
Excelencia en Educación Tecnológica®


MIRNA CASTRO BELLO
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN



C.c.p. Subdirección Académica
C.c.p. Archivo
MCB*rlgr



2025
Año de
La Mujer
Indígena

Av. José Francisco Ruiz Massieu No. 5, Colonia Villa Moderna, Chilpancingo de los
Bravo, Guerrero, México. Tel. (747) 45 4 1300, Ext. 1310
Email: depi@chilpancingo.tecnm.mx <http://chilpancingo.tecnm.mx>
<https://www.facebook.com/TecNMcampusChilpancingo>



DRA. MIRNA CASTRO BELLO

JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO

E INVESTIGACIÓN DEL ITCHILPANCINGO

PRESENTE

Por este medio los integrantes del comité Tutorial del **C. Dominic Brian Román Padilla** con número de control **G23520624** de la Maestría en Ciencias de la Ingeniería, le informamos que hemos revisado el trabajo de tesis de grado denominado **"Prototipo de un contenedor clasificador de residuos sólidos urbanos con redes neuronales convolucionales"**, una vez que se han atendido todas las observaciones que se le indicaron, hemos acordado aceptar el documento por lo que solicitamos la autorización de impresión definitiva

Sin más por el momento, le enviamos un cordial saludo.

Directora



Dra. Mirna Castro Bello

Codirector



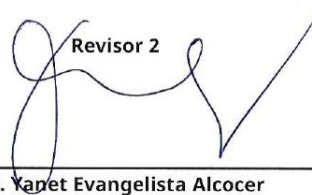
Dr. Cornelio Morales Morales

Revisor 1



Mtro. Diego Esteban Gutiérrez Valencia

Revisor 2



Dra. Yanet Evangelista Alcocer

Índice de cuadros	viii
Resumen	xi
Abstract.....	xiii
Capítulo I. Introducción.....	1
Capítulo II. Antecedentes	4
2.1 A nivel global	5
2.2 En América	8
2.3 En México.....	9
2.4 Conceptos.....	10
Capítulo III. Justificación	17
Capítulo IV. Objetivos	19
4.1 Objetivo general.....	19
4.2 Objetivos específicos	19
Capítulo V. Metodología.....	20
5.1 Materiales.....	20
5.2 Metodología	20
5.2.1 Dataset de RSU	22
5.2.2 Algoritmos avanzados para la clasificación de objetos con CNN.....	25
5.2.3 Desarrollo y entrenamiento del algoritmo CNN de aprendizaje profundo	25
5.2.4 Diseño de la estructura del prototipo contenedor.....	29
5.2.5 Construcción del prototipo	30
5.2.6 Implementación de la CNN del contenedor	41
5.2.7 Validación del prototipo	41
Capítulo VI. Resultados.....	42

6.1 Dataset de RSU	42
6.2 Algoritmos avanzados para la clasificación de objetos con CNN	43
6.3 Desarrollo y entrenamiento del algoritmo CNN de aprendizaje profundo	44
6.4 Diseño de la estructura del prototipo contenedor	47
6.5 Construcción del prototipo.....	50
6.6 Validación el prototipo.	54
Capítulo VII. Discusión	55
Capítulo VIII. Conclusiones	57
Referencias bibliográficas	59
IX. Anexos	65
Anexo 1. Código sistema embebido Raspberry Pi 4 modelo B.....	65
Anexo 2. Código sistema electrónico Arduino Mega 2560	67
Anexo 3. Código CMD de entrenamiento YOLO	71
Anexo 4. Productos académicos	72

Índice de cuadros

<i>Tabla 1 Datasets consultados.</i>	22
<i>Tabla 2 Modelos entrenados.</i>	27
<i>Tabla 3 Comparativa de modelos de CNN.</i>	43
<i>Tabla 4 Cálculos de los modelos YOLO.</i>	47

Índice de figuras

<i>Figura 1 Nuevo paradigma de programación.....</i>	<i>11</i>
<i>Figura 2 Proceso metodológico.</i>	<i>21</i>
<i>Figura 3 Imágenes de TrashNet.</i>	<i>23</i>
<i>Figura 4 Imágenes de Waste Classification Data.</i>	<i>24</i>
<i>Figura 5 Imágenes de Garbage Classification Dataset.</i>	<i>24</i>
<i>Figura 6 Representación del entrenamiento del modelo YOLO.....</i>	<i>26</i>
<i>Figura 7 Diseño esquemático del contenedor.....</i>	<i>30</i>
<i>Figura 8 Unión de la estructura mecánica (armazón).....</i>	<i>32</i>
<i>Figura 9 Materiales para la estructura interna y recubrimientos de acrílico.</i>	<i>33</i>
<i>Figura 10 Instalación de las cubiertas de acrílico.....</i>	<i>34</i>
<i>Figura 11 Integración de la base para sistema mecánico.....</i>	<i>35</i>
<i>Figura 12 Colocación del sistema mecánico.</i>	<i>36</i>
<i>Figura 13 Representación del sistema electrónico.</i>	<i>37</i>
<i>Figura 14 Ejecución del sistema electrónico.</i>	<i>38</i>
<i>Figura 15 Colocación del cubo de plástico para selección del residuo.</i>	<i>39</i>
<i>Figura 16 Diseño general del sistema de selección.</i>	<i>40</i>
<i>Figura 17 Dataset personalizado de RSU.....</i>	<i>42</i>
<i>Figura 18 Rendimiento del entrenamiento de YOLOv4 tiny</i>	<i>45</i>
<i>Figura 19 Matrices de confusión generadas.....</i>	<i>46</i>
<i>Figura 20 Diseño frontal del prototipo</i>	<i>48</i>
<i>Figura 21 Diseño del sistema interior mecánico</i>	<i>49</i>

<i>Figura 22</i>	<i>Diseño del compartimento del sistema electrónico.....</i>	<i>50</i>
<i>Figura 23</i>	<i>Diseño general del contenedor.....</i>	<i>51</i>
<i>Figura 24</i>	<i>Diseño general del contenedor.....</i>	<i>52</i>
<i>Figura 25</i>	<i>Vista lateral del contenedor.....</i>	<i>53</i>
<i>Figura 26</i>	<i>YOLOv9 tiny en Raspberry Pi.....</i>	<i>54</i>
<i>Figura 27</i>	<i>Sistema de clasificación del contenedor.....</i>	<i>54</i>

Resumen

La gestión de Residuos Sólidos Urbanos (RSU) es un gran desafío para las prácticas tradicionales en su separación, debido a un considerable incremento en cantidad, diversidad, complejidad de tipos y una alta demanda en la exactitud para clasificarlos. El reconocimiento y clasificación de imágenes de residuos empleando técnicas de visión por computadora, permite optimizar sus procesos de administración, de recolección con alta precisión, logrando una gestión inteligente en su separación y disposición final, mitigando el impacto ambiental y así contribuir a los objetivos del desarrollo sostenible. Esta investigación consistió en el desarrollo de un prototipo de contenedor clasificador de RSU con Redes Neuronales Convolucionales (CNN). Para ello se realizaron siete fases: 1. Conformación del Dataset de RSU: Trashnet, Garbage Classification y muestras capturadas; 2. Análisis y selección de algoritmos para la detección de objetos con CNN: En función de ventajas y desventajas; 3. Desarrollo y entrenamiento del algoritmo CNN: Aplicando el conjunto de datos, ajuste de hiperparámetros Momentum, BatchSize y Subdivisiones del algoritmo; 4. Diseño de la estructura del prototipo: Análisis de componentes, evaluación comparativa de requerimientos y modelado en SolidWorks; 5. Construcción del prototipo: Caja de estructura metálica de 80x80 cm, cubierta de plástico acrílico blanco, dos contenedores de 20 litros, base de madera, integración de la estructura, sistema mecánico (motor nema 23, driver TB6600) y Arduino mega 2560; 6. Implementación de la CNN en el contenedor: Exportación de los pesos entrenados de YOLOv9 en el sistema embebido Raspberry Pi 4 modelo B ejecutado en un entorno de Python y 7. Validación: Muestras de RSU de entornos reales.

Los resultados obtenidos fueron un conjunto de datos personalizado de 1883 imágenes en dos clases orgánicos e inorgánicos con una resolución de 640x640 píxeles. En cuanto a los modelos para entrenamiento se seleccionaron modelos de la arquitectura YOLO, específicamente YOLOv9 tiny con una precisión superior al 90 %. Para el diseño se realizó un modelo 3D especificando las dimensiones a escala del prototipo propuesto. Por consiguiente, se diseñó y armó la estructura metálica del contenedor, siguiendo un desarrollo lineal, además de su estructura mecánica de unos contenedores rotatorios para el depósito de los residuos, para ello se utilizaron materiales como metal, acrílico, aluminio y madera, además de un balero de banco. El circuito electrónico constó de un Raspberry Pi 4 Modelo B, cámara para inferencias, un sistema rotatorio conformado por un motor a pasos Nema 23, un driver TB6600, un Arduino mega 2560 y un servomotor SG90. También se implementaron los pesos entrenados de YOLOv9 tiny en el sistema embebido Raspberry Pi 4 modelo B ejecutado en un entorno de Python. Finalmente se obtuvo un contenedor cúbico clasificador inteligente de 120 cm² para una gestión eficiente que coadyuve a la separación de RSU y al cuidado del medio ambiente.

Palabras clave: Residuos Sólidos Urbanos, Redes Neuronales Convolucionales, Contenedor, Sistemas Embebidos, Inteligencia Artificial.

Abstract

Municipal Solid Waste (MSW) management is a great challenge for traditional separation practices, due to a considerable increase in quantity, diversity, complexity of types, and a high demand for accuracy in their classification. The recognition and classification of waste images using computer vision techniques allows for the optimization of its administration and high-precision collection processes, achieving intelligent management in its separation and final disposal, mitigating environmental impact, and thus contributing to sustainable development goals. This research consisted of the development of a prototype MSW classifying container using Convolutional Neural Networks (CNN). To achieve this, seven phases were carried out: 1. Creation of the MSW Dataset: Trashnet, Garbage Classification, and self-captured samples; 2. Analysis and selection of algorithms for object detection with CNN: Based on advantages and disadvantages; 3. Development and training of the CNN algorithm: Applying the dataset, adjustment of algorithm hyperparameters: Momentum, BatchSize, and Subdivisions; 4. Design of the prototype structure: Component analysis, comparative evaluation of requirements, and modeling in SolidWorks; 5. Construction of the prototype: An 80x80 cm metal frame box, white acrylic plastic cover, two 20-liter containers, wooden base, structural integration, mechanical system (Nema 23 motor, TB6600 driver), and Arduino Mega 2560; 6. Implementation of the CNN in the container: Exporting the trained weights of YOLOv9 to the Raspberry Pi 4 Model B embedded system running in a Python environment; and 7. Validation: MSW samples from real-world environments. The results obtained were a custom dataset of 1883 images in two classes, organic and inorganic, with a resolution of 640x640 pixels. Regarding the training models, models from the YOLO architecture were selected, specifically YOLOv9 tiny, with an accuracy exceeding 90%. For

the design, a 3D model was created, specifying the scaled dimensions of the proposed prototype. Consequently, the metal structure of the container was designed and assembled following a linear development approach, along with its mechanical structure featuring rotating containers for waste deposition; materials such as metal, acrylic, aluminum, and wood were used for this, in addition to a pillow block bearing. The electronic circuit consisted of a Raspberry Pi 4 Model B, a camera for inference, a rotating system comprising a Nema 23 stepper motor, a TB6600 driver, an Arduino Mega 2560, and an SG90 servomotor. The trained YOLOv9 tiny weights were also implemented on the Raspberry Pi 4 Model B embedded system running in a Python environment. Finally, a 120 cm² smart cubic classifying container was obtained for efficient management that aids in MSW separation and environmental care.

Keywords: Municipal Solid Waste, Convolutional Neural Networks, Container, Embedded Systems, Artificial Intelligence.

Capítulo I. Introducción

La gestión de RSU es un desafío importante por su alta tasa de generación, en el año 2020 se estimó que la cantidad total generada a nivel mundial fue de 2.24 millones de toneladas. Esto debido al rápido crecimiento de la población y el avance continuo de la urbanización. Las proyecciones para el año 2050 indican un aumento del 73% en comparación con los presentados en 2020, alcanzando un total estimado de 3.88 mil millones de toneladas (Jin & Li, 2025). El reconocimiento y la clasificación de imágenes de RSU ofrece la oportunidad de identificarlos y catalogarlos, mediante técnicas de visión por computadora, optimizando los procesos de gestión inteligente y eficaz para mantener la calidad del entorno urbano, mejorar la calidad de vida de los ciudadanos y contribuir a lograr los objetivos del desarrollo sostenible (Zhou et al., 2024). Por otra parte, la tecnología moderna para la identificación de objetos mejora su detección, por ejemplo, los algoritmos de aprendizaje profundo livianos con tamaño reducido alcanzan una precisión similar a los modelos que requieren mayor costo y requisitos. Las CNN también tienen aplicación en la Inteligencia Artificial (IA) mediante la resolución de diferentes tipos de problemas (Martín, 2022). Facilitan la clasificación y reciclaje de RSU de manera eficiente y efectiva, empleando técnicas de aprendizaje automático para identificar el tipo de residuos: plástico, metal, papel, entre otros materiales (Chen et al., 2022). Estos han logrado avances notables, que se destacan por su tamaño compacto y mayor velocidad sin disminuir su precisión proporcionando un respaldo teórico y técnico, orientado a dispositivos integrados para lograr niveles superiores en la eficiencia de clasificación (Carolis et al., 2020; Falaschetti et al., 2024; He et al., 2016). El uso de contenedores inteligentes puede mejorar la efectividad y evitar los riesgos a la salud en la gestión manual de residuos, mediante el uso de sistemas embebidos y visión artificial Gude

et al. (2024) mejorando su clasificación y categorización de residuos en tiempo real. Los avances en aprendizaje profundo y visión por computadora han ofrecido una vía prometedora para mejorar el método de clasificación, aportando beneficios sociales y ambientales. Actualmente este proceso aun presenta desafíos debido a la naturaleza compleja y diversa de los residuos, además de la baja disponibilidad de esa variedad como datos para entrenamiento. Las Redes Neuronales Convolucionales (CNN) son modelos de aprendizaje profundo, enfocados al procesamiento de imágenes y la extracción jerárquica de características, además del modelado de la localidad espacial. Constan de múltiples capas convolucionales y completamente conectadas, facilitando la extracción de características más detalladas y una mejor interpretación de la imagen, capturando a su vez una mayor información del contexto global, comprendiendo la estructura general y semántica del modelo para aprender su significado, algunas de las estructuras de red más profundas son ResNet, Inception, EfficientNet, ShuffleNet y YOLO (Zhou et al., 2024). ResNet-50, arquitectura de CNN profunda enfocada a solucionar problemas como degradación del rendimiento, dificultad de aprendizaje y problemas de convergencia con aumento de capas. Aplicando conexiones residuales para aprender de mapeos y extraer características detalladas de las imágenes, consta de capas convolucionales, de normalización por lotes, de agrupamiento y capas completamente conectadas (L. Wang et al., 2024). ShuffleNet v2 se centra en el diseño de redes más eficientes, optimizando el uso de la memoria y la velocidad de red, manteniendo el mismo número de canales de entrada y salida de la capa convolucional para minimizar el costo de acceso a la memoria, reduce el uso de convoluciones grupales para ahorrar recursos, simplifica la estructura de ramas de la red mejorando la operación paralela, y disminuye las operaciones a nivel de elementos para mejorar la ejecución de la red (Chen et al., 2022). You Only Look Once (YOLO) es un modelo de detección de objetos

diseñado para el procesamiento de imágenes en tiempo real, analiza la imagen una vez mientras se desplaza, capaz de detectar y clasificar objetos mediante un entrenamiento previo basado en un conjunto de datos etiquetado y catalogado, superando a las CNN tradicionales en cuanto a eficiencia para la detección de objetos en tiempo real, utilizando solo una CNN para predecir las cajas delimitadoras y probabilidades de clase simultáneamente. Además de contar con la rapidez en la detección en cuadros, razona de manera global, posee un rápido aprendizaje para identificar y generalizar las características de los objetos (C et al., 2019). De acuerdo con ultralytics YOLO trabaja en la detección de objetos como un único problema de regresión, predicen los recuadros delimitadores y las probabilidades de clase a partir de imágenes completas en una sola evaluación, volviéndolos más rápidos que los detectores de dos etapas anteriores, manteniendo al mismo tiempo una gran precisión, además de permitir una rápida exportación, implementación y ejecución en dispositivos Raspberry pi. Esta investigación se centró en el desarrollo de un contenedor clasificador de RSU usando CNN YOLO v9 tiny, mediante su clasificación categórica orgánica e inorgánica con el uso de motores a paso, integrados en cajones internos del contenedor controlados con un minicomputador Raspberry Pi 4 modelo B y cámara para la obtención de datos. El sistema mecánico constó de un sistema electrónico Arduino mega 2560, un motor a pasos Nema 23, driver TB6600 y dos servomotores. El modelo entrenado obtuvo una precisión del 92 %, logrando clasificar con una alta exactitud.

Capítulo II. Antecedentes

El crecimiento de la población en el mundo ha sido constante, lo que implica un aumento en la urbanización que trae consigo una excesiva generación de RSU difícil de tratar. En la última década la eliminación diaria de RSU por persona pasó de 0.5 kg a 1.7 kg, su mala gestión, provoca graves problemas ambientales, como la contaminación de aguas subterráneas, zonas de vertederos clandestinos, contaminación del aire, así como al deterioro estético de los paisajes naturales, urbanos y rurales (Singh, 2019).

Así como el impacto en la salud de la población, como enfermedades entéricas, tifus, cólera, hepatitis, cisticercosis, triquinosis, leptospirosis, toxoplasmosis, sarnas, micosis, rabia, salmonelosis, entre otras, dependiendo de las condiciones locales (Rondón Toro et al., 2016).

A nivel global aproximadamente el 40% de los RSU se eliminan en vertederos, solo el 19% se somete a recuperación de materiales a través de reciclaje y compostaje, 4% y 11% se trata a través de incineración moderna, y 33% todavía se vierten al aire libre (Kaza et al., 2018).

En México se generan diariamente 102,895.00 toneladas de RSU, de los cuales se recolectan 83.93% y se disponen en sitios de disposición final 78.54%, y se reciclan únicamente el 9.63%. La recolección y clasificación es muy baja comparada con otros países, ya que por mala administración terminan llegando a rellenos sanitarios mal administrados, provocando contaminación que generan gases de efecto invernadero y afectan al subsuelo (SEMARNAT, 2017).

En la actualidad, se han alcanzado avances notables en la tecnología con aplicaciones de modelos de aprendizaje profundo livianos, los cuales destacan por su tamaño reducido y mayor velocidad, sin sacrificar la precisión. Estos avances proporcionan un respaldo teórico y técnico, orientado a dispositivos integrados, para lograr niveles superiores de eficiencia en

la clasificación de RSU. La aplicación de grandes modelos de redes en tareas de visión por computadora es notable, como AlexNet (Qiu et al., 2024), GoogLeNet (Yuesheng et al., 2021) y ResNet50 (Zhou et al., 2024).

2.1 A nivel global

En china, Wu et al. (2021) quienes diseñaron un modelo llamado GC-YOLOv5 de clasificación de baterías, cáscaras de naranja, papel usado, vasos y botellas de papel. Consideraron 642 imágenes en su entrenamiento y 40 para validación, alcanzaron una precisión del 99.86% después de 500 épocas, concretando su aplicación con interfaz JavaScript.

De igual forma en china Zeng et al. (2020) desarrollaron el algoritmo PublicGarbageNet con una precisión final del 96.35%, que clasifica cuatro categorías: residuos de cocina, reciclables, desechos peligrosos, y otros tipos; enfocado a multitarea basado en la arquitectura CNN, entrenado con un conjunto de datos públicos de 10,624 imágenes, 10 clases (residuos de cocina, plástico, papel, productos electrónicos, metal, cristal, papel y plásticos no reciclables, textiles y desechos peligrosos). Mejorando su rendimiento optimizando el backbone, aumento de datos, ajuste de hiperparámetros y suavizado de etiquetas.

En ese mismo país Yang et al. (2022) plantearon un algoritmo de clasificación de residuos resistente a la baja precisión causada por la interferencia de luz y sombra, con capacidad adaptativa en la iluminación de imágenes utilizando un método de reemplazo de umbral para disminuir el ruido de sombra, además del operador Canny para ayudar a recortar el fondo blanco en la imagen. El algoritmo se optimizó con base al modelo MLH-CNN logrando una precisión de 96.77% con el conjunto de datos construido y un 93.72% con TrashNet.

Similarmente en el mismo país, (Fu et al., 2021) generaron un sistema inteligente de clasificación de residuos basado en aprendizaje profundo GNet mediante Linux aplicado en un Raspberry Pi 4B como placa principal, además del hardware integrado de un servo de dos grados de libertad, panel táctil y una cámara. El modelo fue entrenado con el conjunto de datos Huawei Garbage Classification Challenge Cup obtuvo una precisión de 92.62%, eficiencia de 0.63s complementado con el desarrollo de una interfaz gráfica de usuario basada en Python y QT.

Mientras que, en Polonia Majchrowska et al. (2022) presentaron un análisis crítico de 11 conjuntos de datos existentes de residuos, recopilando y resumiendo estudios previos sobre los resultados de los experimentos de cada autor. Además, diseñaron dos conjuntos de datos de referencia, con la combinación de detect-waste y classify-waste y todas las categorías identificadas del estudio. Por último, presentaron un detector de dos etapas para la localización de clasificación de residuo, utilizando EfficientDet-D2 para localizarlos y EfficientNet-B2 para clasificarlos mediante un entrenamiento semisupervisado, utilizando imágenes no etiquetadas, logrando un 70% de precisión promedio en la localización de residuos y 75% en la clasificación.

En Indonesia, Laksono et al. (2024) quienes presentaron un modelo para clasificar envases inorgánicos en los sitios de procesamiento de residuos; compararon cinco arquitecturas de CNN: Xception, Inception V3, ResNet-50, ResNet-50 V2 y DenseNet-201; previamente entrenadas con un conjunto de datos de plástico HDPE, PET, botellas y frascos de vidrio, latas, cartón y envases de plástico flexible, recopilados en tres diferentes recursos de Kaggle: conjunto de datos de imágenes de botellas de agua, clasificación de residuos y clasificación de residuos en 12 clases. Para el entrenamiento consideraron aumento de datos, re escalado,

rotación, desplazamiento, inclinación, zoom y volteo, obteniendo un modelo CNN basado en la arquitectura DenseNet-201 con una precisión del 0.96%.

En Arabia Saudita, (Rahman et al., 2022) proponen un modelo eficiente de clasificación de los desechos digeribles y no digeribles utilizando una CNN, un paradigma popular de aprendizaje profundo con una metodología que combina la clasificación de residuos con una red neuronal convolucional mediante el diseño de cajas de basura inteligentes a través de IoT para monitorear datos en tiempo real, y una eficiencia en la separación del 95.31%.

También en Reino Unido proponen un modelo de clasificación de imágenes de RSU, DenseNet 169 basado en el aprendizaje por transparencia, utilizando una metodología por tres etapas: 1ra. define el conjunto de datos NWNUTRASH con una pequeña colección de imágenes de residuos reciclables, incluidos vidrio, papel, cartón, plástico y metal, y 2528 fotografías; 2da. un modelo de clasificación de imágenes residuales basado en aprendizaje por transferencia con CNN; y en la última etapa aplican el modelo de DenseNet que realiza principalmente la conexión densa de la red neuronal a través del Bloque Denso. Finalmente aplican métricas de rendimiento del modelo para evaluar su desempeño relevante en la clasificación de RSU con matrices de confusión, curva de característica operativa del receptor (ROC) y área bajo la curva (AUC), exactitud, precisión, recuperación y puntuación F1. Los resultados experimentales demostraron que el modelo de clasificación de imágenes logro una precisión de aproximadamente el 82.8%, superando significativamente a los modelos de clasificación de imágenes como AlexNet, GoogLeNet y VGG (Q. Zhang et al., 2021).

En ese mismo país, también proponen un sistema inteligente de clasificación de residuos, capaz de alcanzar una precisión del 87% en un conjunto de datos, mediante un método de aumento de datos se desarrolló con base al modelo previamente entrenado ResNet-50,

garantizando que el proceso de separación sea más rápido e inteligente, reduciendo la participación humana (Adedeji & Wang, 2019).

2.2 En América

En Estados Unidos realizaron una revisión que ofrece una visión general referente de cómo abordar los problemas de incertidumbre relacionados con la eliminación RSU. El estudio informa la importancia en la ubicación y la selección de una tecnología de tratamiento en el diseño de un sistema óptimo para el procesamiento de residuos agrícolas vegetativos (Ajay, 2019). Asimismo, presentan una revisión detallada de la arquitectura y los principios de CNN 1D, con un enfoque de ingeniería en sus aplicaciones. Utilizando la generación de diagramas de sigmoide, tanh, ReLU, Leaky ReLU, PReLU, ELU, swish y mish. Identificando la conexión local, el peso compartido y la reducción de la dimensionalidad del muestreo como ventajas de las CNN (Li et al., 2022).

Por otro lado, en Canadá Kannangara et al. (2018) desarrollaron modelos para una predicción precisa en la generación y desviación de Residuos Sólidos Municipales (RSM), basados en variables demográficas, socioeconómicas y la aplicación planificada de la generación de inventarios de RSM en 220 municipios de la provincia de Ontario, Canadá. Los modelos se generaron mapeando cantidades de RSM residenciales e implementando algoritmos de aprendizaje automático, teniendo como resultado modelos de residuos con un buen modelo de predicción, describiendo el 72% de variación en los datos.

En Colombia construyeron el primer prototipo de punto ecológico inteligente que asista al usuario en la correcta clasificación de RSU en punto de origen utilizando Deep Learning, aportando nuevas tecnologías que estén enfocadas en mitigar el impacto climático. Implementando la metodología LEAN de Toyota y la herramienta Solid Works para diseñar

un prototipo acorde a las necesidades de un punto ecológico, además de un análisis estructural y pruebas de peso con el fin de medir su capacidad de resistencia. El modelo de clasificación es una arquitectura de bajo costo computacional que considera MobileNet con una precisión del 97.91% en la validación y un coeficiente kappa de 0.95 (Pardo Burbano et al., 2021).

Pascasio et al. (2022) desarrollaron un prototipo clasificador de cesto inteligente con redes neuronales convolucionales, aplicando un conjunto de 2527 imágenes para el entrenamiento divididos en 6 clases (papel, cartón, plástico, metal, vidrio y otros). Se probaron 3 modelos de redes neuronales y de las cuales se obtuvieron resultados como VGG16 con 88% de precisión, ResNet50 con 91% de precisión, Inceptionv2 con 80% de precisión y MobileNet con 64% de precisión.

2.3 En México

En Guadalajara (García et al., 2015) diseñaron un Separador Inteligente de Residuos (IWS) que reemplaza la forma tradicional de tratar los residuos; El dispositivo propuesto recibe los residuos entrantes y los coloca automáticamente en diferentes contenedores mediante el uso de un procesador multimedia integrado, procesamiento de imágenes y aprendizaje automático para seleccionar y separar los residuos. El experimento logró una eficiencia del 98,33% utilizando el algoritmo k-NN con $k = 3$. El IWS tiene algunas limitaciones en la etapa actual de creación de prototipos. A) El IWS sólo puede separar latas de aluminio, botellas de plástico y cubiertos de plástico. Pero si es deseable que el sistema separe otros tipos de residuos, se puede entrenar fácilmente para que lo haga. B) El IWS sólo puede procesar un residuo a la vez. Por lo tanto, el usuario debe depositar una pieza a la vez debido a la limitación de segmentación. C) El IWS no puede identificar residuos deformados porque la identificación del objeto depende de la forma de los residuos.

En Michoacán (Jiménez et al., 2021), diseñaron un dispositivo clasificador de residuos como metal, plástico y vidrio, se diseñó utilizando la herramienta SolidWorks apoyándose de un boceto de diseño mecánico, el dispositivo realiza la clasificación por medio de sensores no inductivos y capacitivos. Por medio de la ponderación de aciertos en una tabla de registro, el dispositivo contó con una precisión del 80% para plásticos, un 90% para metales, y un 80% para vidrios.

2.4 Conceptos

Lenguajes de programación: considerados como una habilidad importante muy utilizada actualmente en la sociedad. Los cuales requieren de un conocimiento experto como lógica, sintaxis y semántica en el lenguaje elegido (Perera et al., 2021).

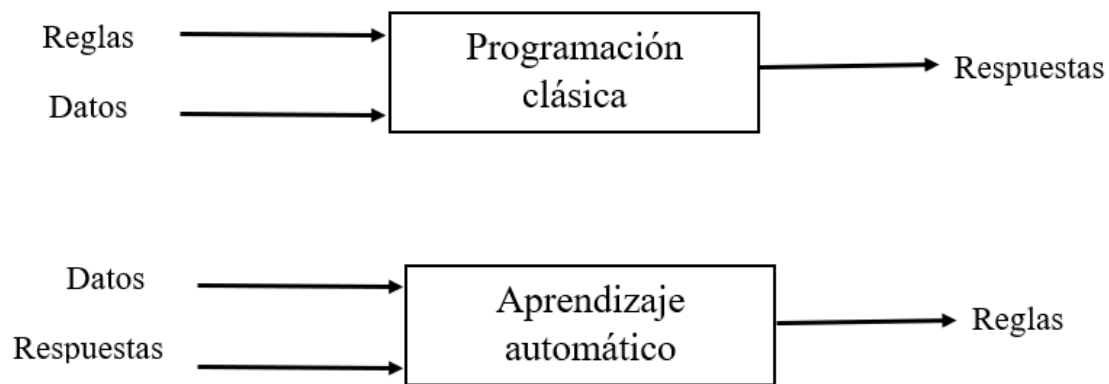
La programación es parte esencial de muchas áreas científicas, como la ingeniería, el análisis de datos y el aprendizaje profundo, estos campos están en rápido crecimiento y sus conceptos se emplean para diversos fines, por ejemplo, obtener información a partir de grandes conjuntos de datos o ajustar modelos de predicción precisos. En los últimos años, Python se ha convertido en el lenguaje de programación elegido para las áreas de investigación, mencionadas anteriormente, no sólo porque es fácil de usar, sino porque, también existe un buen apoyo de la comunidad. Además, de ser un lenguaje de programación interpretado, multiplataforma, alto nivel en la legibilidad del código, se encuentra disponible una gran cantidad de bibliotecas de alta calidad y garantiza el soporte para cualquier tipo de cálculo científico (Perera et al., 2021). Estas características hacen de Python una herramienta adecuada para muchos proyectos industriales y de investigación compleja (Blank & Deb, 2020).

Inteligencia Artificial (IA): La tecnología de IA se ocupa del diseño de sistemas informáticos y programas capaces de imitar rasgos humanos como la resolución de problemas, el aprendizaje, la percepción, la comprensión, el razonamiento y la conciencia del entorno (Abdallah et al., 2020).

La IA es un campo general que abarca el aprendizaje automático (Machine Learning) y aprendizaje profundo (Deep Learning), ha demostrado ser adecuada para resolver problemas lógicos bien definidos, como jugar ajedrez, sin embargo, determinar reglas explícitas para resolver problemas más complejos y difusos resultó más difícil, surgiendo un nuevo enfoque para ocupar el lugar de la IA simbólica el nuevo aprendizaje automático (Chollet, 2021). En la programación clásica, el paradigma de la IA simbólica se ingresan reglas o instrucciones y los datos que se procesaran dan respuestas como en la Figura 1.

Figura 1

Nuevo paradigma de programación



Nota: Adaptado de Deep Learning with Python (Chollet, 2021).

Aprendizaje Automático (Machine Learning): El aprendizaje automático lleva un proceso de entrenamiento en lugar de programarse estáticamente al brindarle ejemplos de una tarea específica, encuentra una estructura estadística en esos ejemplos y con el tiempo, el sistema elaborara reglas que permitan automatizar la tarea. A diferencia de la estadística, el aprendizaje automático tiende a tratar con conjuntos de datos grandes y complejos (como un conjunto de datos de millones de imágenes, cada una de las cuales consta de decenas de miles de píxeles) para los cuales el análisis estadístico clásico, como el análisis bayesiano, sería poco práctico. Como resultado, el aprendizaje automático, y especialmente el aprendizaje profundo, exhibe comparativamente poca teoría matemática y está orientado a la ingeniería (Chollet, 2021).

Aprendizaje profundo (Deep Learning): El aprendizaje profundo es un subcampo específico del aprendizaje automático, es una forma de aprender las representaciones a partir de datos centrados en el aprendizaje de capas sucesivas con una representación cada vez más significativa. El término “profundo” está relacionado por capas sucesivas de representaciones, debido a la cantidad de capas de datos que lo constituyen, se conoce como profundidad del modelo. Este tipo de modelos implica una gran cantidad de capas sucesivas de representaciones y todas aprenden automáticamente a partir de la exposición de los datos de entrenamiento. Sus representaciones en capas en su mayoría aprenden a través de modelos llamados redes neuronales contando con estructuras de capas literales apiladas una sobre otra (Chollet, 2021).

You Only Look Once (YOLO): Este modelo plantea la detección de objetos como un problema de regresión única, que va directamente de los píxeles de la imagen a las coordenadas del cuadro delimitador y probabilidades de clase, solo recibe la imagen y se

observa una vez para predecir objetos presentes y su ubicación. Al realizar una sola red neuronal convolucional predice simultáneamente múltiples cuadros delimitadores y probabilidades de clase para esos cuadros. YOLO se entrena con imágenes completas y optimiza el rendimiento de detección, presenta ventajas referentes a otros métodos tradicionales de detección de objetos, como la velocidad de detección, esto se debe a su propuesta de realizar la detección como un problema de regresión, además del procesamiento de transmisión de video en tiempo real con menos de 25 milisegundos de latencia (Redmon et al., 2015).

YOLOv4 tiny: Considerado un modelo liviano, diseñado a partir de YOLOv4, emplea menos memoria y funciona más rápido, lo que incrementa significativamente la viabilidad de implementarlo en equipos inteligentes (Liu et al., 2022).

YOLOv7 tiny: Versión ligera de YOLOv7, destaca por su velocidad y eficiencia con 6.2M de parámetros y 13.8B de operaciones (FLOPs). Es especialmente efectivo en la detección de objetos de diferentes tamaños, mostrando un rendimiento de 18.8% para objetos pequeños, 42.4% para objetos medianos y 51.9% para objetos grandes (Ultralytics, 2025b).

YOLOv8 Nano: Modelo ligero diseñado para ser rápido y eficiente, especialmente adecuado para dispositivos con recursos computacionales limitados, solo considera 3.2M de parámetros y 8.7B de FLOPS. Optimizado para funcionar en CPU, además su arquitectura más pequeña lo hace ideal para implementaciones en dispositivos Edge mientras mantiene un buen rendimiento en la detección de objetos (Ultralytics, 2025b).

YOLOv9 tiny: Versión simplificada de YOLOv9, diseñada para aplicaciones con recursos limitados cuenta con 2.0M de parámetros y 7.7B de FLOPS, tiempo de inferencia de 2.3ms en GPU T4 con TensorRT10. Esta versión es particularmente útil para el despliegue en

dispositivos edge y aplicaciones móviles donde los recursos computacionales son limitados, sin sacrificar significativamente la precisión (Ultralytics, 2025b).

Algoritmo: Procedimiento computacional bien definido que toma algún valor o conjunto de valores como entrada y produce algún dato o conjunto de datos como salida (Cormen et al., 2009).

Red neuronal artificial: Intentan emular el cerebro humano, combinando informática y estadística para resolver problemas comunes en el campo de la IA, su nombre y estructura están inspirados en el cerebro humano imitando la forma en que las neuronas biológicas se transmiten entre sí (IBM, 2025).

Redes neuronales de propagación hacia delante o perceptrones multicapa (MLP): Se componen de una capa de entrada, una o varias ocultas y una capa de salida compuestas por neuronas sigmoides, no perceptrones, ya que la mayoría de los problemas del mundo real no son lineales. Por lo general, los datos se introducen en estos modelos para entrenarlos, y son la base de la visión por computadora, el procesamiento natural del lenguaje y otras redes neuronales (IBM, 2025).

Redes neuronales convolucionales (CNN): Se utilizan para el reconocimiento de imágenes, el reconocimiento de patrones y/o la visión por computadora. Estas redes aprovechan los principios del álgebra lineal, en particular la multiplicación de matrices para identificar patrones dentro de una imagen (IBM, 2025).

Redes neuronales recurrentes (RNN): Estos algoritmos de aprendizaje se aprovechan principalmente cuando se utilizan datos de series de tiempo para hacer predicciones sobre resultados futuros, como predicciones del mercado de valores o pronósticos de ventas (IBM, 2025).

Unidad Central de Proceso (CPU): Construida a base de millones de transistores, se le conoce como el "cerebro de la computadora", ejecuta las órdenes y los procesos que necesitan la computadora y el sistema operativo. La CPU también determina la rapidez a la que pueden ejecutarse los programas en múltiples tareas (Intel, 2025).

Unidad de Procesamiento Gráfico (GPU): Es un procesador formado por muchos núcleos más pequeños y especializados. Diseñada para procesamiento paralelo, la GPU se utiliza en una amplia gama de aplicaciones, incluida la representación de gráficos y videos (Intel, 2025).

Python: Lenguaje de programación con estructuras de datos de alto nivel eficientes y un simple, pero efectivo sistema de programación orientado a objetos (Amazon Web Services, 2025).

Raspberry Pi: Pequeño computador de bajo costo y tamaño compacto con un sistema operativo Linux (Raspberry Pi Foundation, 2025).

TensorFlow: Biblioteca de software para diseñar e implementar cálculos numéricos con un enfoque clave en aplicaciones de aprendizaje automático (TensorFlow, 2025).

Hiperparámetros: Variables de configuración externa que los científicos de datos utilizan para administrar el entrenamiento de modelos de machine learning o deep learning. A veces llamados hiperparámetros de modelos, se configuran de manera manual antes de entrenar un modelo (Ultralytics, 2025a).

Residuo: Material o producto cuyo poseedor desecha, puede encontrarse en estado sólido o semisólido, también puede ser un líquido, gas contenido en recipientes que puede ser susceptible de ser valorizado o requiere sujetarse a tratamiento o disposición (LGPGIR, 2003).

Residuos Sólidos Urbanos: Residuos generados en las casas habitación, resultan de la eliminación de los materiales que utilizan en sus actividades domésticas, productos que consumen y de sus envases, embalajes o empaques; los residuos que provienen de cualquier otra actividad dentro de establecimientos o en la vía pública que genere residuos con características domiciliarias, y los resultantes de la limpieza de las vías y lugares públicos, siempre que no sean considerados como residuos de otra índole (LGPGIR, 2003).

Residuos sólidos urbanos orgánicos: Todo desecho de origen biológico que alguna vez estuvo vivo o fue parte de un ser vivo como alimentos, excedentes de comida, entre otros (SEMARNAT, 2017).

Residuos sólidos urbanos inorgánicos: Todo desecho que no es de origen biológico como vidrio, plástico y metales (SEMARNAT, 2017).

Contenedor de residuos: Recipiente de carga para el depósito y transporte de estos (Recytrans S.L., 2025).

Contenedor Clasificador: Contenedores de residuos pintados de diferentes colores para su fácil identificación del tipo de residuo depositado (Recytrans S.L., 2025).

Contenedor inteligente: Dispositivo de residuos inteligente basado en inteligencia artificial, diseñado para lugares públicos que facilita el reciclaje. Clasifica y comprime automáticamente los residuos, controla el nivel de llenado y procesa datos para una gestión conveniente de los desechos (Bin-e, 2025).

Capítulo III. Justificación

La disposición inadecuada de los RSU es un problema mundial, aunque en proporciones distintas. Estimaciones realizadas reportan que el 70% de los municipios utilizan sitios inadecuados de disposición, mientras que la generación total y per cápita de estos continúa aumentando y las tasas de reciclaje se mantienen en promedio, por debajo de 10%. La Hoja de Ruta indica que alrededor de 145.000t/día toneladas diarias de residuos se siguen destinando a basurales, quema u otras prácticas inadecuadas, lo que equivale a los residuos generados por 170 millones de personas, o 27% de la población regional (UNEP, 2018)

Su recolección debe estar organizada de tal modo que permita un servicio eficiente y equitativo, sin producción de malos olores, polvos, ruidos molestos, desorden y en condiciones aceptables para un servicio de esta naturaleza. En la actualidad, las principales alternativas disponibles de recipientes para depositar los RSU al paso de los camiones de recogida son los recipientes de almacenamiento, las bolsas o sacos desechables y los contenedores con ruedas (Contreras et al., 2016).

Remplazar la forma tradicional de tratar los RSU por una gestión inteligente permitirá optimizar los procesos de recogida y eliminación, proporcionando varias ventajas (Genia, 2025)

- **Recolección eficiente de RSU:** Los sistemas de gestión inteligente utilizan sensores y dispositivos conectados para controlar los niveles de RSU en los contenedores. La obtención de datos en tiempo real permite optimizar las rutas de recogida, reduciendo los viajes innecesarios y ahorro de combustible.

- **Reducción de la huella de carbono:** Al optimizar las rutas de recolección los sistemas inteligentes de gestión de RSU reducen significativamente las emisiones de carbono de vehículos recolectores, contribuyendo a un ambiente más limpio y verde.
- **Disminuir la capacidad de los vertederos:** Al fomentar la separación y reciclaje adecuado de los RSU, se puede reducir significativamente la cantidad de RSU enviados a los vertederos.
- **Asignación efectiva de recursos:** Los sistemas inteligentes de gestión de RSU permite a las autoridades asignar recursos con base en conocimiento de datos generados para la distribución de sistemas de recolección para su mejor tratamiento.
- **Mejora de la salud pública:** La gestión adecuada de los residuos reduce el riesgo de contaminación, olores y propagación de enfermedades asociadas con los residuos no tratados. Esto conduce a una mejor salud pública y una mejor calidad de vida para los ciudadanos.

El correcto tratamiento de los RSU abre la posibilidad de generar nuevos productos, reducir los costos de operación de otros servicios públicos y su vez evita la contaminación del ambiente (Tecmed, 2023). Con el fin de mejorar la gestión de los RSU este prototipo implementa un contenedor clasificador de RSU con el uso de la IA, facilitando la separación y reciclaje en la sociedad. Por otra parte, se contribuye al objetivo 13: acción por el clima de la agenda 2030.

Capítulo IV. Objetivos

4.1 Objetivo general

- Desarrollar un prototipo de contenedor clasificador de RSU con Redes Neuronales Convolucionales (CNN).

4.2 Objetivos específicos

1. Conformer el conjunto de datos (Datasets) de RSU.
2. Analizar los algoritmos avanzados para la clasificación de objetos con CNN.
3. Desarrollar y entrenar el algoritmo CNN de aprendizaje profundo.
4. Diseñar la estructura del prototipo contenedor.
5. Construir el prototipo.
6. Implementar la CNN del contenedor.
7. Validar el prototipo.

Capítulo V. Metodología

5.1 Materiales

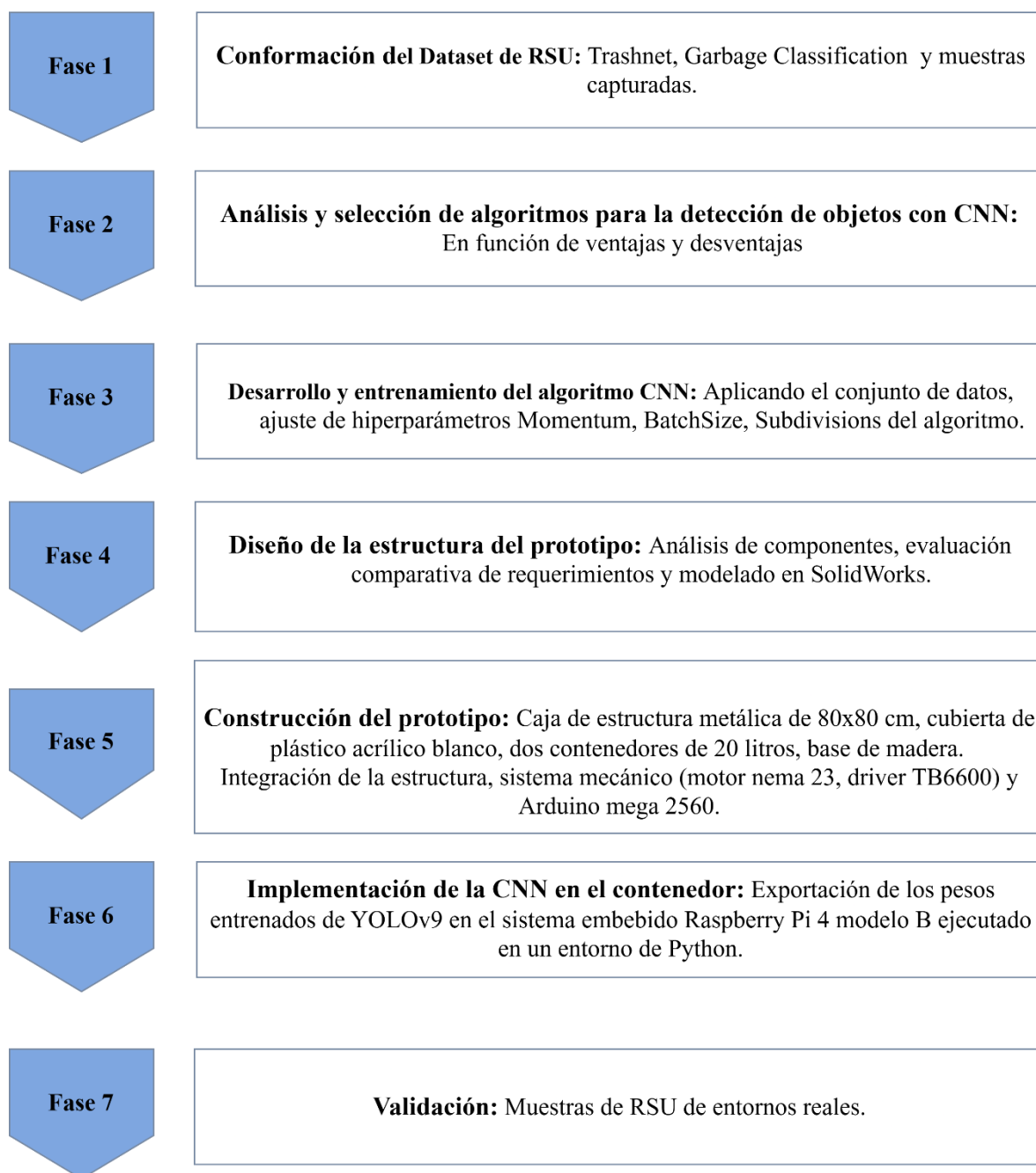
Para este proyecto se utilizaron los materiales: computadora portátil con un procesador i7 13650HX, una gráfica dedicada RTX 4060 8GB VRAM, 32GB RAM. Minicomputador Raspberry Pi 4 modelo B 8gb RAM, Cámara HQ 12MP con lente para enfoque CS, Motor a pasos Nema 23 con controlador TB6600, Servos motores, Python 3.11.0, CudaNN, Pytorch, Dataset de RSU orgánicos e inorgánicos.

5.2 Metodología

En esta investigación se realizó con una metodología cuantitativa-experimental. Cuantitativa por la observación del error y precisión de clasificación de los RSU sin influir en los resultados finales. Experimental por el entrenamiento y validación del modelo de CNN evaluando el desempeño de las épocas de entrenamiento en la detección de objetos de RSU, manipulando cada uno de los hiperparámetros adecuando el rendimiento de los entrenamientos, figura 2.

Figura 2

Proceso metodológico.



5.2.1 Dataset de RSU

La conformación del dataset consistió en un conjunto de datos personalizado al realizar un análisis de las diferentes bases de datos, comparando los tipos de residuos, de imágenes y su cantidad, tabla 1.

Tabla 1

Datasets consultados.

Nombre	No. de categorías	No. de imágenes	Función	Tipo de imágenes	Autor
UAVVaste	1 (residuos de basura en áreas verdes)	772	Segmentación	Capturas de residuos en campo abierto	(Kraft et al., 2021)
Trashnet	6 (cristal, papel, cartón, plástico, metal, basura general)	2527	Clasificación	Fondo limpio o fondo blanco	(Thung & Yang, 2016)
Waste Classification data	2 (orgánicos y objetos reciclables)	~25000	Clasificación	Generadas por búsquedas de google	(Sashaank Sekar, 2019)
Garbage classification	12 (basura, biológico, cartulina, batería, metal, papel, plástico, ropa, vidrio marrón, vidrio blanco, zapatos)	15150	Clasificación y detección	Combinacion de los dataset “clothing dataset” y la herramienta web scrapping	(Mostafa Mohamed, 2020)
TrashBox	7 (residuos médicos, residuos electrónicos,	17785	Clasificación y detección	Generadas por la web	(Kumsetty et al., 2022)

	plásticos, papel, metal, cristal, cartón)				
--	--	--	--	--	--

Se eligieron las imágenes más adecuadas y se separaron en dos categorías orgánico e inorgánico, una herramienta para el etiquetado como es Roboflow delimitando manualmente el área del objeto a detectar, aplicando métodos de preprocesado de imagen, redimensión de imágenes a 640x640 pixeles finalizando con un autoajuste de contraste, así como un aumento de datos y una rotación en las imágenes de $+15^\circ$ y -15° con orientación horizontal y vertical figuras 3, 4 y 5.

Figura 3

Imágenes de TrashNet.



Figura 4

Imágenes de Waste Classification Data.

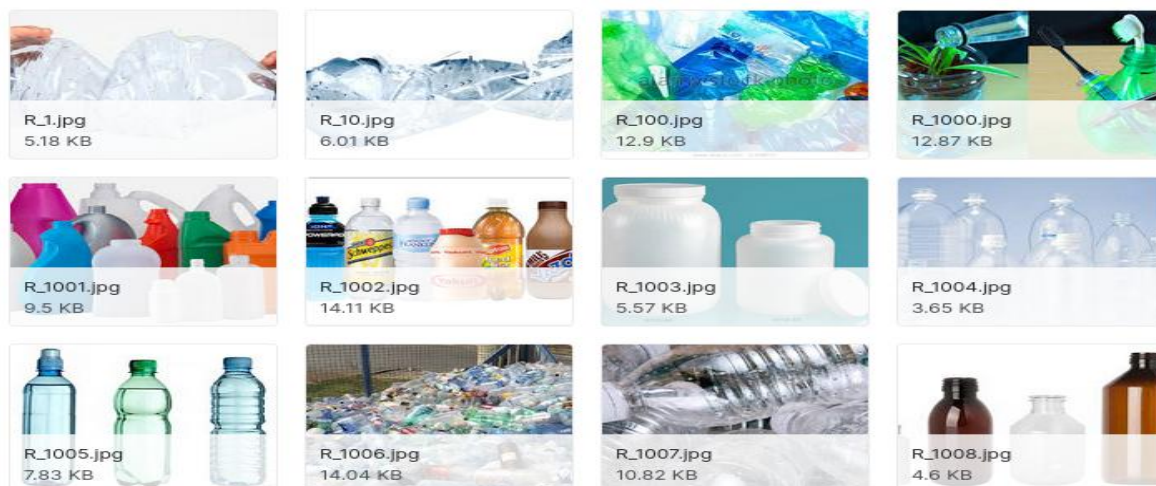


Figura 5

Imágenes de Garbage Classification Dataset.



5.2.2 Algoritmos avanzados para la clasificación de objetos con CNN

En la actualidad existe una gran variedad de modelos desarrollados para la detección y clasificación de objetos, para seleccionar el algoritmo adecuado para la clasificación de objetos con CNN, se realizó una tabla comparativa de las características fundamentales de los algoritmos, enfocándose en las ventajas principales como la precisión, eficiencia computacional o robustez, además de sus desventajas o limitaciones presentadas en los diferentes modelos utilizados en dispositivos de recursos limitados como el Raspberry Pi.

5.2.3 Desarrollo y entrenamiento del algoritmo CNN de aprendizaje profundo

Para entrenar YOLO, la imagen considerada se divide en una cuadrícula de tamaño $S \times S$, donde cada celda genera una región de interés y predice el número de cajas delimitadoras (Bounding Boxes), además de su nivel de confianza en cada una de ellas, y el número de clases del modelo con sus probabilidades. Las predicciones se codificaron en un tensor mediante la ecuación (1), figura 6 (Redmon et al., 2015).

$$S \times S (B \times 5 + C) \quad (1)$$

donde

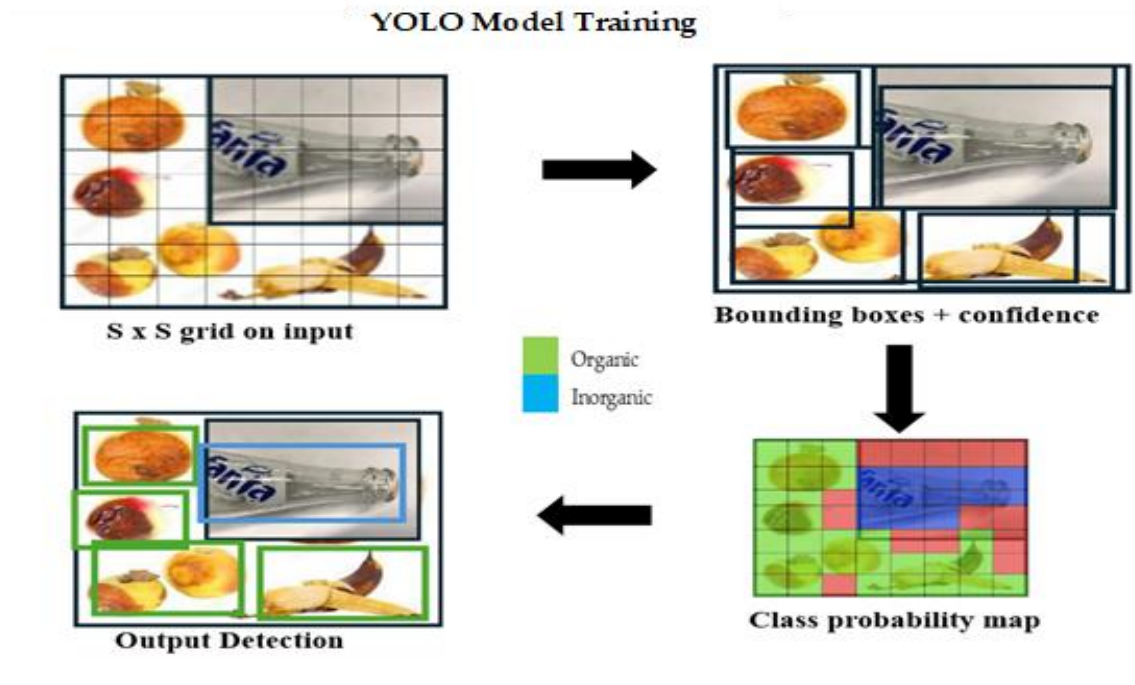
S : número de celdas en la cuadrícula de la imagen ; B : número de cajas delimitadoras;

5: información asociada a cada caja delimitadora (x, y, w, h , confianza)

C : número de clases del modelo;

Figura 6

Representación del entrenamiento del modelo YOLO.



Nota: Adaptado de YOLO: Real-Time Object Detection (Redmon, 2025).

Para el entrenamiento de los modelos se consideraron los hiperparámetros y el conjunto de datos personalizado con 1883 imágenes, 85% para entrenamiento, 15% para pruebas, además de un aumento de datos en línea como flipud: 0.5, fliplr: 0.5 y mosaic: 1.0.

5.2.3.1 YOLOv4 tiny

Para el caso de YOLOv4-Tiny se entrenó con el conjunto de datos personalizado, modificando los hiperparámetros en el archivo yolov4-tiny-custom.cfg, el número de batch=32, 32 imágenes por iteración, subdivisions=16, batch dividido entre 16 (*mini – batches* = 32/16), el cual contuvo dos imágenes por batch para su procesamiento y decay=0.0002 $2e^{-4}$, donde el entrenamiento finaliza al realizar las 6000 iteraciones, Tabla 2.

5.2.3.2 YOLOv7 tiny

Usando la versión más optimizada de CNN YOLOv7-tiny con solo 6.2 Millones de parámetros contra los 36.9 Millones de su versión normal (C.-Y. Wang et al., 2022). Modificando solo el Weight decay: 0.0002 con el mismo dataset, batch: 32 y 125 épocas, Tabla 2.

5.2.3.3 YOLOv8 nano

Versión más optimizada de YOLOv8-nano con solo 3.2 Millones de parámetros, 8.7 billones de FLOPs, logrando un mAP@0.5 en el Dataset COCO. Cuenta con una arquitectura que mejora su velocidad de inferencia sin sacrificar demasiado el rendimiento. Su ajuste en los hiperparámetros fue modificando solo el decay: 0.0002 con el mismo dataset, batch: 32 y 125 épocas, Tabla 2.

5.2.3.4 YOLOv9 tiny

Versión mejorada de YOLOv9-tiny con 2 millones de parámetros y 7.7 GFLOPs, alcanza una precisión mAP@50-95 de 38.3% en el dataset COCO, para el entrenamiento se realizó un ajuste de hiperparámetros como Learning rate con 0.001, enviando un tamaño de batch de 32 con 8 subdivisiones, finalmente en un total de 100 épocas, Tabla 2.

Tabla 2

Modelos entrenados.

Model Settings YOLOv4 tiny		Model Settings YOLOv7 tiny		Model Settings YOLOv8 nano		Model Settings YOLOv9 tiny	
Input	640x640	Input	640x640	Input	640x640	Input	640x640
Learning rate	0.002	Learning rate	0.001	Learning rate	0.001	Learning rate	0.001
Weight decay	0.0002	Optimizer	Adam	Optimizer	Adam	Optimizer	Adam

Optimizer	Adam	Momentum	0.937	Momentum	0.937	Momentum	0.937
Momentum	0.937	Batchsize	32	Batchsize	32	Batchsize	32
Batchsize	32	Subdivisions	8	Subdivisions	8	Subdivisions	8
Subdivisions	8	Total epoch	100	Total epoch	100	Total epoch	100
Total iterations	6000						

5.2.3.5 Métricas de validación

Para evaluar el rendimiento del modelo de clasificación o detección se consideró el número de Verdaderos Positivos (TP), Falsos Positivos (FP), Verdaderos Negativos (TN), Falsos Negativos (FN) obtenidos de las matrices de confusión generadas en la figura 4, y calculados en:

La intersección sobre la unión (IoU)

Cada imagen de entrada contiene cajas delimitadoras (ground-truth), durante el proceso de aprendizaje se generan nuevas cajas delimitadoras predichas. Cuando existe un objeto dentro del área de interés la caja delimitadora generada se superpone con la referenciada, calculando el área de unión con la confianza obtenida, obteniendo una relación entre la zona de intersección y el área de unión, ecuación (2) (Kim et al., 2023).

$$IoU = \frac{TP}{TP+FN+FP} \quad (2)$$

Precisión

Proporción de detecciones correctas entre todas las detecciones realizadas. Un valor alto de precisión indica que el modelo está realizando pocas detecciones falsas, ecuación (3) (Gude et al., 2024).

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

Recall

Mide la proporción de objetos reales que fueron detectados correctamente. Un alto valor de recall indica que el modelo está encontrando la mayoría de los objetos presentes en las imágenes, ecuación (4) (Gao et al., 2024).

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

Gráfica F1-score

De acuerdo con Ultralytics, esta métrica permite observar el equilibrio entre la Precisión (P) y el Recall (R), evaluando más equilibrado el rendimiento de un modelo, identificando si un modelo funciona bien en todas las clases o si está sesgado hacia una clase específica, ecuación (5).

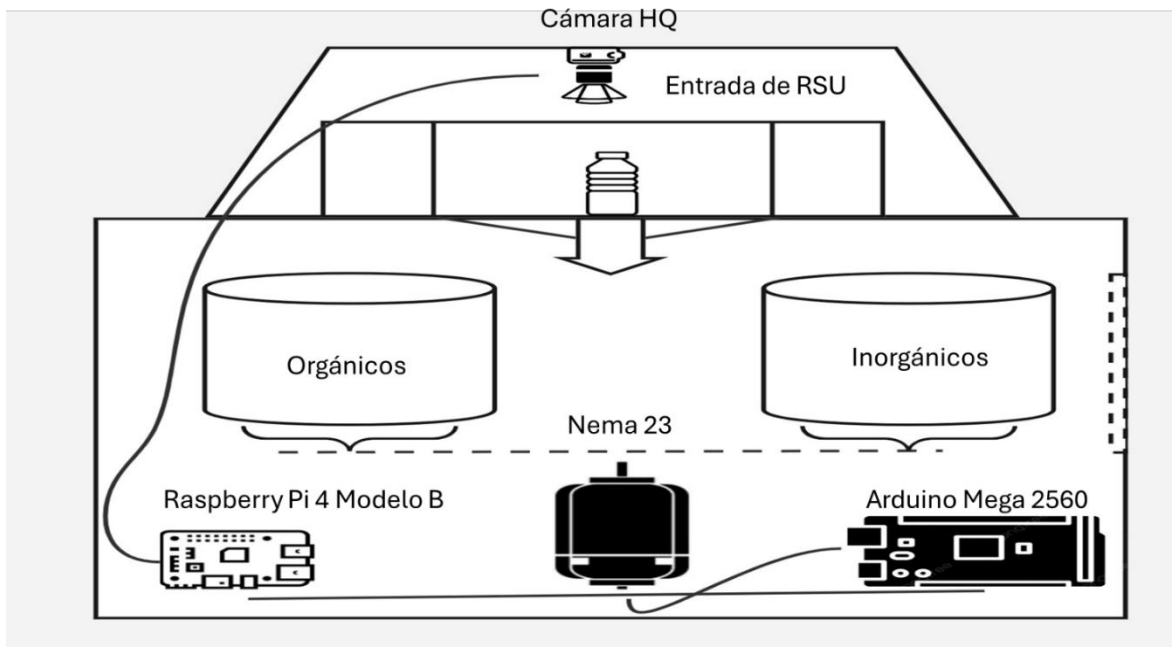
$$F1 - Score = 2 \frac{P * R}{P + R} \quad (5)$$

5.2.4 Diseño de la estructura del prototipo contenedor

Para el diseño esquemático y modelado 3D del contenedor, se consideró una zona de entrada de residuos, sistema mecánico rotatorio por motor, contenedores divididos para las clases y un sistema de detección por cámara utilizando el software SolidWorks, figura 7.

Figura 7

Diseño esquemático del contenedor



El sistema cuenta con una zona de entrada de RSU, el cual tiene una cámara HQ instalada, una compuerta en la parte inferior accionada por servomotores. En la parte interna está instalado un circuito mecánico controlado por un Arduino mega 2560, conformado por un motor nema 23 y un driver TB6600 acoplado a una base para tambos categóricos como orgánicos e inorgánicos. Finalmente, una vez obtenida la imagen se envían los datos al sistema principal Raspberry Pi 4 modelo B que procesa la imagen y envía el control al sistema mecánico y mecanismos de entrada para realizar la clasificación correspondiente.

5.2.5 Construcción del prototipo

La construcción del prototipo consistió en un ensamble con un armazón metálico de 120 cm² soldado con la base del contenedor, un ensamble mecánico interno tomando medidas de los lados frontales, posterior, izquierdo y derecho, realizando cortes de cada medida exacta,

ajustando los ángulos de la estructura antes del ensamble con el armazón. El ajuste de la base requirió dos contenedores circulares de 20 Litros cada uno instalados en una base rectangular de metal de 60 x 20 cm, un motor nema 23 sobre una plataforma de madera 60x60 cm, un acople de metal con un tornillo, ajustando la conexión entre el acople del motor y la base rectangular. Posteriormente, se instaló el motor con los contenedores y se cubrió con paredes de acrílico de 80x80 cm. En la parte superior se instaló una placa de plástico unida a un servomotor MOT-110 en el centro izquierdo (vista exterior). Finalmente, en la parte superior del contenedor se colocó una caja de plástico de 40x40 cm que almacena los circuitos electrónicos y una abertura inferior circular para la cámara, figuras 8-16.

Figura 8

Unión de la estructura mecánica (armazón).



Cortes del material acrílico y base de madera, instalación de los rieles corredera para la base de la estructura de los tambos, figura 14.

Figura 9

Materiales para la estructura interna y recubrimientos de acrílico.



Instalar las placas de acrílico a la estructura metálica, además de combinar la base de los contenedores, figura 15.

Figura 10

Instalación de las cubiertas de acrílico.



Instalación de la base de los contenedores y motor nema 23, atornillando el sistema mecánico rotatorio con la base rectangular, figura 16.

Figura 11

Integración de la base para sistema mecánico.



Instalación de las compuertas, para la extracción de los contenedores, figura 17.

Figura 12

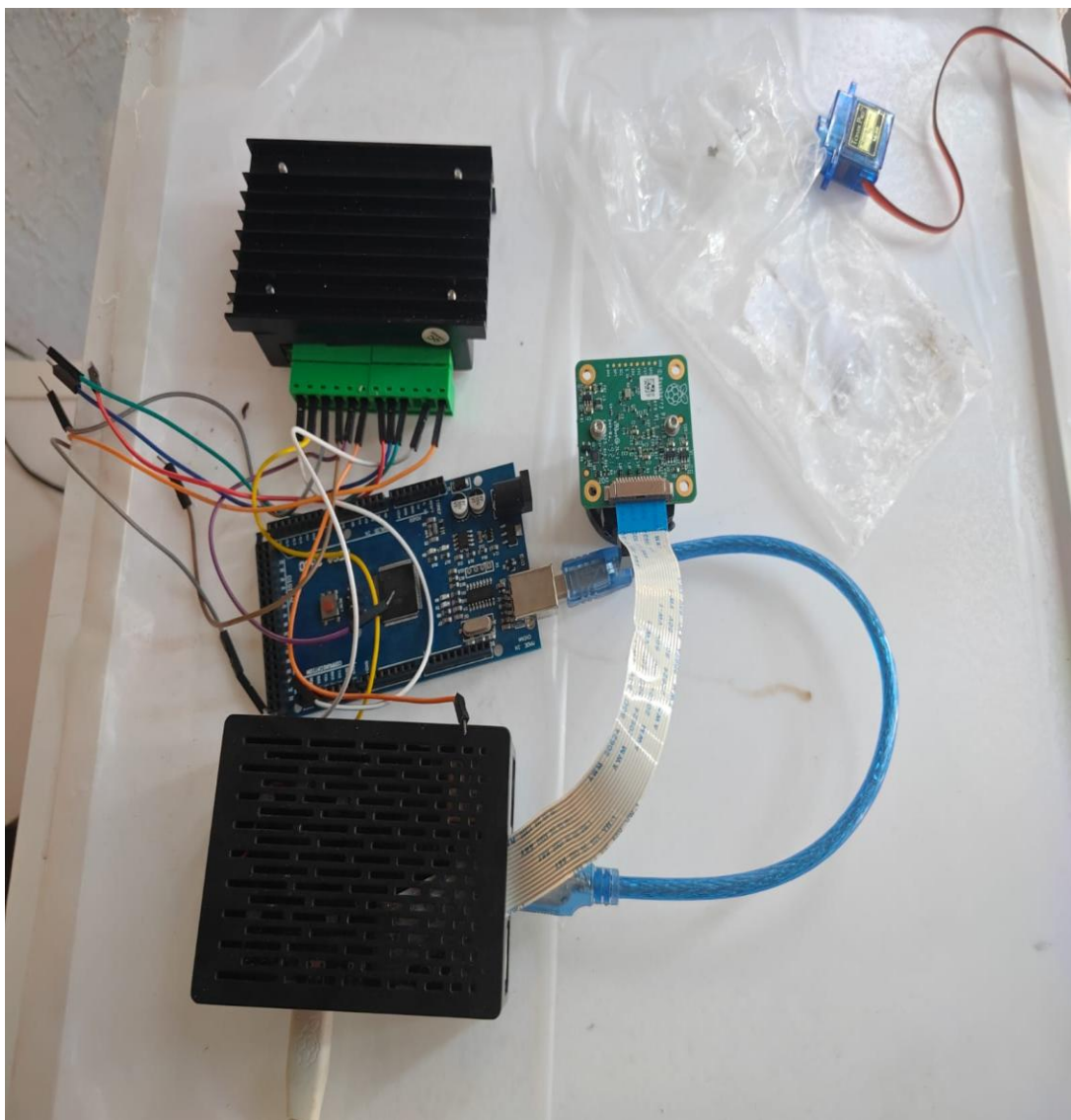
Colocación del sistema mecánico.



Instalación del sistema electrónico, cámara y la visualización en pantalla, 18.

Figura 13

Representación del sistema electrónico.



En este aparatado se realizó la integración del sistema Raspberry Pi 4 Modelo B con el microcontrolador Arduino Mega 2560 y el módulo de cámara en un único sistema electrónico. Estos componentes se colocaron en una carcasa personalizada, diseñada en SolidWorks e impresa en 3D para la correcta disposición y conexión serial entre la Raspberry Pi y el Arduino, figura 19.

Figura 14

Ejecución del sistema electrónico.



Adicionalmente se construyó un actuador de descarga utilizando una base cuadrada de plástico. En el eje central de esta base se acoplará un servomotor SG90 cuyo control estará conectado directamente al Arduino mega 2560. La función del servomotor será cambiar la orientación de la placa de una postura horizontal a una inclinada frontalmente, permitiendo así que cualquier residuo sobre ella se desplace y caiga debido a la pendiente generada, figura 20.

Figura 15

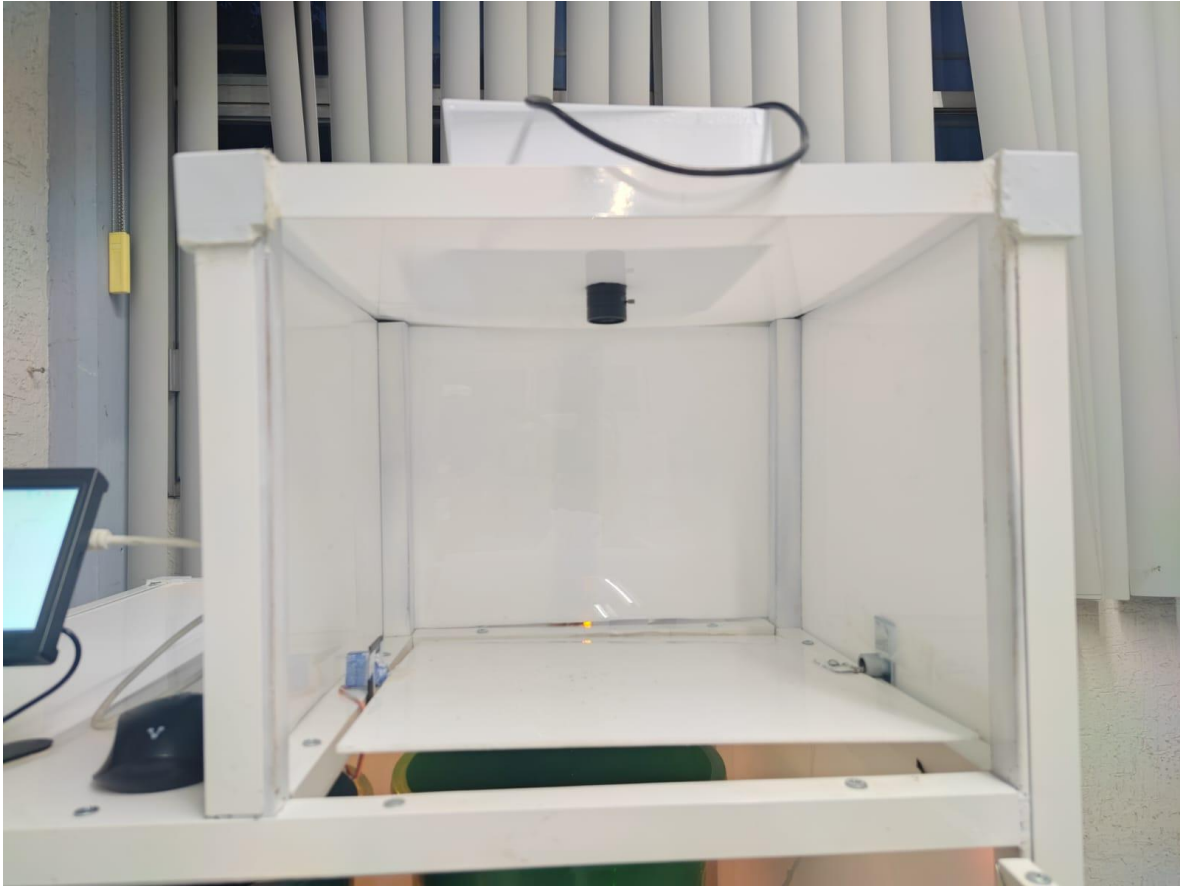
Colocación del cubo de plástico para selección del residuo.



Los componentes se instalaron en la zona de entrada de residuos en la parte superior, integrando la placa de plástico de inclinación, el sistema electrónico y el Raspberry Pi con cámara, figura 21.

Figura 16

Diseño general del sistema de selección.



5.2.6 Implementación de la CNN del contenedor

La implementación consistió en la instalación de un sistema operativo llamado “Raspberry Pi OS”, mediante el uso de una memoria USB boot en el Raspberry Pi 4 modelo B se migran los pesos entrenados generados y resultados del entrenamiento de los modelos, trasladando los archivos con extensión “.pt”, “.yaml”, “.data” y “.cfg” para ser desplegados dentro del dispositivo controlador. Finalmente, el sistema se configura con el código correspondiente para ejecutar el modelo de YOLOv9 tiny en cada prueba.

5.2.7 Validación del prototipo

En esta última fase, se analiza el funcionamiento del contenedor ingresando residuos orgánicos e inorgánicos, iniciando con el sistema controlador Raspberry Pi 4 modelo B, sistema electrónico Arduino mega 2560 y el sistema mecánico nema 23 con driver TB6600, mostrando la inferencia del tipo de residuo y categorizándolo en el recipiente correspondiente.

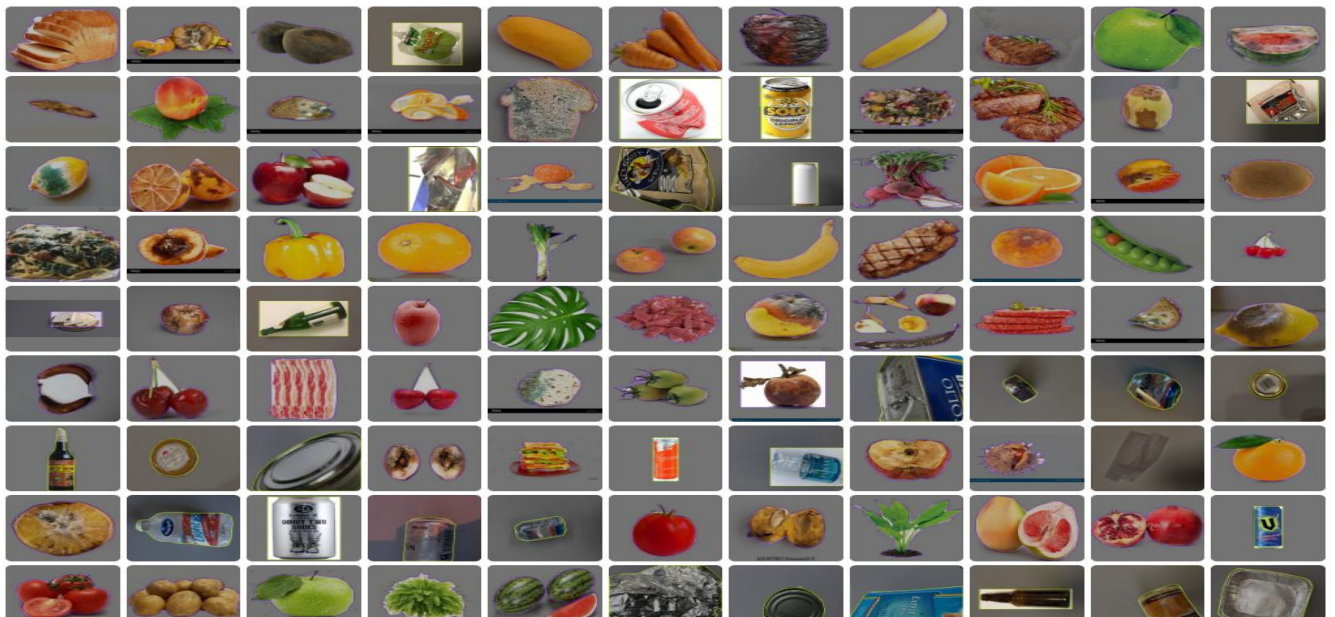
Capítulo VI. Resultados

6.1 Dataset de RSU

De acuerdo con el análisis del conjunto de datos se recopilaron las imágenes más claras y nítidas para subir a la plataforma de Roboflow etiquetando manualmente cada una, delimitando su área específica, generando un conjunto de 1883 imágenes de las cuales 1608 fueron para el entrenamiento y 275 en pruebas, figura 17.

Figura 17

Dataset personalizado de RSU.



6.2 Algoritmos avanzados para la clasificación de objetos con CNN

De acuerdo con el análisis de los modelos YOLO se identificaron sus ventajas y desventajas,

Tabla 3.

Tabla 3

Comparativa de modelos de CNN.

Algoritmo	Ventajas	Desventajas
YOLOv4-tiny , modelo liviano con un aumento en la velocidad de detección y solo una décima parte de los parámetros originales (YOLOv4, 2021).	Rápido rendimiento de detección. Bajo requisito de recursos computacionales. Menor tamaño del modelo.	Menor precisión. Menos clases detectables.
YOLOv8 nano , modelo de última generación, se utiliza en tareas como detección de objetos, clasificación de imágenes y segmentación de instancias (YOLOv8, 2023).	Rápido y eficiente en la detección de objetos. Buen rendimiento en la detección de objetos comunes.	Menor precisión en la detección de objetos pequeños o en condiciones adversas.
YOLOv7-tiny , basado en YOLOv7 con una estructura simplificada diseñada específicamente para GPU de borde de dispositivos (YOLOv7, 2023).	Inferencia más rápida y con mayor precisión en la detección de objetos que sus versiones anteriores. Mas confiable en entornos del mundo real. Facilidad de escalabilidad sobre diferentes tamaños de imagen y conjunto de datos.	Mas complejo que los modelos anteriores (YOLOv5 o inferior) dificulta su implementación y entrenamiento. Mayor requerimiento computacional.

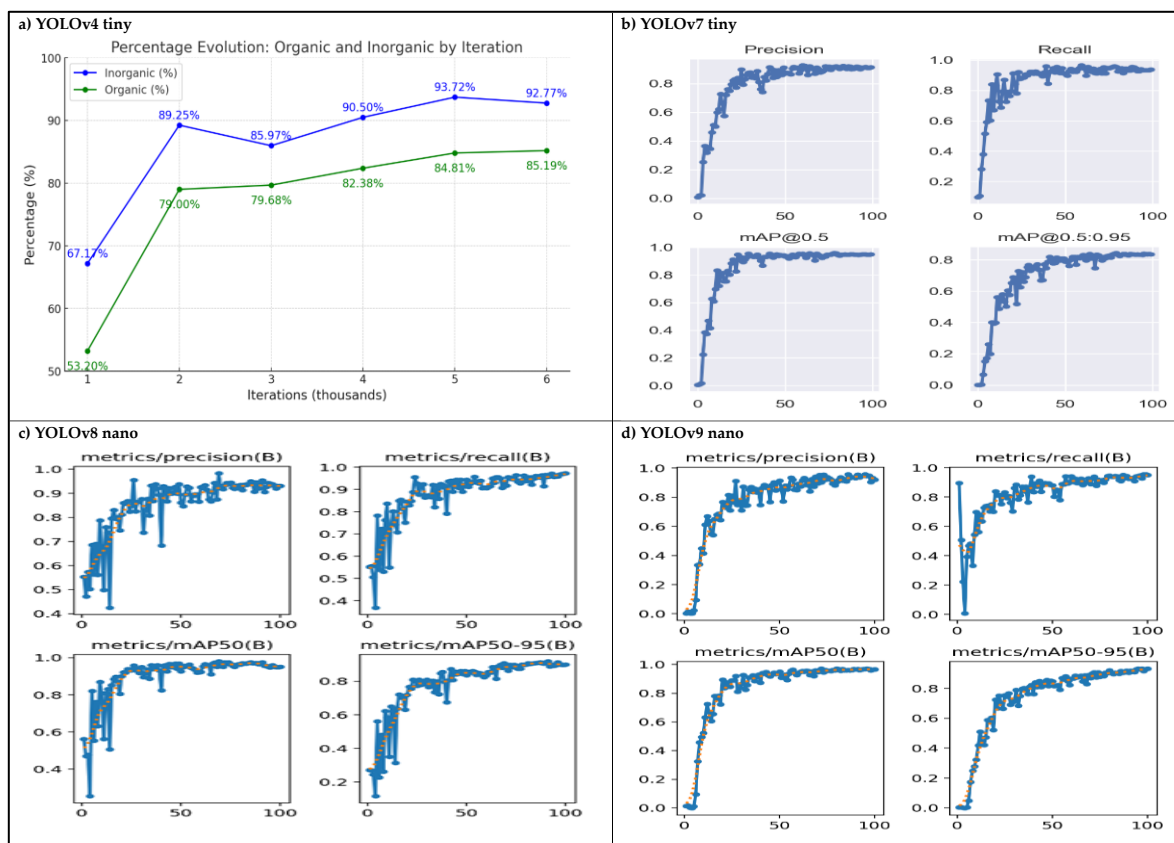
YOLOv9-tiny extrae y procesa características con mayor eficacia, mejorando la capacidad de aprendizaje del modelo, garantiza la retención de información crucial a lo largo de todo el proceso de detección, logrando una alta precisión y rendimiento (YOLOv9, 2024).	Mayor precisión con menos recursos	Mayor complejidad arquitectónica
	Reducción de la pérdida de información	Requisitos de hardware para entrenamiento
	Eficiencia computacional mejorada	Menor velocidad de inferencia en comparación con versiones anteriores

6.3 Desarrollo y entrenamiento del algoritmo CNN de aprendizaje profundo

En el desarrollo y entrenamiento de los modelos, YOLOv4 tiny con 6000 iteraciones el modelo alcanzo 92% de precisión para inorgánicos y 85% para orgánicos; YOLOv7 tiny logro alcanzar una precisión promedio de 91%; YOLOv8 nano alcanzo una exactitud promedio del 94% y YOLOv9 tiny del 92%. Sus métricas indicaron que el comportamiento de las gráficas se estabiliza después de la época 50 para cada uno de los modelos, figura 18.

Figura 18

Rendimiento del entrenamiento de YOLOv4 tiny



Otro aspecto importante se presenta en las matrices de confusión generadas, obtenidas por los entrenamientos y ajustes de hiperparámetros, además se calcular su precisión, recall y F1-Score, figura 19.

Figura 19

Matrices de confusión generadas.

YOLOv4 tiny

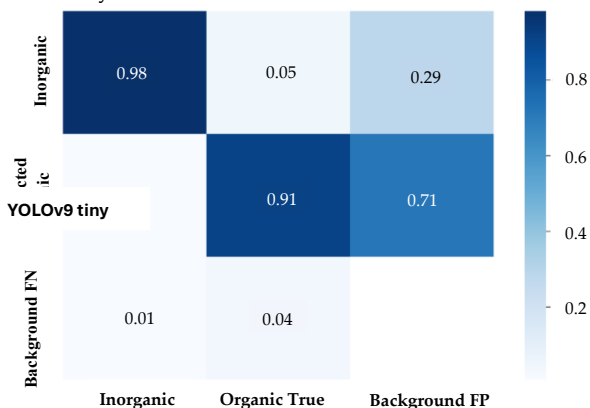
```

calculation mAP (mean average precision)...
Detection layer: 30 - type = 28
Detection layer: 37 - type = 28
128
detections_count = 406, unique_truth_count = 153
class_id = 0, name = inorganic, ap = 94.63% (TP = 76, FP = 23)
class_id = 1, name = organic, ap = 88.79% (TP = 68, FP = 19)

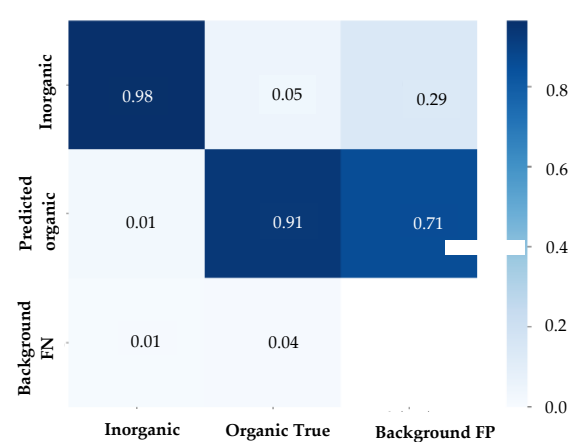
for conf_thresh = 0.25, precision = 0.77, recall = 0.94, F1-score = 0.85
for conf_thresh = 0.25, TP = 144, FP = 42, FN = 9, average IoU = 60.74 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.917109, or 91.71 %
Total Detection Time: 2 Seconds
  
```

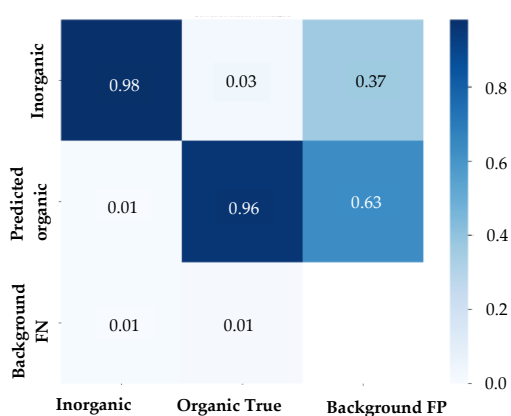
YOLOv7 tiny



YOLOv9 nano



YOLOv8 nano



Cada modelo registró sus parámetros obtenidos, comparando el rendimiento individual, ventajas y desventajas, desempeño general, equilibrio entre eficiencia y precisión, Tabla 4.

Tabla 4*Cálculos de los modelos YOLO.*

Model	Precision	Recall	F1 – Score	IoU
YOLOv4 tiny	0.91	0.94	0.85	0.60
YOLOv7 tiny	0.91	0.93	0.91	0.83
YOLOv8 nano	0.94	0.97	0.85	0.64
YOLOv9 tiny	0.92	0.94	0.92	0.85

6.4 Diseño de la estructura del prototipo contenedor

La representación general del contenedor se detalla en el modelo 3D presentando un diseño adaptado a las características para su construcción final, figuras 20, 21 y 22.

Figura 20

Diseño frontal del prototipo

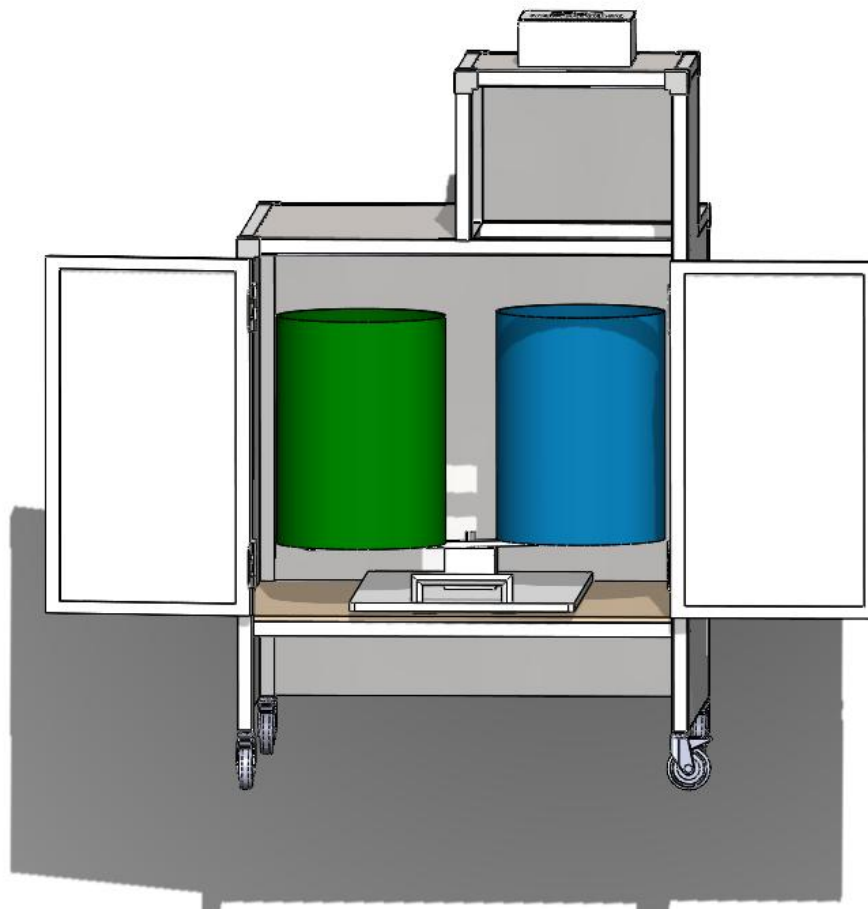


Figura 21

Diseño del sistema interior mecánico

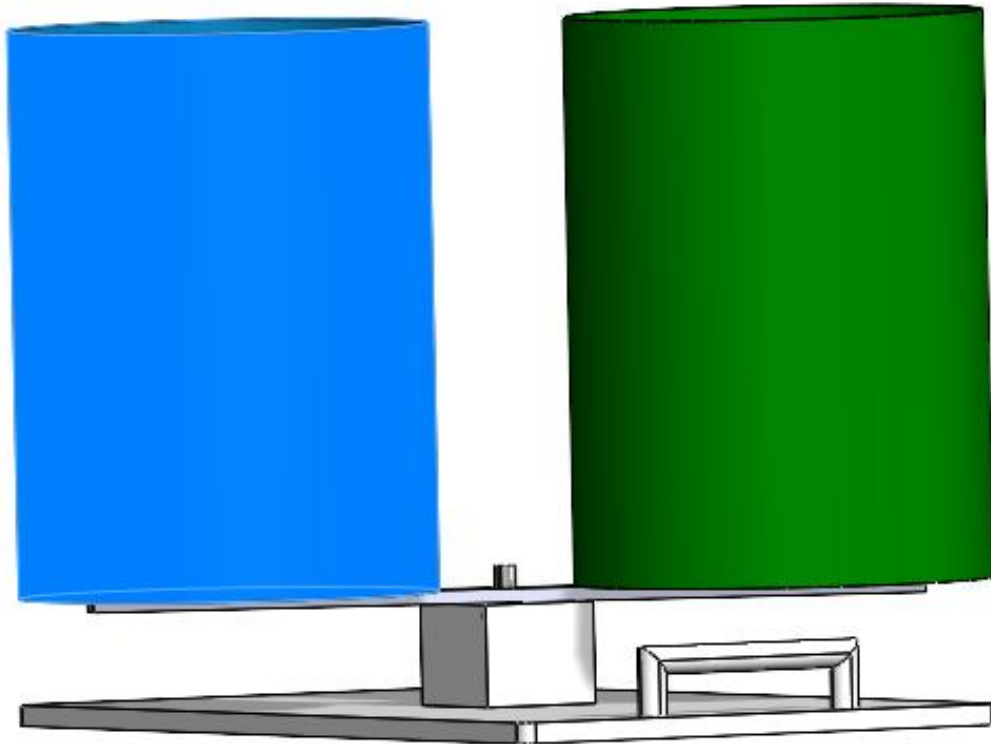
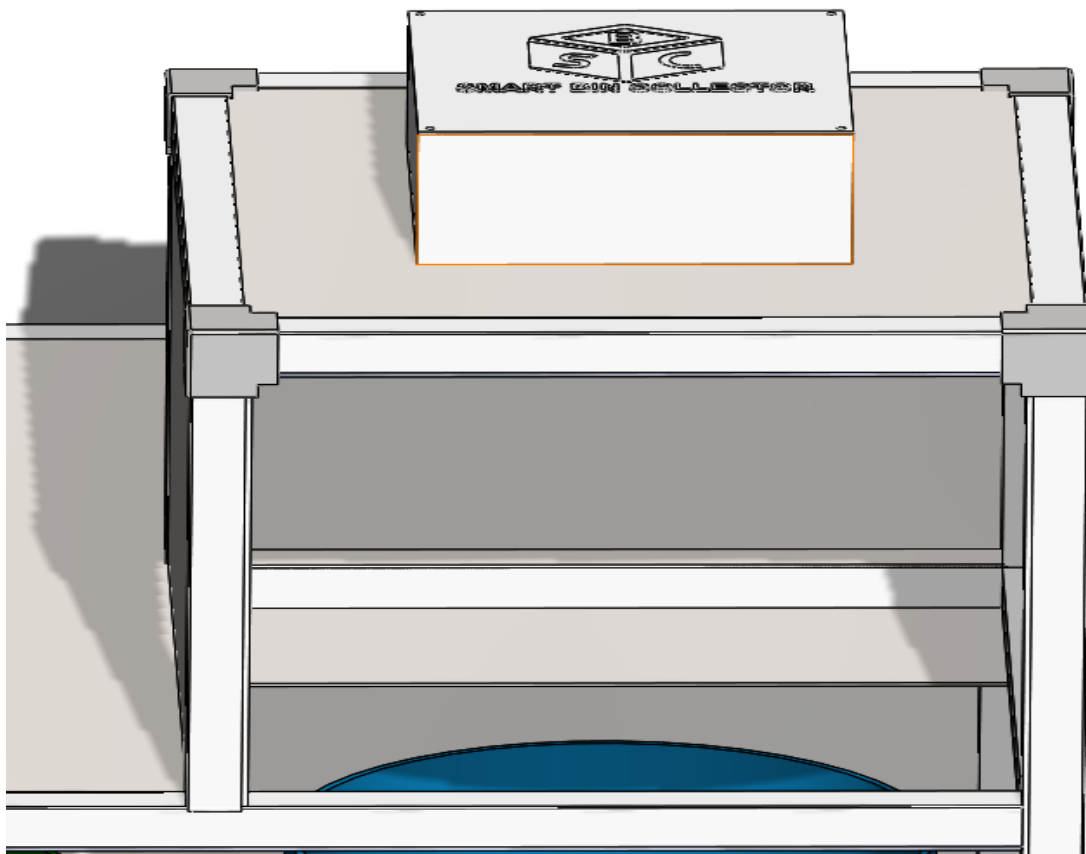


Figura 22

Diseño del compartimento del sistema electrónico.



6.5 Construcción del prototipo

La construcción del prototipo resulto de una estructura 120 cm³ integrada con un sistema embebido Raspberry Pi 4 modelo B, sistema mecánico y electrónico, figura 23, 24, 25.

Figura 23

Diseño general del contenedor.

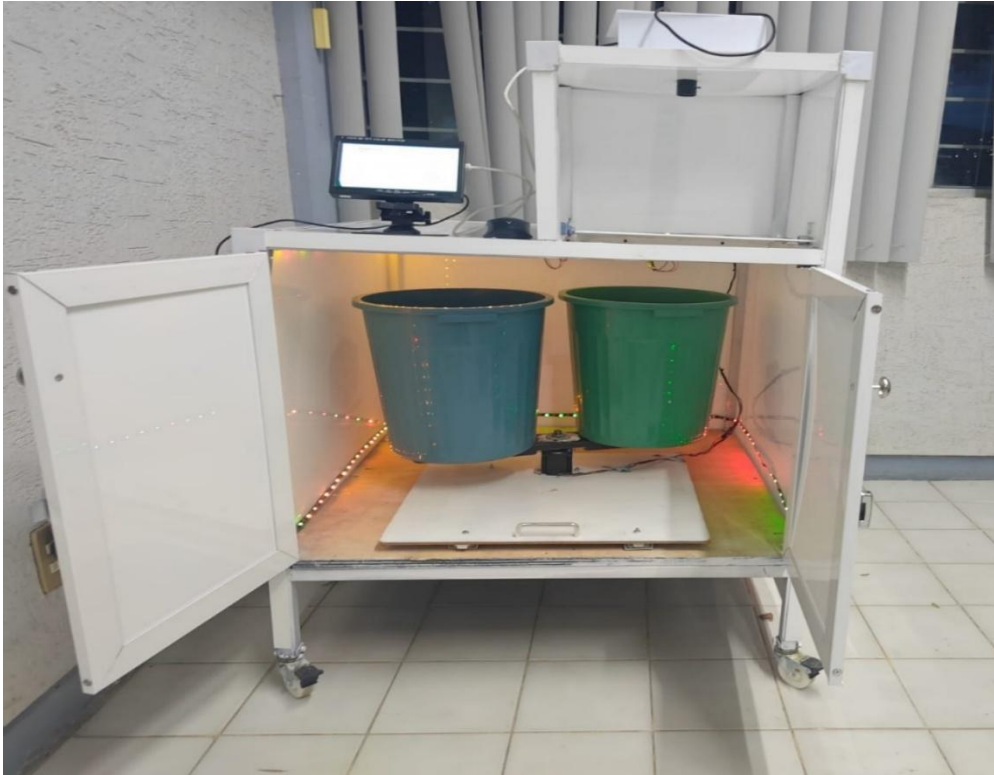


Figura 24

Diseño general del contenedor.

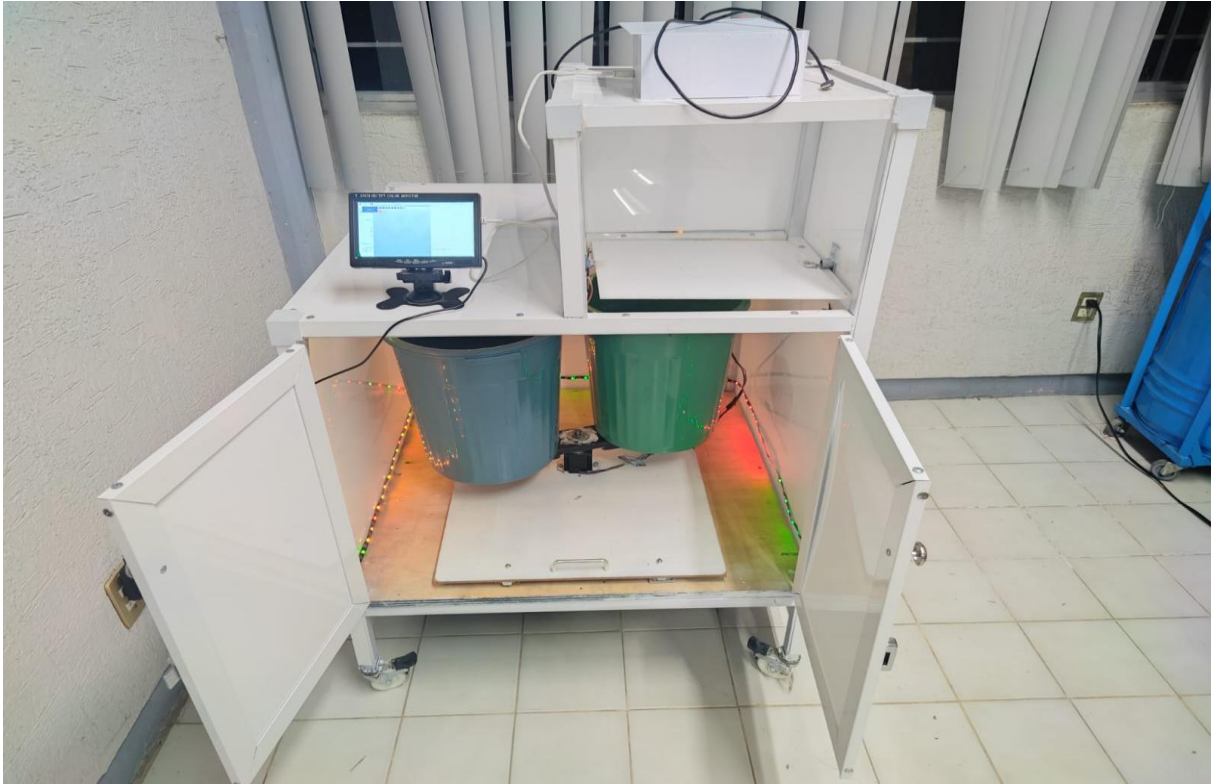


Figura 25

Vista lateral del contenedor.



6.6 Validación el prototipo.

Finalmente se realizan las pruebas del contenedor ingresando residuos orgánicos e inorgánicos para visualizar su funcionamiento, figuras 26 y 27.

Figura 26

YOLOv9 tiny en Raspberry Pi.

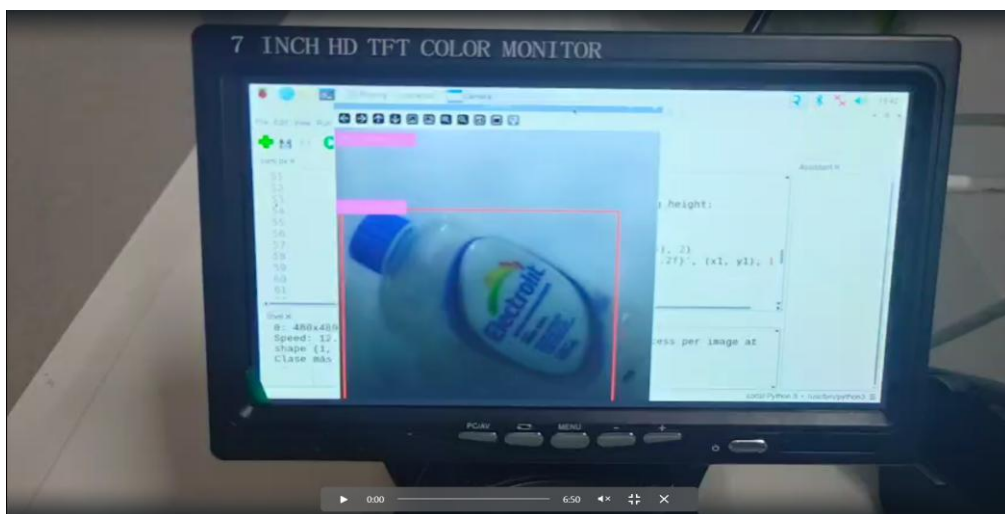


Figura 27

Sistema de clasificación del contenedor.



Capítulo VII. Discusión

Elegir un modelo óptimo para la detección de RSM con características compactas, basada en un sistema embebido con el uso de un minicomputador Raspberry Pi 4 modelo B, requiere evaluar los resultados que se obtienen al momento de realizar pruebas en tiempo real. Es el caso de (Patel et al., 2021) quienes compararon los modelos EfficientDet-D1, SSD ResNet-50 V1, Faster R-CNN ResNet-101 V1, CenterNet ResNet-101 V1 y YOLOv5M mediante 25 pruebas de imágenes de residuos no utilizados en el entrenamiento, lo que se diferencia de este estudio, donde utilizamos imágenes en tiempo real con cuatro modelos YOLO en sus versiones más compactas. Otro aspecto importante, es la evaluación a través del $mAP@0.5$ donde (Patel et al., 2021) en su modelo YOLO5m obtuvieron 0.613 en su promedio más alto, siendo menor a los obtenidos en YOLOv7 tiny = 0.949, YOLOv8 nano = 0.976 y YOLOv9 tiny = 0.968 de esta investigación. Además de que (Patel et al., 2021) no realizaron pruebas en un sistema embebido. Sin embargo (D. Zhang et al., 2022, 2023, 2024) desarrollaron un modelo CNN liviano con un algoritmo eficiente en un dispositivo embebido para la detección de defectos superficiales en productos industriales, alcanzando una precisión comparable a modelos de última generación del estado del arte; lo que concuerda con este trabajo en la implementación de modelos livianos de YOLO en sistemas embebidos.

Referente al conjunto de datos se coincide con (Zhou et al., 2024) quienes también utilizaron Trash-net y Public GarbageNet para entrenar el modelo de detección de RSM. Cabe mencionar que (Zhou et al., 2024) evaluaron un solo modelo con cuatro datasets, mientras que en esta investigación fueron evaluados cuatro modelos con un dataset combinado de Trash-net y Public GarbageNet. El uso de un dispositivo Raspberry Pi 4B como placa principal para la detección de objetos con sistemas embebidos (Chen et al., 2022), lo que

concuerta con este trabajo. Referente a los tiempos de inferencia, en (Chen et al., 2022) evaluaron GCNet, MobileNet v2 y ResNet50, obteniendo 200 ms, 200 ms y 600 ms respectivamente diferente a 1800 ms obtenidos en esta experimentación que detecta más de un tipo de residuo simultáneamente, mientras que (Chen et al., 2022) solo detectan un tipo de residuo.

Al comparar este prototipo con otros trabajos relacionados, se observan diferentes enfoques. Caso de García et al. (2015) en Guadalajara, México, quienes utilizaron un procesador multimedia y un algoritmo k-NN para clasificar latas, botellas de plástico y cubiertos. A la vez que el contenedor propuesto clasifica de manera eficiente residuos orgánicos e inorgánicos, sin depender de la forma o deformación del residuo. Por otro lado, Jiménez et al. (2021) en Michoacán, México, emplean sensores inductivos y capacitivos para la detección de metal, plástico y vidrio. En tanto que esta investigación se diferencia de un sistema más preciso con CNN, logrando una precisión del 92% comparada al 83% de residuos inorgánicos de ese desarrollo.

En Colombia, Pardo-Burbano et al. (2021) desarrollaron un prototipo de un punto ecológico inteligente utilizando MobileNet y diseño en SolidWorks; Pascasio et al. (2022) probaron arquitecturas como VGG16, ResNet50, Inceptionv2 y MobileNet para una tarea de clasificación de seis clases. En cambio, este proyecto radica en la aplicación del modelo YOLOv9 tiny, una arquitectura de CNN reciente para la clasificación binaria de RSU de forma automática.

Capítulo VIII. Conclusiones

Los modelos de detección de objetos en sus versiones YOLO requieren de dispositivos de altas características, sin considerar el consumo de recursos en su implementación, comúnmente no son aplicados en sistemas embebidos, caso del Raspberry Pi 4 Modelo B y la cámara HQ de 12.3 MP, debido a su capacidad limitada en CPU, GPU. Referente a la comparación de los modelos en sus distintas versiones se ha investigado el rendimiento de modelos para la detección de objetos. Esta investigación presenta un análisis comparativo de cuatro modelos YOLO experimentados en las mismas condiciones ambientales y los mismos conjuntos de datos, en sus versiones YOLOv4 tiny, YOLOv7 tiny, YOLOv8 nano y YOLOv9 tiny para conocer el rendimiento de cada uno en cuanto a uso CPU, tiempo de inferencia, FPS, precisión, recall, mAP@0.5, F1-Score y tamaño del modelo. Los resultados obtenidos muestran que YOLOv9 tiny logró un desempeño bastante sólido en las pruebas experimentales con una precisión promedio de 92.11%, Recall 94.97%, F1-Score 0.729, tamaño compacto 4.43 MB, y una ausencia de errores durante las pruebas que lo posicionaron como el mejor modelo en cuanto a estabilidad y rendimiento general. Mientras que YOLOv7 tuvo una alta precisión para reconocer residuos, detectando objetos inexistentes, alto uso de la CPU (97%), dependencia a la ubicación e iluminación del objeto, debilidad en la distancia con residuos orgánicos y alto tiempos de espera para inferir el tipo de objeto. Lo que permitirá observar el comportamiento de los modelos YOLO livianos aplicados en un dispositivo Raspberry Pi 4B sin dependencia de sistemas remotos, implementados, por ejemplo, en un contenedor inteligente clasificador de residuos orgánicos e inorgánicos garantizando su gestión contribuyendo a los Objetivos del Desarrollo Sostenible, 13: acción por el clima.

El prototipo del contenedor clasificador de RSU con CNN propuesto es una alternativa para mitigar la mala gestión de residuos en la actualidad, además de que podría fomentar una cultura en las prácticas de separación. La implementación de modelos de detección de objetos en sistemas embebidos mejora la precisión al momento de determinar la clase de los tipos de residuos. Por consiguiente, el desarrollo de este proyecto establece las bases en los sistemas inteligentes para la clasificación de RSU.

Referencias bibliográficas

- Abdallah, M., Abu Talib, M., Feroz, S., Nasir, Q., Abdalla, H., & Mahfood, B. (2020). Artificial intelligence applications in solid waste management: A systematic research review. *Waste Management (New York, N.Y.)*, 109, 231–246. <https://doi.org/10.1016/j.wasman.2020.04.057>
- Adedeji, O., & Wang, Z. (2019). Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network. *Procedia Manufacturing*, 35, 607–612. <https://doi.org/https://doi.org/10.1016/j.promfg.2019.05.086>
- Amazon Web Services. (2025). *¿Qué es Python?* <https://aws.amazon.com/es/what-is/python/>
- Bin-e. (2025). *Bin-e | Smart Waste Bin*. <https://www.bine.world/>
- Blank, J., & Deb, K. (2020). Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8, 89497–89509. <https://doi.org/10.1109/ACCESS.2020.2990567>
- C, S., Manasa, N., Sharma, V., & A., N. K. A. (2019). Vegetable Classification Using You Only Look Once Algorithm. *2019 International Conference on Cutting-Edge Technologies in Engineering (ICon-CuTE)*, 101–107. <https://doi.org/10.1109/ICon-CuTE47290.2019.8991457>
- Carolis, B. De, Ladogana, F., & Macchiarulo, N. (2020). YOLO TrashNet: Garbage Detection in Video Streams. *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 1–7. <https://doi.org/10.1109/EAIS48028.2020.9122693>
- Castro-Bello, M., Roman-Padilla, D. B., Morales-Morales, C., Campos-Francisco, W., Marmolejo-Vega, C. V., Marmolejo-Duarte, C., Evangelista-Alcocer, Y., & Gutiérrez-Valencia, D. E. (2025). Convolutional Neural Network Models in Municipal Solid Waste Classification: Towards Sustainable Management. *Sustainability*, 17(8). <https://doi.org/10.3390/su17083523>
- Chen, Z., Yang, J., Chen, L., & Jiao, H. (2022). Garbage classification system based on improved ShuffleNet v2. *Resources, Conservation and Recycling*, 178, 106090. <https://doi.org/https://doi.org/10.1016/j.resconrec.2021.106090>
- Chollet, F. (2021). *Deep Learning with Python* (Second). Manning Publications. <https://www.manning.com/books/deep-learning-with-python-second-edition>
- Contreras, E., Gálvez, A., Pacheco, J. F., Toro, E. R., & Narea, M. S. (2016). *Guía general para la gestión de residuos sólidos domiciliarios*. Comisión Económica para América Latina y el Caribe (CEPAL). <https://hdl.handle.net/11362/40407>
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

- Falaschetti, L., Manoni, L., Palma, L., Pierleoni, P., & Turchetti, C. (2024). Embedded Real-Time Vehicle and Pedestrian Detection Using a Compressed Tiny YOLO v3 Architecture. *IEEE Transactions on Intelligent Transportation Systems*, 25(12), 19399–19414. <https://doi.org/10.1109/TITS.2024.3447453>
- Fu, B., Li, S., Wei, J., Li, Q., Wang, Q., & Tu, J. (2021). A Novel Intelligent Garbage Classification System Based on Deep Learning and an Embedded Linux System. *IEEE Access*, 9, 131134–131146. <https://doi.org/10.1109/ACCESS.2021.3114496>
- Gao, X., Wang, G., Qi, J., Wang, Q. (Jenny), Xiang, M., Song, K., & Zhou, Z. (2024). Improved YOLO v7 for Sustainable Agriculture Significantly Improves Precision Rate for Chinese Cabbage (*Brassica pekinensis* Rupr.) Seedling Belt (CCSB) Detection. *Sustainability*, 16(11). <https://doi.org/10.3390/su16114759>
- García, A. T., Aragón, O. R., Gandara, O. L., García, F. S., & Luis Enrique, G. J. (2015). *Intelligent Waste Separator*. Jesuit University of Guadalajara.
- Genia, B. (2025). *Recogida selectiva, ¿qué es y cómo funciona?* <https://geniabioenergy.com/que-es-la-recogida-selectiva/>
- Gude, D. K., Bandari, H., Challa, A. K. R., Tasneem, S., Tasneem, Z., Bhattacharjee, S. B., Lalit, M., Flores, M. A. L., & Goyal, N. (2024). Transforming Urban Sanitation: Enhancing Sustainability through Machine Learning-Driven Waste Processing. *Sustainability*, 16(17). <https://doi.org/10.3390/su16177626>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- IBM. (2025). *¿Qué son las redes neuronales?* <https://www.ibm.com/mx-es/think/topics/neural-networks>
- Intel. (2025). *CPU o GPU: ¿cuál es la diferencia?* <https://www.intel.la/content/www/xl/es/products/docs/processors/cpu-vs-gpu.html>
- Jiménez, E. V. H., Méndez González, L. C., Rodríguez Picón, L. A., & Pérez Olguin, I. J. C. (2021). Diseño de un Clasificador Automático de Residuos Residenciales. *Academia Journals*, 13.
- Jin, B., & Li, W. (2025). Spatial Effects and Driving Factors of Consumption Upgrades on Municipal Solid Waste Eco-Efficiency, Considering Emission Outputs. *Sustainability*, 17(6). <https://doi.org/10.3390/su17062356>
- Kannangara, M., Dua, R., Ahmadi, L., & Bensebaa, F. (2018). Modeling and prediction of regional municipal solid waste generation and diversion in Canada using machine learning approaches. *Waste Management*, 74, 3–15. <https://doi.org/10.1016/j.wasman.2017.11.057>

- Kaza, S., Yao, L. C., Bhada-Tata, P., & Woerden, F. Van. (2018). What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050. *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. <https://doi.org/10.1596/978-1-4648-1329-0>
- Kim, K., Kim, K., & Jeong, S. (2023). Application of YOLO v5 and v8 for Recognition of Safety Risk Factors at Construction Sites. *Sustainability*, 15(20). <https://doi.org/10.3390/su152015179>
- Kraft, M., Piechocki, M., Ptak, B., & Walas, K. (2021). Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle. *Remote Sensing*, 13(5). <https://doi.org/10.3390/rs13050965>
- Kumsetty, N. V., Bhat Nekkare, A., S., S. K., & Kumar M., A. (2022). TrashBox: Trash Detection and Classification using Quantum Transfer Learning. *2022 31st Conference of Open Innovations Association (FRUCT)*, 125–130. <https://doi.org/10.23919/FRUCT54823.2022.9770922>
- Laksono, P. W., Anisa, A., & Priyandari, Y. (2024). Deep learning implementation using convolutional neural network in inorganic packaging waste sorting. *Franklin Open*, 8, 100146. <https://doi.org/https://doi.org/10.1016/j.fraope.2024.100146>
- LGPGIR. (2003, October 8). *LEY GENERAL PARA LA PREVENCIÓN Y GESTIÓN INTEGRAL DE LOS RESIDUOS*. https://dof.gob.mx/nota_detalle.php?codigo=688657&fecha=08/10/2003#gsc.tab=0
- Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999–7019. <https://doi.org/10.1109/TNNLS.2021.3084827>
- Liu, S., Jin, Y., Ruan, Z., Ma, Z., Gao, R., & Su, Z. (2022). Real-Time Detection of Seedling Maize Weeds in Sustainable Agriculture. *Sustainability*, 14(22). <https://doi.org/10.3390/su142215088>
- Majchrowska, S., Mikołajczyk, A., Ferlin, M., Klawikowska, Z., Plantykowski, M. A., Kwasigroch, A., & Majek, K. (2022). Deep learning-based waste detection in natural and urban environments. *Waste Management*, 138, 274–284. <https://doi.org/https://doi.org/10.1016/j.wasman.2021.12.001>
- Mostafa Mohamed. (2020). *Garbage Classification (12 classes)*. <https://www.kaggle.com/datasets/mostafaabla/garbage-classification>
- Pardo Burbano, M., Pinto Rodríguez, V., & Muñoz Ordóñez, J. (2021). Ecología: inteligencia artificial para el cuidado del medio ambiente, un prototipo de clasificación de residuos sólidos en el punto de origen. *Investigación e Innovación En Ingenierías*, 9(3), 46–56. <https://doi.org/10.17081/invinno.9.3.5312>

- Pascasio, J. J., Mela, R. H., Vélez, M. L., & Rangel, J. C. (2022). Implementación de redes neuronales para la clasificación de desechos dentro de un cesto inteligente. *Centros: Revista Científica Universitaria*, 11(1), 229–245.
- Patel, D., Patel, F., Patel, S., Patel, N., Shah, D., & Patel, V. (2021). Garbage Detection using Advanced Object Detection Techniques. *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, 526–531. <https://doi.org/10.1109/ICAIS50930.2021.9395916>
- Perera, P., Tennakoon, G., Ahangama, S., Panditharathna, R., & Chathuranga, B. (2021). A Systematic Mapping of Introductory Programming Languages for Novice Learners. *IEEE Access*, 9, 88121–88136. <https://doi.org/10.1109/ACCESS.2021.3089560>
- Qiu, J., Lu, X., Wang, X., Chen, C., Chen, Y., & Yang, Y. (2024). Research on image recognition of tomato leaf diseases based on improved AlexNet model. *Heliyon*, 10(13). <https://doi.org/10.1016/j.heliyon.2024.e33555>
- Rahman, Md. W., Islam, R., Hasan, A., Bithi, N. I., Hasan, Md. M., & Rahman, M. M. (2022). Intelligent waste management system using deep learning with IoT. *Journal of King Saud University - Computer and Information Sciences*, 34(5), 2072–2087. <https://doi.org/https://doi.org/10.1016/j.jksuci.2020.08.016>
- Raspberry Pi Foundation. (2025). *¿Qué es Raspberry Pi?* <https://raspberrypi.cl/que-es-raspberry/>
- Recytrans S.L. (2025). *Gestión de residuos*. <https://www.recytrans.com/gestion-de-residuos/>
- Redmon, J. (2025). *YOLO: Real-Time Object Detection*. <https://pjreddie.com/darknet/yolo/>
- Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, abs/1506.02640. <http://arxiv.org/abs/1506.02640>
- Rondón Toro, E., Szantó Narea, M., Francisco Pacheco, J., Contreras, E., & Gálvez, A. (2016). *Guía general para la gestión de residuos sólidos domiciliarios*. <https://hdl.handle.net/11362/40407>
- Sashaank Sekar. (2019). *Waste Classification data*. <https://www.kaggle.com/datasets/techsash/waste-classification-data/data>
- SEMARNAT. (2017, January 10). *Residuos sólidos urbanos (RSU)*. <https://www.gob.mx/semarnat/acciones-y-programas/residuos-solidos-urbanos-rsu>
- Singh, A. (2019). Managing the uncertainty problems of municipal solid waste disposal. *Journal of Environmental Management*, 240, 259–265. <https://doi.org/https://doi.org/10.1016/j.jenvman.2019.03.025>
- Tecmed. (2023). *Técnicas Medioambientales de México S.A. de C.V.* <https://www.tecmedmx.com/>

- TensorFlow. (2025). *TensorFlow: An end-to-end open source machine learning platform*. <https://www.tensorflow.org/>
- Thung, G., & Yang, M. (2016). *Classification of Trash for Recyclability Status*. <https://api.semanticscholar.org/CorpusID:27517432>
- Ultralytics. (2025a). *Guía de ajuste de hiperparámetros en Ultralytics YOLO*. <https://docs.ultralytics.com/guides/hyperparameter-tuning/>
- Ultralytics. (2025b, March 22). *Ultralytics Home*. <https://docs.ultralytics.com/es#how-can-i-train-a-custom-yolo-model-on-my-dataset>
- UNEP. (2018). *Outlook of Waste Management in Latin America and the Caribbean*. UNEP. <https://www.unep.org/es/resources/informe/perspectiva-de-la-gestion-de-residuos-en-america-latina-y-el-caribe>
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. <https://arxiv.org/abs/2207.02696>
- Wang, L., Ji, W., Wang, G., Feng, Y., & Du, M. (2024). Intelligent design and optimization of exercise equipment based on fusion algorithm of YOLOv5-ResNet 50. *Alexandria Engineering Journal*, 104, 710–722. <https://doi.org/https://doi.org/10.1016/j.aej.2024.07.105>
- Wu, Z., Zhang, D., Shao, Y., Zhang, X., Zhang, X., Feng, Y., & Cui, P. (2021). Using YOLOv5 for Garbage Classification. *2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, 35–38. <https://doi.org/10.1109/PRAI53619.2021.9550790>
- Yang, Z., Xia, Z., Yang, G., & Lv, Y. (2022). A Garbage Classification Method Based on a Small Convolution Neural Network. *Sustainability*, 14(22). <https://doi.org/10.3390/su142214735>
- YOLOv4. (2021). <https://github.com/AlexeyAB/darknet>
- YOLOv7. (2023). <https://github.com/WongKinYiu/yolov7>
- YOLOv8. (2023). <https://github.com/ultralytics/ultralytics>
- YOLOv9. (2024). <https://github.com/WongKinYiu/yolov9>
- Yuesheng, F., Jian, S., Fuxiang, X., Yang, B., Xiang, Z., Peng, G., Zhengtao, W., & Shengqiao, X. (2021). Circular Fruit and Vegetable Classification Based on Optimized GoogLeNet. *IEEE Access*, 9, 113599–113611. <https://doi.org/10.1109/ACCESS.2021.3105112>
- Zeng, M., Lu, X., Xu, W., Zhou, T., & Liu, Y. (2020). PublicGarbageNet : A Deep Learning Framework for Public Garbage Classification. *2020 39th Chinese Control Conference (CCC)*, 7200–7205. <https://doi.org/10.23919/CCC50068.2020.9189561>

- Zhang, D., Hao, X., Liang, L., Liu, W., & Qin, C. (2022). A novel deep convolutional neural network algorithm for surface defect detection. *Journal of Computational Design and Engineering*, 9(5), 1616–1632. <https://doi.org/10.1093/jcde/qwac071>
- Zhang, D., Hao, X., Wang, D., Qin, C., Zhao, B., Liang, L., & Liu, W. (2023). An efficient lightweight convolutional neural network for industrial surface defect detection. *Artificial Intelligence Review*, 56(9), 10651–10677. <https://doi.org/10.1007/s10462-023-10438-y>
- Zhang, D., Wang, Y., Meng, L., Yan, J., & Qin, C. (2024). Adaptive critic design for safety-optimal FTC of unknown nonlinear systems with asymmetric constrained-input. *ISA Transactions*, 155, 309–318. <https://doi.org/https://doi.org/10.1016/j.isatra.2024.09.018>
- Zhang, Q., Yang, Q., Zhang, X., Bao, Q., Su, J., & Liu, X. (2021). Waste image classification based on transfer learning and convolutional neural network. *Waste Management*, 135, 150–157. <https://doi.org/https://doi.org/10.1016/j.wasman.2021.08.038>
- Zhou, Y., Wang, Z., Zheng, S., Zhou, L., Dai, L., Luo, H., Zhang, Z., & Sui, M. (2024). Optimization of automated garbage recognition model based on ResNet-50 and weakly supervised CNN for sustainable urban development. *Alexandria Engineering Journal*, 108, 415–427. <https://doi.org/https://doi.org/10.1016/j.aej.2024.07.066>

IX. Anexos

Anexo 1. Código sistema embebido Raspberry Pi 4 modelo B

```
1  import cv2
2  from picamera2 import Picamera2
3  import pandas as pd
4  from ultralytics import YOLO
5  import cvzone
6  import numpy as np
7  import serial
8  import time
9  from collections import Counter # Para contar la frecuencia de las detecciones
10
11 # Configurar la comunicación serial con Arduino Mega
12 arduino = serial.Serial('/dev/ttyUSB0', 9600, timeout=1) # Ajusta el puerto si es necesario
13 time.sleep(2) # Esperar a que se establezca la comunicación
14
15 # Configurar la cámara
16 picam2 = Picamera2()
17 picam2.preview_configuration.main.size = (480, 480)
18 picam2.preview_configuration.main.format = "RGB888"
19 picam2.preview_configuration.align()
20 picam2.configure("preview")
21 picam2.start()
22
23 # Cargar el modelo YOLO
24 model = YOLO('best.pt')
25
26 # Leer la lista de clases
27 with open("coco1.txt", "r") as f:
28     class_list = f.read().split("\n")
29
30 detections = [] # Lista para almacenar detecciones en un intervalo de tiempo
31 sampling_time = 5 # Segundos para promediar
32 last_time = time.time()
33
34 while True:
35     im = picam2.capture_array()
36
37     # Realizar la predicción
38     results = model.predict(im, imgsz=480)
39     a = results[0].boxes.data
40     px = pd.DataFrame(a).astype("float")
41
42     detected_classes = [] # Lista temporal para cada frame
43
44     for index, row in px.iterrows():
45         x1, y1, x2, y2 = map(int, row[:4])
46         conf, d = row[4], int(row[5])
47         detected_class = class_list[d].lower() # Convertir a minúsculas para evitar errores
48
49         # Dibujar el cuadro y la clase detectada
50         cv2.rectangle(im, (x1, y1), (x2, y2), (0, 0, 255), 2)
51         cvzone.putTextRect(im, f'{detected_class} {conf:.2f}', (x1, y1), 1, 1)
52
53         detected_classes.append(detected_class) # Guardamos la detección de este frame
54
55     # Agregar al historial de detecciones
56     detections.extend(detected_classes)
```

```

57
58     # Si ha pasado el tiempo de muestreo (5 segundos)
59     if time.time() - last_time >= sampling_time:
60         if detections:
61             # Contar la frecuencia de cada clase
62             counted = Counter(detections)
63             most_common_class = counted.most_common(1)[0][0] # Clase más detectada
64
65             # Enviar comando al Arduino según el promedio
66             if most_common_class == "organic":
67                 arduino.write(b'O')
68             elif most_common_class == "inorganic":
69                 arduino.write(b'I')
70
71             # Esperar respuesta del Arduino
72             while arduino.in_waiting > 0:
73                 respuesta = arduino.readline().decode('ISO-8859-1').strip()
74
75                 print(f"Arduino: {respuesta}")
76
77             # Reiniciar el contador y limpiar detecciones
78             detections.clear()
79             last_time = time.time()
80
81         # Mostrar FPS en la imagen
82         fps = 1 / (time.time() - last_time)
83         cvzone.putTextRect(im, f'FPS: {fps:.2f}', (10, 30), 1, 1)
84
85         cv2.imshow("Camera", im)
86
87         if cv2.waitKey(1) == ord('q'):
88             break
89
90     cv2.destroyAllWindows()
91     arduino.close()
92

```

Anexo 2. Código sistema electrónico Arduino Mega 2560

```
1  #include <Servo.h>
2  #include <Adafruit_NeoPixel.h>
3
4  // Pines del motorA
5  const int ena      = 2;
6  const int dir      = 3;
7  const int pul      = 4;
8  const int pasos    = 3450;
9  const int ajuste0  = 100;
10 const int ajusteI   = 150;
11 const int velocidad = 700;
12
13 // Ángulos de compuertas
14 const int ANG_CERRADO = 140;
15 const int ANG_ABIERTO = 80;
16
17 int estado = 0; // 0 = indefinido, 1 = Orgánico, 2 = Inorgánico
18
19 // Servos
20 Servo compuerta1;
21 Servo compuerta2;
22
23 // WS2812B
24 #define LED_PIN      6
25 #define NUMPIXELS    150
26 Adafruit_NeoPixel strip(NUMPIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);
27
28 unsigned long ultimoCambio = 0;
29 int colorIndex = 0;
```

```

30
31 void setup() {
32     compuerta1.attach(9);
33     compuerta2.attach(10);
34     compuerta1.write(ANG_CERRADO);
35     compuerta2.write(ANG_CERRADO);
36
37     pinMode(ena, OUTPUT);
38     pinMode(dir, OUTPUT);
39     pinMode(pul, OUTPUT);
40     digitalWrite(ena, LOW);
41     digitalWrite(pul, HIGH);
42
43     strip.begin();
44     strip.show();
45
46     Serial.begin(9600);
47 }
48
49 void loop() {
50
51     if (Serial.available()) {
52         char comando = Serial.read();
53
54         if (comando == 'O' && estado != 1) {
55             digitalWrite(dir, HIGH);
56             moverMotor(pasos - ajuste0);
57             estado = 1;

```

```

56 moverMotor(pasos - ajuste0);
57 estado = 1;
58 Serial.println("Movimiento: Orgánico");
59 encenderColor(0, 255, 0); // Verde
60 delay(1000);
61 manejarCompuertas();
62 delay(2000);
63 }
64 ✓ else if (comando == 'I' && estado != 2) {
65     digitalWrite(dir, LOW);
66     moverMotor(pasos - ajusteI);
67     estado = 2;
68     Serial.println("Movimiento: Inorgánico");
69     encenderColor(0, 0, 255); // Azul
70     delay(1000);
71     manejarCompuertas();
72     delay(2000);
73 }
74 ✓ } else {
75     // Mostrar colores RGB en ciclo si no hay comandos
76     cicloRGB();
77 }
78 }
79
80 ✓ void moverMotor(int numPasos) {
81 ✓     for (int i = 0; i < numPasos; i++) {
82         digitalWrite(pul, HIGH);
83         delayMicroseconds(velocidad);
84         digitalWrite(pul, LOW);

```

```

83     delayMicroseconds(velocidad);
84     digitalWrite(pul, LOW);
85     delayMicroseconds(velocidad);
86 }
87 }
88
89 void manejarCompuertas() {
90     compuerta1.write(ANG_ABIERTO);
91     compuerta2.write(ANG_ABIERTO);
92     delay(2000);
93
94     compuerta1.write(ANG_CERRADO);
95     delay(500);
96     compuerta2.write(ANG_CERRADO);
97     delay(500);
98 }
99
100 void encenderColor(uint8_t r, uint8_t g, uint8_t b) {
101     for (int i = 0; i < NUMPIXELS; i++) {
102         strip.setPixelColor(i, strip.Color(r, g, b));
103     }
104     strip.show();
105 }
106
107 void cicloRGB() {
108     unsigned long ahora = millis();
109     if (ahora - ultimoCambio >= 500) {
110         ultimoCambio = ahora;

```

```

100 void encenderColor(uint8_t r, uint8_t g, uint8_t b) {
101     for (int i = 0; i < NUMPIXELS; i++) {
102         strip.setPixelColor(i, strip.Color(r, g, b));
103     }
104     strip.show();
105 }
106
107 void cicloRGB() {
108     unsigned long ahora = millis();
109     if (ahora - ultimoCambio >= 500) {
110         ultimoCambio = ahora;
111         colorIndex = (colorIndex + 1) % 3;
112
113         switch (colorIndex) {
114             case 0:
115                 encenderColor(255, 0, 0); // Rojo
116                 break;
117             case 1:
118                 encenderColor(0, 255, 0); // Verde
119                 break;
120             case 2:
121                 encenderColor(0, 0, 255); // Azul
122                 break;
123         }
124     }
125 }

```

Anexo 3. Código CMD de entrenamiento YOLO

```

3 python train.py --workers 8 --device 0 --batch-size 32 --epochs 100 --img 640 640 --hyp data/hyp.scratch.tiny.yaml --name yolov7-custom --weights yolov7-tiny.pt
4 *incrementar batch size
5
6 python detect.py --weights best.pt --conf 0.25 --img-size 640 --source 0
7
8 python detect.py --weights best.pt --conf 0.5 --img-size 640 --source img.jpg --view-img --no-trace
9
10
11
12 darknet.exe detector map data/obj.data cfg/yolov4-tiny-custom.cfg ../training/yolov4-tiny-custom_xxxx.weights -points 0
13
14
15 yolo task=detect mode=train model=yolov8n.pt data= data.yaml epochs=100 imgsz=640 plots=True device=0 batch=32
16
17 yolo train model=yolov9t.yaml data=data.yaml epochs=100 imgsz=640

```

Anexo 4. Productos académicos

1. Artículo de Revista Indizada en Scopus y Web of Science (JCR), Convolutional Neural Network Models in Municipal Solid Waste Classification: Towards Sustainable Management (Castro-Bello et al., 2025).