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

Braian Pezet[1], Lucia Osés[1], Marcelo Cappelletti[1,2], Mauro Salina[1], 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, maurosalina85} @gmail.com {josio, mcappelletti, martin.morales } @unaj.edu.ar

Abstract. Este trabajo se enfoca en el desarrollo de un sistema de clasificación de objetos para ser utilizado en un recolector de residuos inteligente, empleando técnicas de Aprendizaje Profundo (Deep Learning). Se crearon modelos de redes neuronales convolucionales (CNN) capaces de identificar distintos objetos reciclables en diferentes imágenes, en tiempo real. Se llevaron a cabo pruebas con clasificación binaria (reciclable - no reciclable) y clasificación multiclase (plástico, vidrio, metal, papel-cartón, orgánico, no reciclable). También se realizaron pruebas utilizando modelos pre-entrenados con aprendizaje por transferencia (Transfer Learning) para comparar resultados. La implementación de estos modelos se llevó a cabo utilizando el lenguaje de programación Python, aprovechando el Framework de backend TensorFlow y la librería de alto nivel Keras. 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). Este sistema permite analizar en tiempo real los fotogramas capturados por la cámara y aplicar el modelo de clasificación de manera instantánea, accediendo de esta manera a las coordenadas de dichos objetos en el fotograma para poder recolectarlos y separarlos para su posterior reciclaje.

Keywords: Machine Learning, Deep Learning, IoT, sistema de reciclaje, procesamiento de imágenes.

1 - Introducción

La protección del medio ambiente y la importancia del reciclaje son temas de gran relevancia en la sociedad actual. El reciclaje es fundamental para reutilizar elementos u objetos ya utilizados, los cuales, de otra manera, terminarían como desechos y contribuirían al aumento de la formación de basura y al daño ambiental permanente [1]. En nuestro país, la producción de basura es alarmante, con una tonelada de basura producida cada dos segundos y una gran parte 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, plásticos, vidrio, metales y pilas [2]. Es esencial realizar la separación en origen para minimizar la generación de residuos y la contaminación. Sin embargo, en nuestro país, 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]. Por esta razón, en este trabajo se propone desarrollar un sistema de visión por computadora que permita detectar y clasificar objetos reciclables para minimizar la cantidad de residuos que se generan diariamente.

En la literatura, se pueden encontrar trabajos previos relacionados con recolectores de residuos inteligentes [3-4]. En [3] se plantea un basurero inteligente para ciudades, que consiste en contenedores con hardware basado en un microcontrolador y un sensor infrarrojo. Los datos son enviados mediante un módulo GPRS. En [4] se propone una solución similar, con la diferencia en que utiliza tags RFID para verificar y controlar a los vehículos recolectores. Los vehículos también tienen un módulo GPRS; además, se usan aplicaciones móviles para la monitorización de los contenedores y la implementación de un servicio en la nube llamado ThingSpeak para almacenar los datos.

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 [5]. Para llevar a cabo este proyecto, se utilizó la Inteligencia Artificial (IA), que ha tenido un avance significativo en la última década en áreas como la detección de objetos y la clasificación de imágenes [6]. Esto se debe, en gran parte, a las nuevas técnicas de Aprendizaje Automático (Machine Learning) y Aprendizaje Profundo (Deep Learning), 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 [7-12].

El 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 [9]. El desafío reside en construir un modelado automático de una función matemática en la que se parte de una entrada y se obtiene una salida. Por otro lado, el Deep Learning es un subcampo del Machine Learning que utiliza redes neuronales artificiales compuestas por varios niveles jerárquicos para realizar procesos de aprendizaje automático [12]. Cada nivel aprende patrones cada vez más complejos, generando información más sofisticada.

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.

Para llevar a cabo el desarrollo del algoritmo se utilizó la Inteligencia Artificial (IA) para poder automatizar la toma de decisiones, la resolución de problemas y el aprendizaje. Esto implica utilizar técnicas de Machine Learning (Aprendizaje Automático) y Deep Learning (Aprendizaje Profundo).

Los avances en IA han impulsado el uso de Big Data debido a su habilidad para procesar enormes cantidades de datos y proporcionar ventajas comunicacionales, comerciales y empresariales que la han llevado a posicionarse como la tecnología esencial de las próximas décadas.

Machine Learning utiliza algoritmos para analizar datos, aprender de esos datos y tomar decisiones basadas en lo aprendido [1]. El 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 tanto, cuando un sistema de Deep Learning tiene que reconocer un objeto, lo compara con aquellos que ya conoce.

Actualmente existe un campo de la Inteligencia Artificial que viene prosperando: la Visión Artificial [2]. 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 un proceso.

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”, 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, 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, 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.

El uso de la visión artificial crece rápidamente gracias al descubrimiento de 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.

El trabajo busca enfocarse en el desarrollo de una aplicación de software encargada de realizar una búsqueda de objetos en imágenes a partir de una clasificación de las mismas en tiempo real, mediante la cual se buscará detectar la presencia de determinados objetos en dicha imagen. El desarrollo se enfoca específicamente en el uso de redes neuronales convolucionales, las cuales han demostrado ser las más eficientes en el área del procesamiento de imágenes. Específicamente, el software permite realizar la tarea de detección de objetos reciclables en imágenes o video.

El tema presenta un marcado interés tecnológico y social. Por lo tanto, la utilización del software combinando ambos permite, entre otras posibilidades, por ejemplo, ser un recurso importante en el área estatal para que los gobiernos puedan disminuir la tasa de residuos apoyándose en la visión artificial para garantizar el reciclado. Otro ejemplo en el que el software puede ser de utilidad es en los distintos hogares o instituciones que pueden contar con cestos de residuos inteligentes que utilizan la visión artificial para realizar la tarea de separación en origen facilitando su recolección y futuro reciclaje.

2 - Marco Teórico

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. Y en el caso de trabajar con imágenes como datos de entrada se presenta una problemática, recordemos que para una computadora una imagen es representada como una matriz de píxeles, por lo tanto cada píxel de la imagen de entrada estaría conectado a cada neurona de la primer capa oculta de la red, y el problema se encuentra en que la cantidad de conexiones necesarias es demasiado grande y esto hace que el uso redes "fully connected" sea prácticamente inviable (incluso para imágenes de tamaño relativamente pequeño en redes muy profundas). Por ejemplo, si esta tarea se intentara realizar con una arquitectura de red neuronal tradicional el poder computacional y 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. Y eso es si solo se tiene 1 color (escala de grises). En una imagen a color, se necesitan 3 canales (red, green, blue) y 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. [21].

2.2 - Arquitectura de las redes neuronales convolucionales

La arquitectura de una red neuronal convolucional (RNC) 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, que comienza con un aplanamiento y continua con una red neuronal tradicional totalmente conectada, en donde se realizará la clasificación final [22].

3 – Herramientas

3.1 Herramientas de Software

A continuación, se describen de manera resumida las herramientas y los procedimientos utilizados en este trabajo, todas estas herramientas son librerías, 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, aún no está disponible para R y Scala.

- **TensorFlow:** Es una librería de código abierto para Machine Learning. Esta librería de computación matemática ejecuta de forma rápida y eficiente gráficos de flujo [15]. 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)[16] y[17].

- ***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 [18].

- ***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.

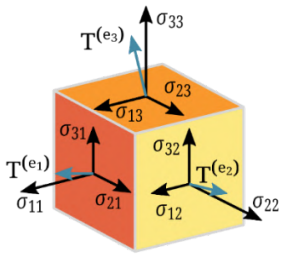


Fig. 1. Representación de un tensor

3.2 Hardware

El uso de técnicas de aprendizaje automático [19], 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.

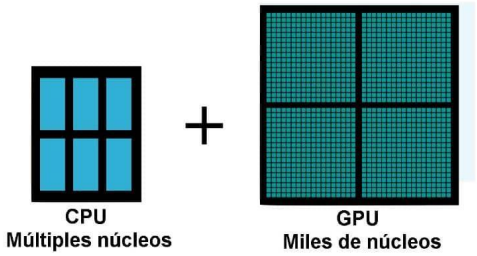


Fig. 2. Comparación entre núcleos de CPU y núcleos de GPU

4 – Implementación

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

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, para los modelos binarios, se utilizó una parte del dataset de 6 clases, reacomodando y reetiquetando las imágenes para dividir en las clases: Reciclable, No-Reciclable. Utilizando un total de imágenes de 8340 imágenes muy bien balanceadas en ambas clases, a su vez, se dividieron los datos en tres partes separando 100 imágenes para el testeo del modelo, 1708 imágenes para la validación y 6532 imágenes utilizadas para el entrenamiento.

El set de datos utilizado es de creación propia del grupo de investigación, en el mismo se incluyen imágenes descargadas de diversos sitios de internet como reddit, kaggle y github entre otros e imágenes tomadas manualmente.

El objetivo del algoritmo es detectar en tiempo real si en las imágenes se encuentra 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. Este etiquetado y la división de las clases fue realizado de forma manual por los miembros del equipo de trabajo que llevó a cabo este trabajo.

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. Se utilizó Google Colab para desarrollar el modelo, y se crearon tres directorios: uno para almacenar todas las imágenes y archivos XML, otro para los datos de entrenamiento (80% de las imágenes totales) y otro para los datos de validación (20% de las imágenes totales), así mismo, se testeo cada uno de los modelos probados con un set de 100 imágenes nuevas, las cuales no eran conocidas 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 disponen aplicando en ellas pequeños cambios, a fin de evitar el sobreajuste y mejorar la generalización del modelo. Se entrenó el algoritmo durante 100 épocas.

Una vez finalizado el entrenamiento, se utilizó la biblioteca Matplotlib para imprimir un gráfico que muestra 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.

5 - Resultados Obtenidos

5.1 - Evaluación de rendimiento

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

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

✓ Localizar: se predicen las coordenadas del cuadro delimitador alrededor del objeto cuando un objeto está presente en la imagen. En esta parte se comparan las ground-truth y los bounding box predichos.

Por lo que, 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 hacerlo se utiliza el concepto de Intersección sobre Unión (IoU). Se puede establecer un valor de umbral para que IoU determine si la detección de objetos es válida o no. Si se establece, por ejemplo, IoU en 0.5:

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

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

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

✓ La clasificación como verdadero negativo (TN) es cada parte de la imagen donde no se predice un objeto. Esta métrica no es útil para la detección de objetos, por lo que se ignora este TN.

Se utilizó Precisión, Recall , Accuracy y F1-Score como métricas para evaluar el rendimiento (ecuaciones 1, 2, 3 y 4). La precisión y la recuperación se calcularon utilizando verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN). Para el acierto se utiliza, además, verdaderos negativos (TN) y la métrica F1 se calcula utilizando la precisión y la recuperación, 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.

5.2 - Resultados

Durante todo el proyecto se desarrollaron alrededor de 40 modelos de redes neuronales convolucionales. Los primeros 25 modelos se entrenaron desde cero, mientras que los últimos 15 modelos utilizaron 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 el inciso 4.

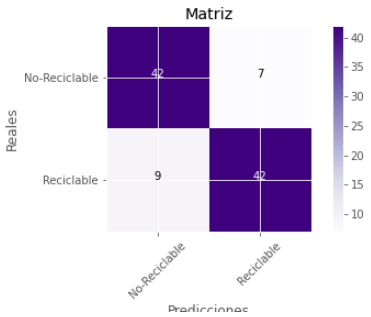
Durante el desarrollo de los primeros modelos, se identificó el overfitting (sobreajuste) como uno de los principales problemas, con buenos resultados para el set de entrenamiento y malos para el set de validación. Para mejorar estos resultados, se probaron pequeñas variaciones en los hiperparámetros 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.

Se presentan a continuación las matrices de confusión obtenidas de los 2 mejores modelos implementados, en la figura 3-A se presenta la matriz de confusión correspondiente al modelo binario con entrenamiento desde cero, y en la figura 3-B la matriz del modelo con transfer learning. Así mismo, en la figura 4-A se representa la matriz de confusión correspondiente al modelo multiclase con entrenamiento desde cero, y en la figura 4-B la matriz del modelo con transfer learning.

A) Matriz de confusión – Training from scratch



B) Matriz de confusión – Transfer Learning

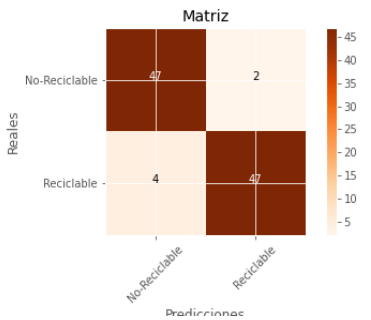


Figura 3. Matrices de confusión modelos binarios

A) Matriz de confusión – Training from scratch

B) Matriz de confusión – Transfer Learning

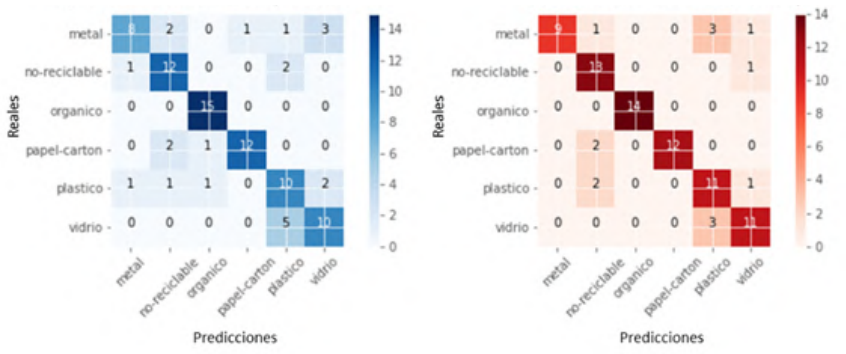


Figura 4. Matrices de confusión modelos multiclase

A continuación, se presentan las métricas obtenidas (tabla 1) para cada uno de los casos, siempre evaluando los modelos que presentaron un mejor resultado.

Para el modelo binario entrenado desde cero: *Precision* ≈ 0,82 *Recall* ≈ 0,85 F1 ≈ 0,83 Accuracy ≈ 0,84	Modelo binario con aprendizaje por transferencia: *Precision* ≈ 0,92 *Recall* ≈ 0,95 F1 ≈ 0,93 Accuracy ≈ 0,94
Modelo de 6 clases entrenado desde cero: Accuracy ≈ 0,744 La métrica F1-score se calculó para cada una de las clases, se muestra a continuación como ejemplo el cálculo para la clase “Vidrio” y “Orgánico” F1-vidrio ≈ 0,66 (clase desbalanceada respecto al resto) F1-organico ≈ 0,937 (clase con buen balance)	Modelos de 6 clases con transfer learning: Accuracy ≈ 0,833 La métrica F1-score se calculó para cada una de las clases, se muestra a continuación como ejemplo el cálculo para la clase “Metal” y “Orgánico” F1-metal ≈ 0,781 (clase desbalanceada respecto al resto) F1-organico =1 (clase con buen balance)

Tabla 1 - Métricas obtenidas para los mejores modelos

En la siguiente figura (figura 5) se presentan algunas métricas obtenidas durante el entrenamiento del modelo de 6 clases utilizando YOLO.

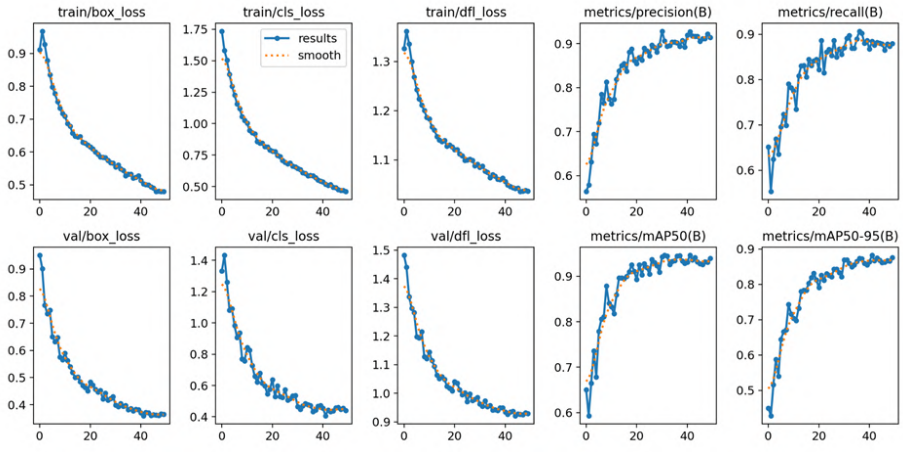


Figura 5. Métricas del entrenamiento durante las primeras 50 épocas.

En las siguientes imágenes (figura 6 y 7) 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:

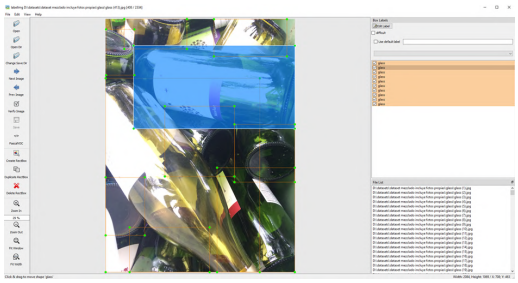


Figura 6. Etiquetado de imagen que contiene vidrio.

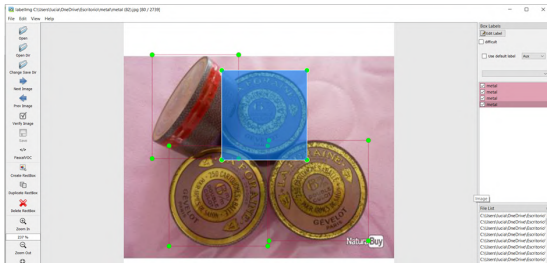


Figura 7. Etiquetado de imagen que contiene metal.

Estos modelos que obtuvieron los mejores resultados 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 se presentan a modo de ejemplo algunos capturas de los videos de prueba. (Figura 8 y 9).

Como tareas a futuro se pretende realizar pruebas intensivas del modelo en sistemas de cómputo de bajas prestaciones, para determinar su aplicabilidad en sistemas de bajo consumo.

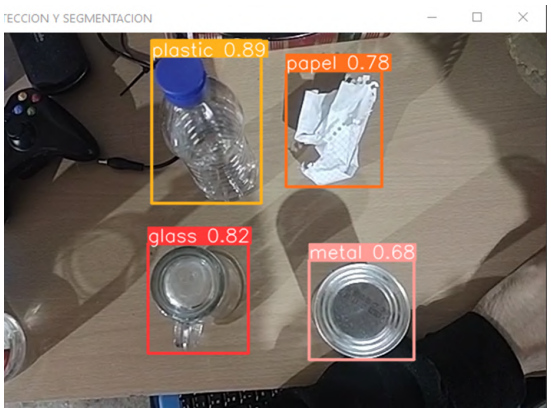


Figura 8. Captura ejemplo de clasificación 1

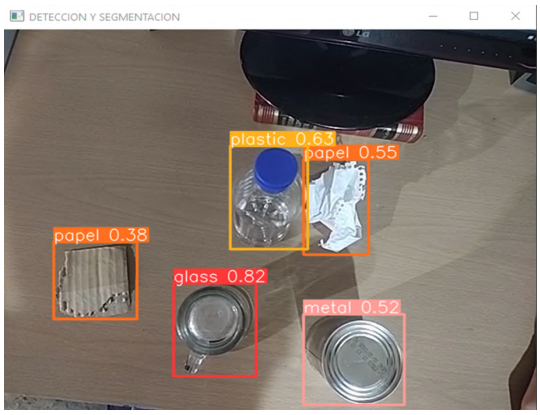


Figura 9. Captura ejemplo de clasificación 2

6 - Conclusiones

Para la realización del trabajo se eligió Python como lenguaje de programación dado que éste tiene 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. Durante la selección 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 Pytorch tiene una API de bajo nivel además de proveer más información de interés. Dado que las dos bibliotecas tienen un apoyo comunitario muy fuerte, se realizaron pruebas con ambas.

Otra problemática fue la elección entre la creación de un modelo desde cero o la creación de un modelo a partir de uno pre entrenado. Por lo que observando la limitante en cantidad de datos (imágenes) que se requieren para un entrenamiento de un modelo de detección de objetos y adicionalmente sabiendo 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 del modelo ya que es bastante difícil encontrar los valores óptimos que se ajusten.

En conclusión, se pudo cumplir con los objetivos inicialmente propuestos que esperaban tasas de acierto superiores al 80% y obtener muy buenos resultados en la detección de objetos reciclables. Entre las múltiples pruebas realizadas se tuvieron muy pocos falsos positivos, lo que permite pensar en tareas futuras que permitan ampliar el dataset y perfeccionar el algoritmo durante el entrenamiento. También se logró implementar la detección en una minicomputadora, se busca como tarea futura implementar junto con el algoritmo el manejo de un brazo robótico pudiendo no solo detectar los objetos sino también separarlos.

7- 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] 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

[4] Kumar, N. S., Vuayalakshmi, B., Prarthana, R. J., Shankar, A. (2016). IOT based smart garbage alert system using Arduino UNO. In: Region 10 Conference (TENCON), 2016 IEEE, pp. 1028-1034. doi: 10.1109/TENCON.2016.7848162

[5] Transfer Learning Wikipedia (2020), Transfer Learning, https://en.wikipedia.org/wiki/Transfer_learning, recuperado 01 julio de 2020.

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

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

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

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

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

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

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

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

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

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

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

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

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

[19] Aprende Machine Learning (2020), https://www.aprendemachinelearning.com/, recuperado 25 de abril 2020.

[20] Bismart, 2022, "¿Cuál es la diferencia entre el machine learning y el deep learning?", https://blog.bismart.com/diferencia-machine-learning-deep-learning, España.

[21] 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.

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