Técnicas de Deep Learning aplicadas a un sistema de clasificación de objetos para un recolector de residuos inteligente

Mauro Salina[1], Braian Pezet[1], Lucia Osés[1], Marcelo Cappelletti[1,2], Jorge Osio[1,2], Martín Morales[1,3]

[1] Programa TICAPPS, Univ. Nac. Arturo Jauretche, Florencio Varela (1888), Argentina.
[2] Grupo de Control Aplicado (GCA), Instituto LEICI (UNLP-CONICET), La Plata (1900), Argentina
[3] Centro UTN CODAPLI-FRLP, Berisso, Argentina.
{braianpezet, luciaoses1997} @gmail.com {josio, mdsalina, mcappelletti, martin.morales } @unaj.edu.ar

Resumen. El objetivo principal del trabajo consiste en el desarrollo de un sistema de clasificación de objetos para ser utilizado en un recolector de residuos inteligente, aplicando técnicas de Aprendizaje Profundo (Deep Learning). Durante el desarrollo del trabajo se crearon modelos de redes neuronales convolucionales (CNN) capaces de identificar distintos objetos reciclables en diferentes imágenes, en tiempo real. También se realizaron pruebas utilizando modelos pre-entrenados con aprendizaje por transferencia (Transfer Learning) para comparar resultados. Estos modelos fueron implementados utilizando como lenguaje de programación Python, apoyándose en el Framework de backend TensorFlow y la librería de alto nivel Keras. Adicionalmente, se fueron evaluando una amplia variedad de herramientas de inteligencia artificial que permiten aplicar las técnicas de Deep learning de forma eficiente. En pos de conseguir mejores resultados, se llevaron a cabo pruebas con distintos datasets y diferentes modelos dejando en evidencia la importancia que tiene el armado de un dataset bien nutrido, con una buena distribución de las muestras al momento de aplicar las métricas. En todos los casos se realizó clasificación multiclase, en donde con los primeros modelos se contó con aproximadamente 8000 imágenes divididas en 4 clases (plástico, vidrio, metal, papel-cartón), y para modelos posteriores se incrementó el dataset contando con más de 15000 imágenes separadas en 6 clases distintas, agregando a las anteriores las clases “orgánico” y “no-reciclable”. La implementación de estos modelos se llevó a cabo utilizando el lenguaje de programación Python, mediante el algoritmo “You Only Look Once” (YOLO). Como parte de la validación, se probó el modelo final en una aplicación (versión beta) desarrollada en Python, utilizando una mini computadora Raspberry Pi y un módulo de cámara (picam). El sistema desarrollado permite analizar en tiempo real los fotogramas capturados por la cámara y aplicar el modelo de clasificación de manera instantánea, accediendo de esta forma a las coordenadas de dichos objetos en el fotograma para poder recolectarlos y separarlos para su posterior reciclaje.

Palabras clave: Aprendizaje Automático, Aprendizaje Profundo, sistema de reciclaje, procesamiento de imágenes.

Abstract. The main objective of the work is the development of an object classification system to be used in an intelligent waste collector, applying Deep Learning techniques. During the work, models of convolutional neural networks (CNN) were created, capable of identifying different recyclable objects in different images in real time. Tests were also carried out using pre-trained models with transfer learning to compare results. These models were implemented using Python as the programming language, with support from the TensorFlow backend framework and the high-level Keras library. Additionally, a wide variety of artificial intelligence tools that allow for the efficient application of Deep Learning techniques were evaluated. In order to achieve better results, tests with different datasets and different models were carried out, highlighting the importance of building a well-nourished dataset with a good distribution of samples when applying metrics. In all cases, multi-class classification was performed. The initial models used approximately 8000 images divided into 4 classes (plastic, glass, metal, paper-cardboard), and for subsequent models, the dataset was increased to over 15,000 images divided into 6 different classes, adding "organic" and "non-recyclable" classes to the previous ones. The implementation of these models was carried out using the Python programming language, using the "You Only Look Once" (YOLO) algorithm. As part of the validation, the final model was tested in a Python-based application (beta version) using a Raspberry Pi mini computer and a camera module (picam). The developed system allows for the real-time analysis of frames captured by the camera and the instantaneous application of the classification model. This provides access to the coordinates of the objects in the frame to collect and separate them for subsequent recycling.

Keywords: Machine Learning, Deep Learning, recycling system, image processing

Received January 2024; Accepted March 2024; Published May 2024

https://doi.org/10.24215/15146774e041

ISSN 1514-6774

1 - Introducción

La urgencia de abordar la problemática ambiental contemporánea se refleja de manera acuciante en la necesidad imperiosa de implementar soluciones efectivas, y una de las áreas primordiales en este sentido es la gestión de residuos a través de su separación para fines de reutilización. El proceso de reciclaje se erige como un pilar fundamental en la búsqueda de la sostenibilidad ambiental, ya que implica la reincorporación de materiales descartados al ciclo productivo, mitigando así el impacto negativo de su acumulación descontrolada en vertederos y la consiguiente degradación del entorno natural. Este enfoque no solo promueve la reducción de la generación de desechos, sino que también fomenta una conciencia colectiva sobre la importancia de valorar y aprovechar los recursos de manera responsable, en aras de preservar la salud del planeta para las generaciones futuras. [1]

En la actualidad existen normativas e iniciativas estatales y privadas que apoyan la recuperación de residuos, cooperativas de recicladores urbanos, contenedores, puntos verdes y también recicladores informales. Pero, aun así, el problema de la basura sigue siendo un desafío colectivo. Es por ello que se considera sumamente importante el reciclado de desechos para el cuidado del medioambiente. En Argentina, cada dos segundos se produce una tonelada de basura. Una fracción grande de ella termina en rellenos sanitarios que están al borde del colapso [2]. La basura doméstica es uno de los principales desechos y está compuesta principalmente por papel, plástico, vidrio, metales y pilas; es por esto que se considera indispensable realizar la separación en origen para minimizar la generación de residuos y la contaminación. Sin embargo, en Argentina, solo el 24% de la población se esfuerza por separar los residuos inorgánicos, lo que se debe en gran parte al esfuerzo que se requiere para clasificar y separar los residuos [2].
Para dimensionar el daño que se está generando, actualmente estamos consumiendo más recursos que nunca, superando la capacidad de generación del planeta. Mientras tanto, aumenta el impacto en términos de uso de recursos, degradación ambiental, desechos y contaminación.
Es por esto, que la clave es profundizar el cambio cultural para dejar de pensar al residuo como un desecho y entenderlo como un recurso. El consumo y la producción sostenibles (CPS) busca desvincular el crecimiento económico de la degradación ambiental, aumentar la eficiencia de recursos y promover estilos de vida sostenibles [3].

Las adquisiciones sostenibles permiten a nivel nacional, regional y mundial reducir las emisiones de gases de efecto invernadero, mejorar la eficiencia de los recursos y apoyar el reciclaje. Argentina se encuentra dentro de la agenda 2030 de desarrollo sostenible [4], Meta 12.5: de aquí a 2030, reducir considerablemente la generación de desechos mediante actividades de prevención, reducción, reciclado y reutilización.

Teniendo en cuenta los avances tecnológicos de los últimos años, en este trabajo se pretende diseñar soluciones eficientes que puedan inspirar y motivar a las personas a llevar estilos de vida más sostenibles, reduciendo los efectos negativos y mejorando el bienestar. Por esta razón, se propone desarrollar un sistema de visión por computadora que permita detectar y clasificar objetos reciclables para automatizar la separación de los residuos que se generan diariamente.
En la literatura, se pueden encontrar trabajos previos relacionados con recolectores de residuos inteligentes en donde se plantea un basurero inteligente para ciudades, que consiste en contenedores con hardware basado en un microcontrolador y un sensor infrarrojo cuyo objetivo es simplemente determinar si el contenedor está lleno [5]. En relación a la detección de objetos reciclables usando aprendizaje profundo se han realizado varios trabajos similares [6-8]. En [6] se aplicaron técnicas de aprendizaje profundo para la detección de objetos reciclables con un dataset de 3800 imágenes y 12 categorías, lo que se considera contraproducente para el entrenamiento debido a que se dispone de muy pocas imágenes por categoría. Esto se ve reflejado en los resultados, debido a que cuando se utilizan imágenes similares a las del entrenamiento para la verificación del modelo la exactitud es alta, pero con imágenes diferentes a las del entrenamiento, la exactitud disminuye

notablemente. En [7] se aplicaron técnicas de aprendizaje profundo para implementar un sistema de clasificación de imágenes que contenga objetos reciclables según 5 categorías (vidrio, plástico, papel, cartón, orgánico). Si bien muestran un resultado aceptable con métricas que rondan el 92% de acierto, nuestro trabajo cuenta con dos categorías extra a la hora de clasificar (metal, no-reciclable) y adicionalmente se basa en la detección de zonas de interés en una imagen, lo que permite detectar más de un objeto a la vez. Inclusive, nuestro trabajo logró métricas similares con un set de datos menor al utilizado en [7]. En [8] se presenta un trabajo en donde se utilizan 4 categorías, (pensado para realizar separación de residuos en centros de reciclaje), en donde se utilizó un dataset de 1974 expandiendo el set original mediante la técnica de data augmentation y para la fase de entrenamiento de la categoría metal se utilizaron solo 332 imágenes lo que explica la baja probabilidad de acierto obtenida en la matriz de confusión (62%). Finalmente, en [9] se implementó un sistema de detección de bolsas de residuos mediante visión artificial por lo que el propósito es detectar bolsas de residuos en la vía pública para que posteriormente sean recolectadas, aunque el propósito de este trabajo no es la clasificación y separación de residuos, resulta interesante el uso de una minicomputadora para la ejecución y procesamiento del algoritmo.

Por su parte, nuestro trabajo se enfoca en el desarrollo de una aplicación de software que utiliza redes neuronales convolucionales para clasificar imágenes en tiempo real y detectar la presencia de objetos reciclables en la imagen [10]. Para dicho desarrollo se utilizaron herramientas de Inteligencia Artificial (IA), la cual ha tenido un avance significativo en la última década en áreas tales como la detección de objetos y la clasificación de imágenes [11]. Esto se debe, en gran parte, a las nuevas técnicas de Aprendizaje Automático (Machine Learning) y Aprendizaje Profundo (Deep Learning) [12-17], así como a las innovaciones en el manejo de Big Data y el aumento en la capacidad de cómputo mediante el uso de diferentes tecnologías, como la computación en la nube o el uso de GPU (unidad de procesamiento gráfico). Este avance puede verse reflejado en distintas áreas como: medicina, seguridad, turismo, finanzas, robótica, entre otras. Algunos ejemplos de dichos avances en el área se aplican en: control de vehículos autónomos, detección de rostros, detección de matrículas, diagnóstico de enfermedades, realidad aumentada, etc.

En particular, Machine Learning es un subcampo de la IA que utiliza diferentes algoritmos para recolectar datos y realizar un aprendizaje para luego hacer una predicción o sugerencia sobre algo [14]. De esta manera es posible resolver problemas de forma intuitiva y automatizada, sin que el mecanismo de elección se encuentre previamente programado. En la práctica esto se traduce en una función matemática en la que se parte de una entrada y se obtiene una salida, por lo que el desafío reside en construir un modelado automático de esta función matemática.

Por su parte, Deep Learning es un subcampo de Machine Learning que utiliza redes neuronales artificiales compuestas por varios niveles jerárquicos para realizar procesos de aprendizaje automático [17]. En cada nivel se aprenden patrones cada vez más complejos, generando información más sofisticada que se traslada al nivel posterior. En otras palabras, Deep Learning estructura los algoritmos en capas, para crear una red neuronal artificial, que puede aprender y tomar decisiones por sí misma. Estas redes neuronales identifican patrones y clasifican diferentes tipos de información. Las diferentes capas de las redes neuronales sirven como filtro, yendo desde los elementos más generales a los más sutiles, aumentando la probabilidad de detectar y generar un resultado correcto. Por lo tanto, cuando un sistema de Deep Learning tiene que reconocer un objeto, busca en él las características que permitan clasificarlo en alguna de las clases predefinidas.

El reconocimiento de objetos en forma artificial brinda una aproximación a la capacidad del sentido de la vista humana, de manera tal que un sistema es capaz de conocer por sí solo su entorno e interactuar con él, según la información recolectada, tal y como lo hace el cerebro humano. Actualmente, existe un campo de la Inteligencia Artificial denominado Visión Artificial que ha evolucionado notablemente [18]. Ésta tiene como objetivo programar un software para que "entienda" una escena o las características de una imagen. Esta información es empleada para tomar decisiones o controlar procesos.

Los sistemas de Visión Artificial en los procesos tecnológicos y dentro de estos los procesos de producción pueden realizar tareas de manera más efectivas y adecuadas que la visión humana, tal es el caso de los siguientes aspectos:

✓ Dentro del espectro electromagnético la visión humana solamente capta un pequeño rango de frecuencias y amplitudes: "rango de luz visible", mientras que los sistemas de visión artificial pueden captar todo el espectro, es decir, además del rango de luz visible puede captar ondas de radio, de televisión, microondas, infrarrojos, ultravioletas, rayos X, rayos gamma y rayos cósmicos.

✓ La velocidad de respuesta de la visión humana es de 0.06 segundos, mientras que en las cámaras de estado sólido es de 0.00001 segundos y este tiempo se va reduciendo según se mejora la electrónica de estos sistemas.

✓ A diferencia de los sistemas artificiales, la visión humana se cansa, se ve afectada por las emociones y es poco consistente por la fatiga y las distracciones, en cambio la visión artificial mantiene su nivel de rendimiento constante a lo largo de su vida útil. Es ideal en trabajos repetitivos y monótonos.

✓ El ser humano puede discernir entre 10 o 20 niveles de gris, sin embargo, los sistemas de visión artificial tienen una definición muy superior.

✓ La visión humana tiene muy poca precisión y para obtener información cuantitativa necesita apoyarse en instrumentos de medida, mientras que los sistemas de visión artificial tienen gran precisión en la medición, dependiendo solamente de la resolución espacial de los componentes del sistema.

✓ Los sistemas de visión artificial pueden trabajar en entornos muy peligrosos, con residuos radiactivos, químicos, biológicos, ruido, polución, temperaturas muy altas y muy bajas.

Adicionalmente, el uso de la visión artificial crece rápidamente gracias a las ventajas que tiene para las industrias, entre ellas:

✓ Procesa de una manera más simple y rápida: permite a los clientes y a las industrias chequear los productos. Además, les da acceso a sus productos.

✓ Fiabilidad: las computadoras y las cámaras no tienen el factor humano del cansancio. La eficiencia suele ser la misma, no depende de factores externos como pueden ser bajas por enfermedad o errores humanos por agotamiento.

✓ Precisión: esta tecnología asegura una mejor precisión en el producto final.

✓ Una amplia gama de usos: se puede ver el mismo sistema informático en varios campos y actividades diferentes (fábricas con seguimiento de almacenes y envío de suministros, y en la industria médica a través de imágenes escaneadas, entre otras múltiples opciones).

✓ La reducción de los costes: el tiempo y la tasa de error se reducen en el proceso.

Este trabajo presenta el desarrollo de un sistema de visión por computadora que permite la detección y clasificación de objetos reciclables en tiempo real. El sistema está basado en el uso de redes neuronales convolucionales, las cuales han demostrado ser las más eficientes en el área del procesamiento de imágenes. Como característica relevante, se definieron seis categorías (metal, papel, plástico, vidrio, orgánico y no reciclable), lo que posibilita la separación tanto en origen como en centros de reciclaje. Entre las mejoras propuestas, respecto a trabajos similares, se desarrolló un amplio dataset que está en el orden de las 15000 imágenes, se incrementó el número de épocas durante los entrenamientos y se realizaron pruebas de ejecución en tiempo real en sistemas de bajo consumo (mini computadora raspberry PI). Este sistema posee un marcado interés tecnológico y social, dado que permite automatizar la separación de residuos que son generados diariamente.

2 - Materiales y Métodos

2.1 - Redes neuronales convolucionales

Las redes neuronales tradicionales son redes completamente conectadas (fully connected), esto quiere decir que todas las neuronas de una capa oculta están conectadas con todas las neuronas de la capa siguiente y de la capa anterior. El problema de este tipo de redes es que cuando los datos de entrada son imágenes, la cantidad de conexiones necesarias es demasiado grande, lo cual hace que el uso de redes "fully connected" sea prácticamente inviable (incluso para imágenes de tamaño relativamente pequeño en redes muy profundas). Esto es debido a que, para una computadora, una imagen es representada como una matriz de píxeles, y por lo tanto, cada píxel de la imagen de entrada se encuentra conectado a cada neurona de la primera capa oculta de la red. En este caso, el poder computacional y el tiempo requerido aumentará considerablemente, debido a que, si la red toma como entrada los píxeles de una imagen, y se tiene una imagen con apenas 28×28 píxeles de alto y ancho, eso equivale a 784 neuronas, considerando un solo color (escala de grises). En una imagen a color, se necesitan 3 canales (red, green, blue) por lo que entonces se usarán 28x28x3 = 2352 neuronas de entrada, las cuales formarán parte de la capa de entrada. Para mitigar este problema se presentan las redes neuronales convolucionales (CNN).

Las CNN son un tipo de redes neuronales artificiales que procesan capas imitando al córtex visual del cerebro humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y "ver".

Su principal ventaja es que cada parte de la red es entrenada para realizar una tarea específica, aprendiendo diferentes niveles de abstracción, por lo que se reduce significativamente el número de conexiones en las capas ocultas y el entrenamiento es más rápido. Las CNNs son muy potentes para todo lo que tiene que ver con el análisis de imágenes, debido a que la CNN contiene varias capas ocultas especializadas y con una jerarquía, esto significa que las primeras capas puedes detectar características básicas como líneas, curvas o bordes y se van especializando hasta llegar a capas más profundas capaces de reconocer formas complejas como un rostro, una silueta o un objeto. [19].

La arquitectura de una red neuronal convolucional (CNN) se puede separar en dos etapas. Por un lado, la etapa de extracción de características, que a su vez está compuesta por una o más capas convolucionales, en donde se realizan pasos fundamentales como la convolución y el agrupamiento. Y, por otro lado, la etapa de clasificación, también llamada capa completamente conectada, la cual comienza con un aplanamiento y continúa con una red neuronal tradicional totalmente conectada, en donde se realizará la clasificación final [20].

2.2 - Herramientas de Software

A continuación, se describen de manera resumida las herramientas y los procedimientos utilizados en este trabajo. Todas estas herramientas son librerías y frameworks de uso libre desarrolladas por distintas empresas.

- ***LabelImg:*** Es una herramienta de código abierto que se utiliza para la anotación de imágenes y la creación de conjuntos de datos etiquetados para el entrenamiento de modelos de aprendizaje automático.
- ***Conda:*** Anaconda es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos y aprendizaje automático (machine learning). Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue y administración de los paquetes de software.
- ***Google Colab:*** Es un servicio cloud, basado en los Notebooks de Jupyter, que permite el uso gratuito de las GPUs y TPUs de Google, con librerías como: Scikit-learn, PyTorch, TensorFlow, Keras y OpenCV. Todo ello bajo Python 2.7 y 3.6 ([21] y [22]).

- **TensorFlow:** Es una librería de código abierto para Machine Learning. Esta librería de computación matemática ejecuta de forma rápida y eficiente gráficos de flujo [23]. Estos gráficos están formados por operaciones matemáticas representadas sobre nudos y cuyas entradas y salidas son vectores multidimensionales de datos, conocidos como tensores (ver Fig. 1) [24] y[25].
- ***Keras:*** Es una Interfaz de Programación de Aplicaciones (API) de alto nivel y de código abierto escrita en Python. Está especialmente diseñada para facilitar una rápida experimentación en Deep Learning [26].
- ***PyTorch:*** Es una biblioteca de aprendizaje automático de código abierto para Python que se utiliza para la construcción de modelos de aprendizaje profundo y se caracteriza por su facilidad de uso y flexibilidad.

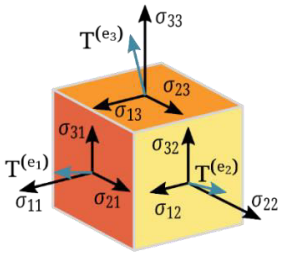

Fig. 1. Representación de un tensor

2.3 - Hardware utilizado

El uso de técnicas de aprendizaje automático [27], como el aprendizaje profundo, específicamente las redes neuronales convolucionales, conlleva un alto costo computacional. Esto se debe a que estas redes están compuestas por un gran número de capas y neuronas, y trabajan con imágenes de alta resolución en muchos casos.

El procesador central (CPU) de una computadora es capaz de realizar operaciones matemáticas a una velocidad considerable, pero las realiza de manera secuencial, una operación por núcleo, o al menos, una operación a la vez. En contraste, las tarjetas gráficas (GPU), como se muestra en la Fig. 2, tienen muchos más núcleos, lo que les permite realizar un mayor número de operaciones en el mismo período de tiempo. La GPU, o Unidad de Procesamiento Gráfico, es un coprocesador diseñado específicamente para el procesamiento de imágenes. Con miles de núcleos optimizados para trabajar en paralelo en operaciones sencillas, la GPU está especialmente preparada para el cálculo matricial. Esencialmente, al ser un procesador adicional, su función es aliviar la carga del CPU y aumentar el rendimiento de la computadora. Toda esta potencia de cálculo aritmético puede aprovecharse para la ejecución de algoritmos de aprendizaje profundo.

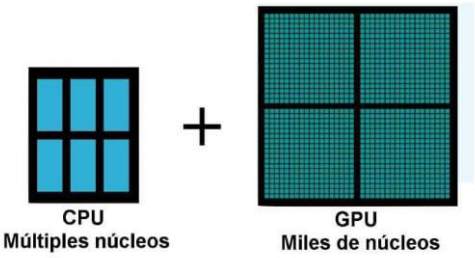

Fig. 2. Comparación entre núcleos de CPU y núcleos de GPU

En este trabajo, la implementación de los distintos modelos se llevó a cabo utilizando el siguiente hardware:

- Procesador: Intel Core i7 4770
- Memoria RAM: 8GB (dual-channel)
- GPU integrada: Intel HD 4600

2.4 - Implementación de los modelos

Se desarrolló un algoritmo de detección de objetos reciclables en imágenes utilizando un conjunto de datos de 15270 imágenes divididas en 6 clases: Papel (2350 imágenes), Metal (2765 imágenes), Vidrio (2348 imágenes), Plastico (2591 imágenes), Orgánico (2497 imágenes), No-Reciclable (2719 imágenes). Así mismo, se analizaron modelos binarios, para lo cual se utilizó una parte del dataset mencionado de 6 clases, re-acomodando y re-etiquetando las imágenes, para obtener un nuevo dataset dividido en dos clases: Reciclable y No-Reciclable. En el caso de los modelos binarios, se utilizaron un total de 8340 imágenes muy bien balanceadas en ambas clases.

El set de datos utilizado incluye imágenes descargadas de diversos sitios de internet, tales como Reddit, Kaggle y Github entre otros, y también imágenes tomadas manualmente por los autores de este trabajo.

El objetivo del algoritmo desarrollado es detectar en tiempo real si las imágenes contienen uno o más objetos de las clases antes mencionadas. Para etiquetar las imágenes, se asignó un nombre a cada objeto que representa su clase, y se guardó un archivo XML para cada imagen con las coordenadas de los objetos ubicados y su clase correspondiente. En la Fig. 3 se muestra el proceso de etiquetado de las imágenes utilizadas para armar el dataset con el que se implementaron los modelos de redes neuronales convolucionales. En la fig. 3-A se muestra el proceso de etiquetado de una imagen que contiene vidrio, en la fig. 3-B se muestra el etiquetado de una imagen que contiene metal y en la fig. 3-C se muestra el etiquetado de una imagen multiclase.

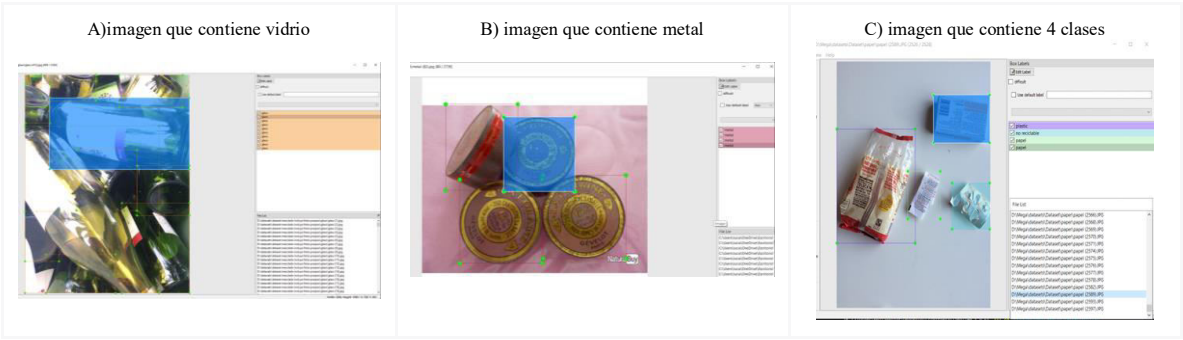

Fig. 3. Etiquetado de imágenes con múltiples clases.

Si bien se probaron modelos implementados desde cero, para aprovechar el conocimiento y eficiencia adquirida por modelos ya probados, ganando además tiempo de ejecución de entrenamiento, se decidió utilizar el método de transfer learning (aprendizaje por transferencia) basados en redes pre-entrenadas. Luego de diversas pruebas se seleccionó el algoritmo YOLO para realizar el entrenamiento. La elección se basa en la precisión y velocidad de detección, a la buena adaptación que tiene en escenarios específicos, las facilidades que provee para el uso e implementación, la eficiencia y la cantidad de recursos con los que dispone.

También, se utilizó Google Colab para desarrollar el modelo, y se crearon tres directorios: uno para almacenar todas las imágenes y archivos XML, otro para las imágenes utilizadas en la etapa de entrenamiento (80% de las imágenes totales) y otro para las imágenes utilizadas en la etapa de validación (20% de las imágenes totales).

Así mismo, se testeó cada uno de los modelos probados con un set de 100 imágenes nuevas, las cuales no eran conocidas previamente por los modelos.

Se generó un archivo CSV a partir de los datos XML para cada conjunto de datos. También se aplicó la técnica de aumento de datos (data augmentation) aplicada a la generación/expansión de los datos de entrenamiento logrando aumentar el set de datos en un 20%. El uso de esta técnica permitió incrementar significativamente la cantidad de imágenes que se dispone, aplicando en ellas pequeños cambios, a fin de evitar el sobreajuste y mejorar la generalización del modelo.

Durante el desarrollo de la aplicación se llevaron a cabo entrenamientos de diversos modelos de redes neuronales convolucionales. Estos modelos se sometieron a diferentes enfoques de entrenamiento, algunos de ellos utilizando únicamente datos correspondientes a objetos reciclables y no reciclables (enfoque binario). Luego se realizaron entrenamientos con un conjunto de datos que incluían todas las categorías de reciclables y por último, entrenamientos abarcando tanto materiales orgánicos como no reciclables, además de las cuatro categorías de reciclables (vidrio, metal, papel y plástico).

En la Fig. 4 se presenta una comparación entre la cantidad de etiquetas utilizadas en dos de los enfoques mencionados. Esto permite visualizar la diferencia en la cantidad de categorías consideradas en el entrenamiento de los modelos.

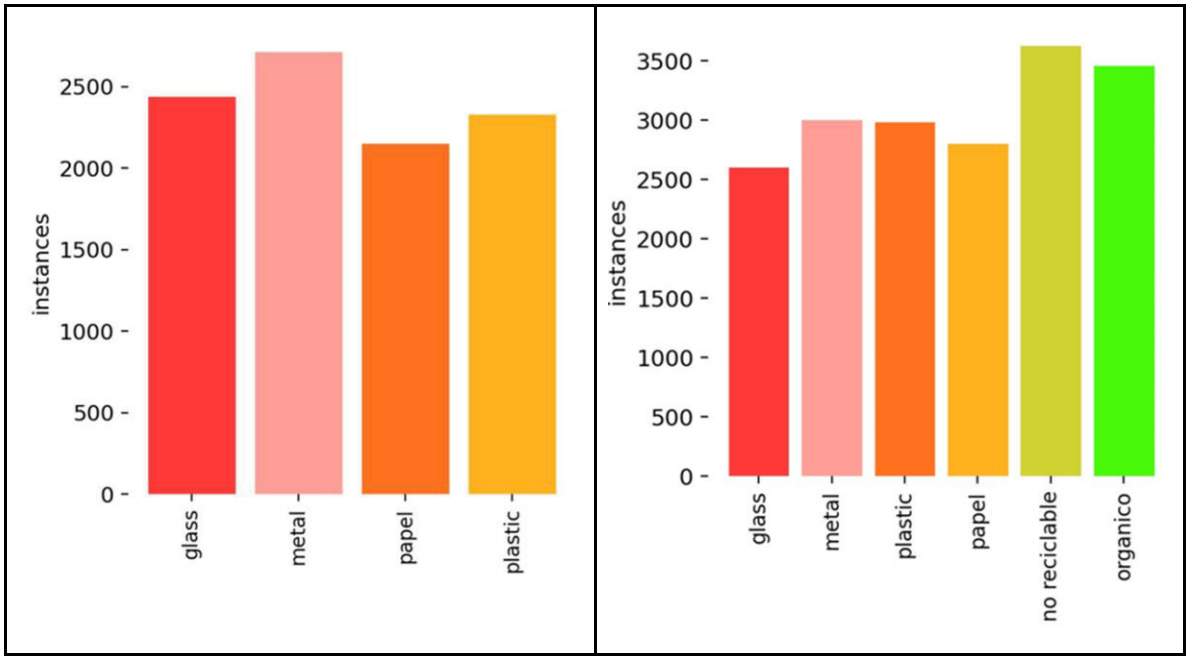

Fig. 4. Comparación de los dos enfoques utilizados

Para cada uno de los modelos se entrenó el algoritmo durante 100 épocas. Una vez finalizado el entrenamiento, se utilizó la biblioteca Matplotlib para graficar la precisión del modelo a lo largo del tiempo.

El mismo set de datos fue utilizado de forma similar para realizar pruebas con TensorFlow y la Api de alto nivel Keras. Con este Framework se probó además del modelo ResNet50, el modelo MobileNet y XceptionV3.

2.5 - Evaluación del rendimiento de los modelos

Para evaluar el rendimiento de un modelo de detección de objetos, primero se debe comprender cuáles son sus objetivos:

✓ Clasificación: se identifica si un objeto está presente en la imagen y la clase del objeto.

✓ Localización: se predicen las coordenadas del cuadro delimitador alrededor del objeto cuando el mismo está presente en la imagen. Aquí se comparan las ground-truth y los bounding box predichos.

Por lo tanto, para evaluar un modelo de detección de objetos, es necesario evaluar el rendimiento tanto de la clasificación como de la localización del uso de cuadros delimitadores en la imagen. Para ello, se utilizó la métrica Intersección sobre Unión (IoU), la cual indica el grado de solapamiento entre datos reales de la imagen y la predicción realizada por el modelo. Se puede establecer un valor umbral para que la IoU determine si la detección de objetos es válida o no. Por ejemplo, si el valor umbral de IoU es 0.5 entonces:

✓ Si IoU $\geq$0.5, se clasifica la detección de objetos como verdadero positivo (TP).

✓ Si IoU <0.5, se considera una detección incorrecta y se clasifica como falso positivo (FP).

✓ Además, cuando un ground-truth está presente en la imagen y el modelo no puede detectar el objeto, se clasifica como falso negativo (FN).

✓ Mientras que la clasificación como verdadero negativo (TN) se da en cada parte de la imagen donde no se predice un objeto. Esta métrica no es útil para la detección de objetos, por lo que generalmente se ignoran los TN.

Además de la métrica IoU, en este trabajo, se utilizaron las métricas Precisión, Recall, Accuracy y F1-Score como métricas para evaluar el rendimiento de los modelos (ecuaciones 1, 2, 3 y 4). La precisión y el recall se calcularon utilizando verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN). Para la exactitud se utilizan, además, verdaderos negativos (TN), mientras que la métrica F1-Score se calcula utilizando la precisión y el recall. Esta métrica aporta generalización a los resultados debido a que tiene en cuenta el desbalance de las clases.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad \text{Ec[1]}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \text{Ec[2]}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative} \quad \text{Ec[3]}$$

$$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} \quad \text{Ec[4]}$$

Para el acierto de cada objeto detectado por el modelo en la imagen se consideraron todos los cuadros delimitadores predichos con un acierto por encima de cierto umbral. Los cuadros delimitadores por encima del valor de umbral se consideraron cuadros positivos y todos los cuadros delimitadores predichos por debajo del valor de umbral se consideraron negativos.

En la fig. 5 se presentan algunas métricas obtenidas durante el entrenamiento del modelo de 6 clases utilizando YOLO v8.

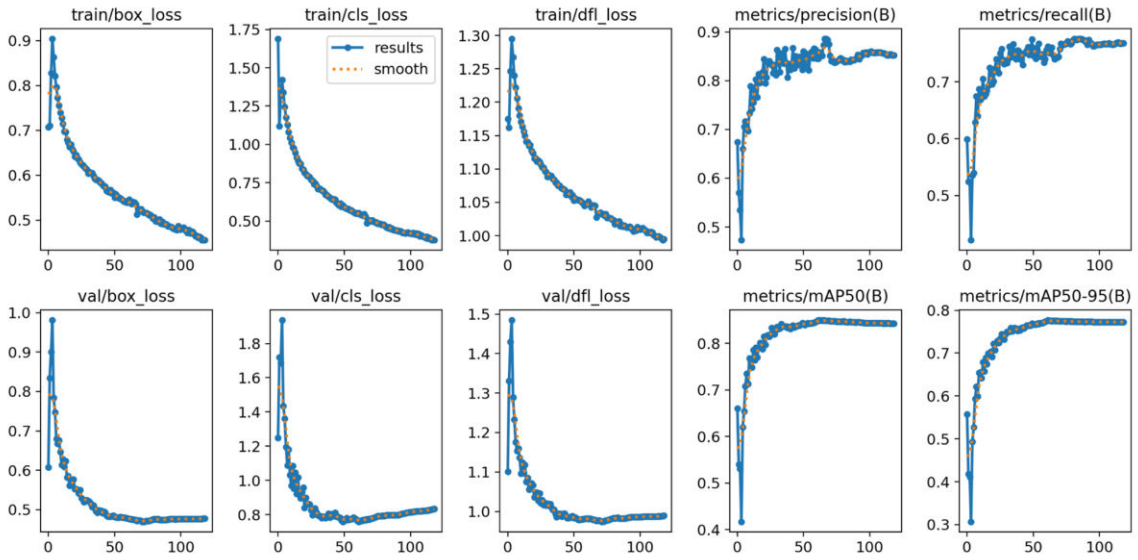

Fig. 5. Métricas del entrenamiento durante 100 épocas

3 - Resultados Obtenidos

3.1 - Resultados

En este trabajo, se desarrollaron 40 modelos diferentes de redes neuronales convolucionales, 25 de los cuales fueron entrenados desde cero, mientras que para los 15 modelos restantes se utilizó transfer learning con redes pre-entrenadas.

El entrenamiento de los modelos duró aproximadamente dos meses de horas máquina, con un tiempo promedio de entrenamiento de 12 horas para los modelos más simples y 48 horas para los modelos más complejos, resultando en un promedio general de 30 horas por modelo propuesto, utilizando el hardware mencionado en la sección 2.3.

Durante el desarrollo de los primeros modelos, se identificó el overfitting (sobreajuste) como uno de los principales problemas, con buenos resultados para el set de entrenamiento y malos para el set de validación. Para mejorar estos resultados, se probaron pequeñas variaciones en los hiperparámetros de la red y la profundidad de los modelos. Además, se implementaron técnicas como data augmentation, early stopping y dropout para reducir el sobreajuste del modelo. Se utilizaron tres funciones "callbacks": "EarlyStopping", "ModelCheckpoint" y "ReduceLROnPlateau" para monitorear y modificar los hiperparámetros durante el entrenamiento.

Se aplicó la técnica de dropout que varió entre un 25% y un 40% de las neuronas de la capa completamente conectada del modelo, variando el porcentaje de neuronas al que se le aplicaba, lo que permitió evitar que el modelo memorizara un resultado.

Estas modificaciones permitieron una mejora considerable en el porcentaje de acierto de los modelos propuestos, con valores superiores al 70% para el set de datos dividido en seis clases. Luego, se utilizaron modelos pre-entrenados, logrando resultados superiores al 80% para el mismo set de datos.

En primer lugar, se presentan los resultados obtenidos para los dos mejores modelos binarios implementados (objetos reciclables y no reciclables). La Fig. 6-A muestra la matriz de confusión correspondiente al modelo binario con entrenamiento desde cero, mientras que la Fig. 6-B muestra la matriz de confusión del modelo con transfer learning.

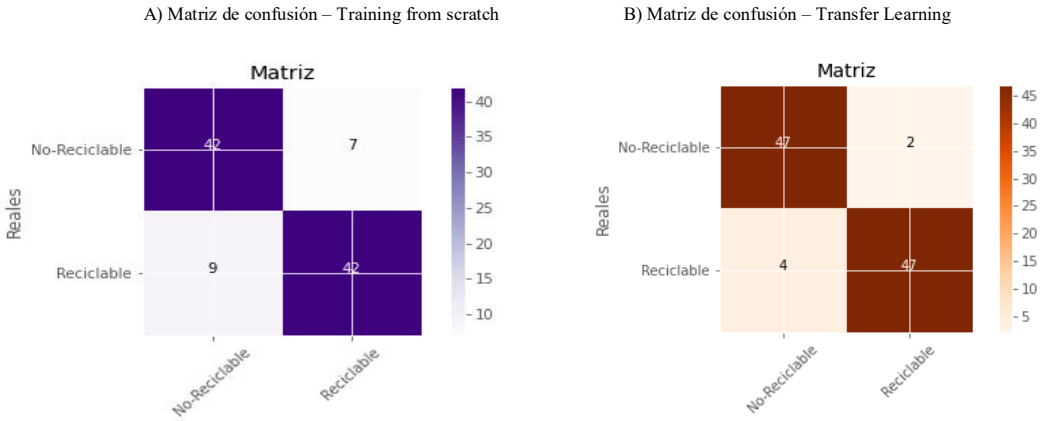

Fig. 6. Matrices de confusión modelos binarios

A continuación, se presentan los resultados obtenidos para los modelos denominados modelo 1 y modelo 2. El modelo 1 corresponde a objetos solo reciclables (papel, vidrio, metal y plástico), mientras que el modelo 2, además de los elementos del modelo 1, incluye a las clases elementos orgánicos y objetos no reciclables.

En la Fig. 7 se observa la matriz de confusión normalizada correspondiente al modelo 1. El objetivo es que la matriz sea lo más diagonal posible, lo cual indica que el modelo apenas se ha confundido en las distintas categorías de imágenes que conforman el dataset. En un color celeste claro se puede observar que el modelo ha hecho alguna predicción errónea, detectando como residuos algo que en realidad era fondo. Estos valores no son significativos teniendo en cuenta que los valores correctos son del 0.94, es decir una probabilidad muy alta cercana a 1.

En la matriz de confusión de la Fig, 7, Background se refiere a objetos de fondo que no pertenecen a ninguna de las clases pero que el modelo detecta como una de ellas.

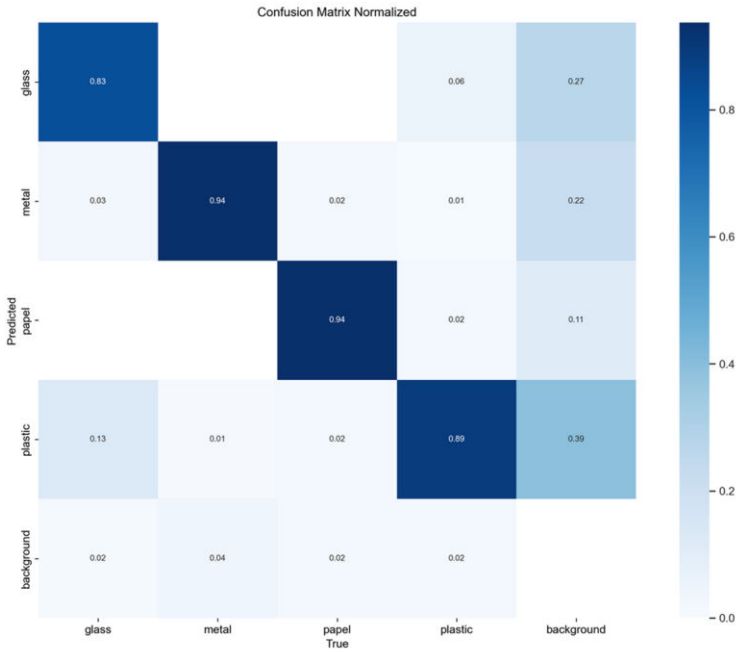

Fig. 7. Matriz de confusión normalizada del modelo 1

A partir de la Fig. 7 se observa que las clases vidrio (glass) y plástico (plastic) están por debajo del rendimiento óptimo en comparación con las otras clases, ya que poseen un 0.83 y 0.89 respectivamente, en comparación con el papel y metal que tienen un 0.94. Esto puede deberse a que en objetos transparentes estos materiales tienen muchas similitudes.

La Fig. 8 muestra la matriz de confusión normalizada correspondiente al modelo 2. Nuevamente aquí, Background se refiere a objetos de fondo que no pertenecen a ninguna de las clases pero que el modelo detecta como una de ellas.

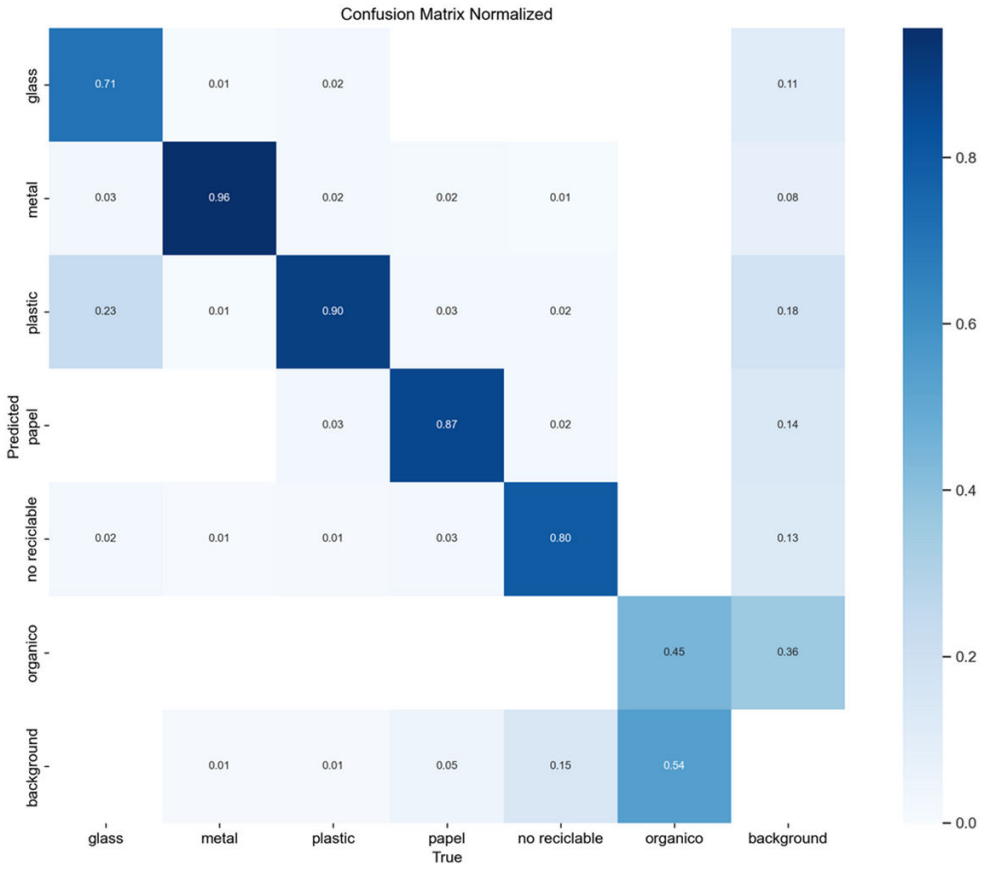

Fig. 8. Matriz de confusión normalizada del modelo 2

Las Fig. 9-A y Fig. 9-B ilustran las matrices de confusión correspondientes al modelo 2, en los casos de entrenamiento desde cero, y con transfer learning, respectivamente.

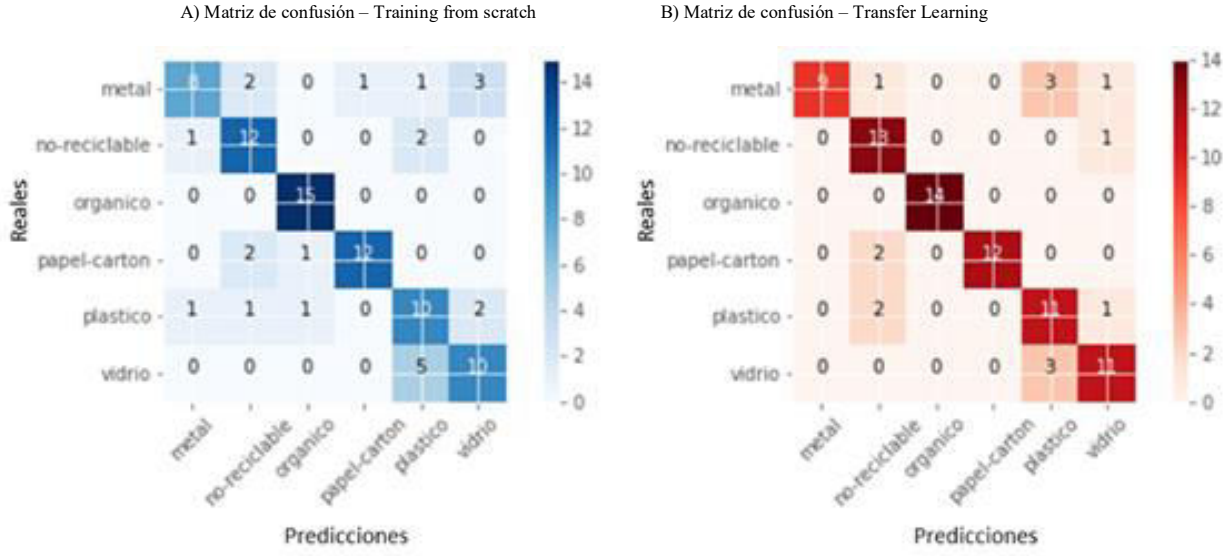

Fig. 9. Matrices de confusión modelos multiclase

La Tabla 1 resume los resultados obtenidos de las métricas calculadas para los mejores modelos binarios y modelo 2, utilizando el dataset inicial.

Modelo binario entrenado desde cero: Precision ≈ 0.82 Recall ≈ 0.85 F1-score ≈ 0.83 Accuracy ≈ 0.84	Modelo binario con aprendizaje por transferencia: Precision ≈ 0.92 Recall ≈ 0.95 F1-score ≈ 0.93 Accuracy ≈ 0.94
Modelo 2 (6 clases) entrenado desde cero: Accuracy ≈ 0.744	Modelo 2 (6 clases) con transfer learning: Accuracy ≈ 0.833

Tabla 1 - Métricas obtenidas para los mejores modelos

La métrica F1-score se calculó para cada una de las clases por separado. Se muestra a continuación como ejemplo el cálculo para las clases “Vidrio” y “Orgánico” para el modelo 2 entrenado desde cero, y para las clases “Metal” y “Orgánico” para el modelo 2 entrenado con transfer learning.

F1-vidrio ≈ 0,66 (clase desbalanceada respecto al resto)
F1-organico ≈ 0,937 (clase con buen balance)
F1-metal ≈ 0,781 (clase desbalanceada respecto al resto)
F1-organico =1 (clase con buen balance)

Luego de la ampliación y de los notables ajustes en el dataset, se obtuvieron importantes mejoras en las métricas, tal como se muestra en la Tabla 2, en donde se observan los resultados aproximados de dos de los mejores modelos obtenidos para el modelo 1 y el modelo 2. En dicha tabla, la métrica mAP50 se refiere a la precisión media calculada con un umbral de IoU de 0.5.

Métrica	Modelo 1	Modelo 2
Precisión	0.914	0.889
Recall	0.879	0.909
F1-score	0.91	0.92
mAP50	0.939	0.945

Tabla 2. Comparación de resultados de los modelos 1 y 2

Las Fig. 10 y la Fig. 11 muestran las gráficas Precisión-Recall (P-R) para los modelos 1 y 2, respectivamente. La curva P-R evalúa la precisión y la proporción de objetos detectados de forma correcta. Los valores obtenidos se acercan exitosamente al valor ideal 1.0, identificados con la línea azul, tanto en el eje de precisión como en el de recall. Se considera que se mantiene mejor equilibrio entre precisión y sensibilidad, es decir, mejor curva P-R cuanto mayor es el área por debajo de la curva.

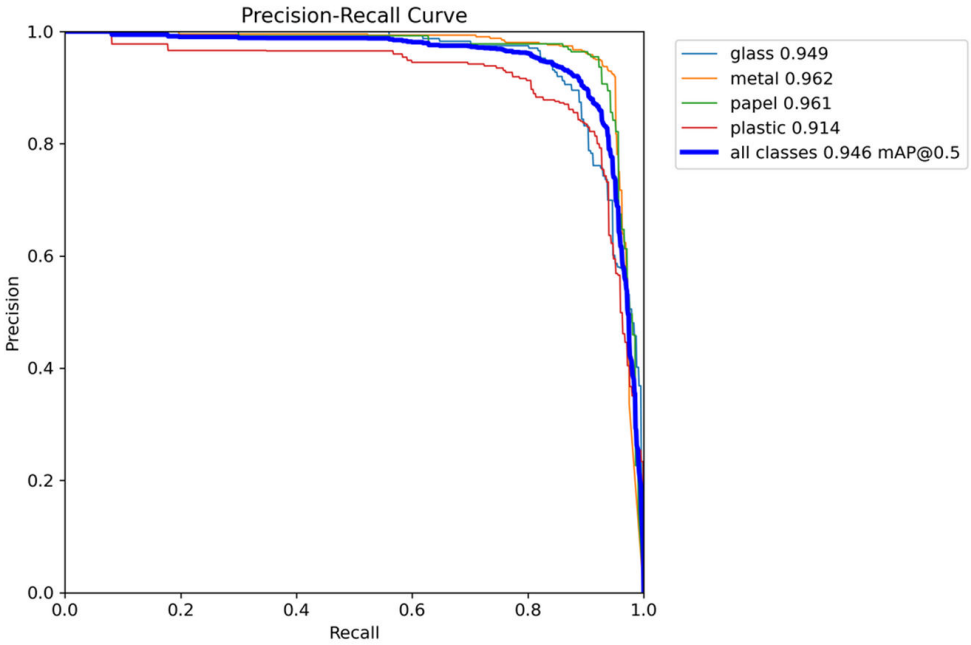

Fig.10. Curva precisión - recall (P-R) del modelo 1

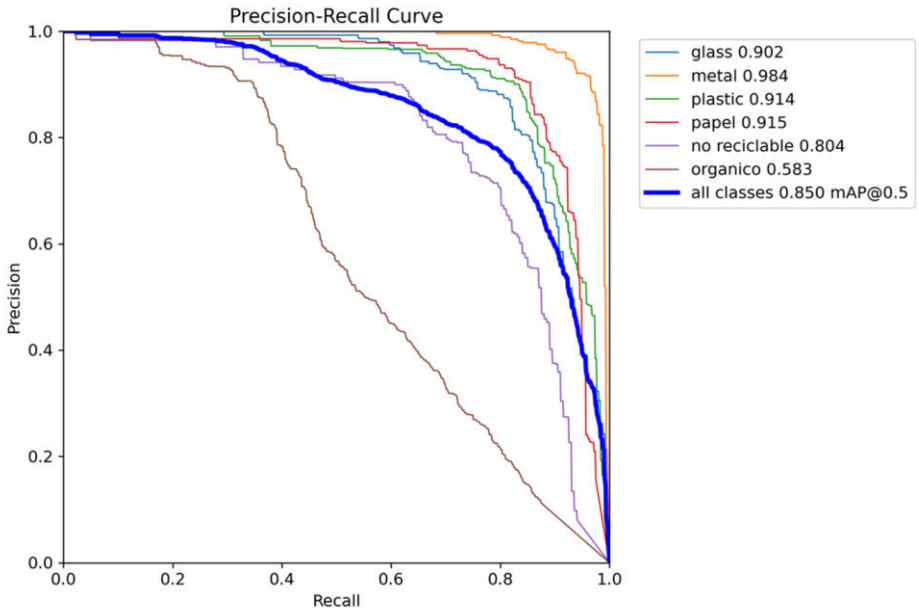

Fig. 11. Curva precisión - recall (P-R) del modelo 2

Los modelos que obtuvieron los mejores resultados (para los modelos 1 y 2), fueron probados en una minicomputadora Raspberry Pi 3 con el módulo Pi-Cam en donde se implementó el prototipo de una aplicación para realizar la detección de objetos en tiempo real. En las siguientes figuras, (Fig. 12 y 13), se presentan a modo de ejemplo algunas capturas de los videos de prueba, en donde se puede observar el alto porcentaje de certeza con que se clasifican los diferentes objetos. En la Fig. 12 se observa la vista en perspectiva, donde los objetos de papel y vidrio son detectados con una exactitud del 94% y 92%, respectivamente, mientras que el metal y plástico poseen una exactitud del 86%. Por otro lado, en la Fig. 13 se observa la vista desde arriba, en cuyo caso el objeto de plástico es quien presenta la mejor exactitud (89%), mientras que los objetos de papel, vidrio y metal reducen fuertemente su exactitud a valores del 78%, 82% y 68%, respectivamente. En el caso del metal y el vidrio, esta disminución es debido fundamentalmente a que en la imagen solo se observa un círculo, y las principales características se obtienen de las diferentes tonalidades.

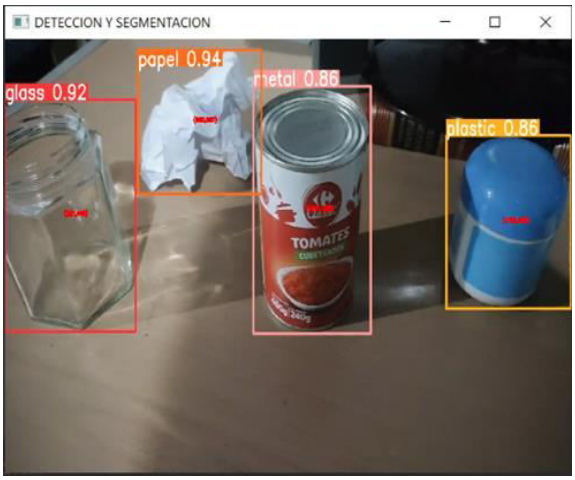

Fig. 12. Captura ejemplo de clasificación 1

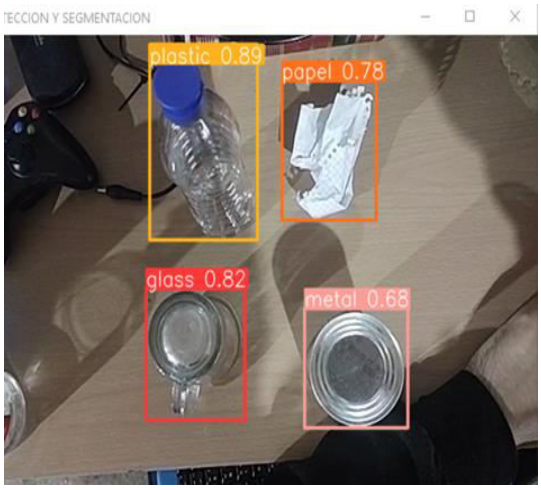

Fig. 13. Captura ejemplo de clasificación 2

La exactitud se podría mejorar analizando el dataset e identificando sí existen datos que pudieran estar causando el problema o incluso cambiar la configuración de los parámetros para el entrenamiento. Otra de las posibles mejoras es realizar un aumento de las imágenes y etiquetas sobre las clases plástico y vidrio. Aun así, se debe destacar que los resultados obtenidos son muy buenos y que en pruebas realizadas en tiempo real con movimiento y fondo variable las predicciones fueron altamente satisfactorias.

4 - Conclusiones

Para la realización del trabajo se seleccionó a Python como lenguaje de programación dado que cuenta con una gran variedad de bibliotecas abocadas a la inteligencia artificial. Por consiguiente, el dilema estuvo en la elección entre las dos bibliotecas de aprendizaje automático de código abierto más potentes, Keras o PyTorch. Finalmente, se determinó que Keras tiene más información y soporte acerca de modelos de clasificación de objetos que para modelos de detección de objetos y es una librería de más alto nivel. Por el contrario, para la creación de un modelo de detección de objetos con Pytorch se debían codificar y utilizar librerías debido a que tiene una API de bajo nivel además de proveer más información de interés. Dado que las dos bibliotecas tienen un apoyo comunitario muy fuerte, durante la implementación del sistema se realizaron pruebas con ambas.

Otro punto clave, fue la elección entre la creación de un modelo desde cero o la creación de un modelo a partir de uno pre-entrenado. Considerando la limitante en cantidad de datos (imágenes) que se requieren para el entrenamiento de un modelo de detección de objetos y sabiendo adicionalmente que se deben detectar 6 tipo de objetos por imagen, se optó por un modelo pre-entrenado aprovechando lo brindado por éste en la etapa de extracción de características.

Otra situación crítica fue la elección de los hiperparámetros para poder controlar el proceso de entrenamiento de los modelos analizados, ya que es bastante difícil encontrar los valores óptimos que se ajusten.

En referencia a los resultados, este estudio logró alcanzar con éxito los objetivos inicialmente propuestos. Los modelos exhibieron métricas prometedoras, con tasas de precisión superiores al 90% en algunos casos. Se atribuye este logro al meticuloso esfuerzo invertido en la elección de los hiperparámetros y en la construcción de un dataset diverso, que abarcó una amplia variedad de entornos y que contó con una cantidad significativa de datos y fotografías cuidadosamente seleccionadas.

Se realizaron diversas pruebas, obteniéndose muy buenos resultados en la detección de objetos reciclables. Durante las mismas se observó que hubo una cantidad muy reducida de resultados falsos positivos, lo que permite pensar en tareas futuras que permitan optimizar el dataset y perfeccionar el algoritmo mediante pequeños ajustes durante el entrenamiento. Además, se logró con éxito la implementación de la detección en una minicomputadora.

Como tareas a futuro se pretende mejorar los resultados obtenidos e integrar los modelos con un brazo robótico, (modelo LewanSoul LeArm 6DOF), para sumar a la detección de objetos la capacidad de separarlos. Es por este motivo que el algoritmo muestra en pantalla el centro de los objetos detectados (ver Fig. 12), ya que se requiere esta información para poder mapear las coordenadas del objeto en la imagen con la ubicación sobre una cinta transportadora. Asimismo, se pretende realizar pruebas intensivas de los modelos en sistemas de cómputo de bajas prestaciones, para determinar su aplicabilidad en sistemas de bajo consumo.

Finalmente, estos resultados abren la puerta a futuras investigaciones e implementaciones de sistemas de automatización impulsados por inteligencia artificial que permitan aportar en la resolución de los desafíos ambientales y de sustentabilidad actuales. La utilización de la aplicación permitirá, entre otras posibilidades, ser un recurso importante en el sector público para automatizar la separación de residuos apoyándose en la visión artificial. También podrá integrarse en robots móviles para la recolección de residuos en espacios públicos o en cestos de residuos inteligentes que utilicen la visión artificial para realizar la tarea de separación en origen facilitando su recolección y futuro reciclaje.

Referencias

[1] Maquituls España (2017), La importancia del reciclaje. Cuidemos el Medio Ambiente, https://www.maquituls.es/noticias/la-importancia-del-reciclaje-cuidemos-el-medio-ambiente/#:~:text=El%20reciclar%20o%20el%20reciclaje,de%20manera%20continua%20al%20planeta., recuperado 10 de mayo 2020.

[2] Diario El Cronista (2018), Producción de basura: cuál es la realidad en Argentina y que se podría hacer, https://www.cronista.com/responsabilidad/Produccion-de-basura-cual-es-la-realidad-en-Argentina-y-que-se-podria-hacer-20180302-0075.html, recuperado 10 de mayo 2020.

[3]Naciones Unidas. (2016). Por qué importa [PDF]. Recuperado de https://www.un.org/sustainabledevelopment/es/wp-content/uploads/sites/3/2016/10/12_Spanish_Why_it_Matters.pdf

[4]Gobierno de Argentina. (2023). Informe país [PDF]. Recuperado de https://www.argentina.gob.ar/sites/default/files/informe_pais_baja.pdf

[5] Joshi, J., Reddy, J., Reddy, P., Agarwal, A., Agarwal, R., Bagga, A., Bhargava, A. (2016). Cloud computing based smart garbage monitoring system. In: Electronic Design (ICED), 3rd International Conference on, 6 p. doi: 10.1109/ICED.2016.7804609

[6] J. D. Giraldo Quiñones, "CLASIFICADOR DE RESIDUOS SÓLIDOS HACIENDO USO DE DEEP LEARNING", Tesis de grado, Departamento de automática y mecatrónica, Universidad Autónoma de Occidente, Santiago de Cali, 2022.

[7] Marina Martin Moreno, "Desarrollo de un sistema inteligente para la clasificación de residuos sólidos", Tesis de grado, Departamento de Ingeniería Audiovisual y Comunicaciones, Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, Universidad Politécnica de Madrid, 2022.

[8] K. Erin, B. Bingöl, B. Ború, "YOLO – Based Waste Detection", Journal of Smart Systems Research (JOINSSR) , vol. 3, issue 2, pp. 120-127, December 2022

[9] J. F. Palacios, S. Vitery, L. F. Giraldo, "Deep Learning-Based Garbage Bags and Potholes Detection Model Using Raspberry Pi", Department of Electric and Electronic Engineering, Universidad de Los Andes, Bogotá D.C., 2021.

[10] Sarkar, D. (2018). Hands-On Transfer Learning with Python. Packt Publishing.

[11] Wolfgang Ertel, Introduction to Artificial Intelligence, second edition, Springer International Publishing AG 2017.

[12]Kevin Murphy, Machine Learning - A probabilistic perspective, University of Cambridge, 2012

[13]Zoubin Gharamani, Automatic Machine Learning, University of Cambridge, 2018

[14]Miroslav Kubat, An Introduction to Machine Learning, second edition, Springer International Publishing AG 2017.

[15]Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT 2017.

[16] Nikhil Buduma, Fundamentals of Deep Learning, Editorial O'reilly. 2017

[17] Sandro Skansi, Introduction to Deep Learning – From Logical Calculus to Artificial

[18] Robert Laganière, OpenCV 2 Computer Vision Application Programming Cookbook, Editorial Packt Publishing, Birmingham, UK. 2011.

[19] Bootcamp AI, 2019, "Intro a las redes neuronales convolucionales", https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8#:~:text=Las%20redes%20neuronales%20convolucionales%20es,poder%20diferenciar%20unos%20de%20otros.

[20] Calvo, Diego, 2017, "Red Neuronal Convolucional CNN", https://www.diegocalvo.es/red-neuronal-convolucional/.

[21] Andreas Muller, Sarah Guido, Introduction to Machine Learning with Python, Editorial O'reilly, 2016.

[22] Francois Chollet, Deep Learning with Python, MEAP edition, Manning Publications 2017.

[23] Aurelien Gerón, Hands-On Machine Learning withs cikitLearn & TensorFlow, Editorial O'reilly, 2017

[24] Tom Hope, Yehezkel Resheff, Itay Lieder, Learning TensorFlow, Editorial O'reilly. 2017

[25] TensoFlow (2020), TensorFlow API documentation, https://www.tensorflow.org/api_docs/, recuperado 01 octubre de 2020.

[26] Keras io (2020), Keras API references, https://keras.io/api/, recuperado 01 octubre de 2020.

[27] Géron, A. (2019). Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow: Conceptos, herramientas y técnicas para conseguir sistemas inteligentes. Barcelona, España: Marcombo.