UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA ELECTRÓNICA

MONITOREO DE RESIDUOS SÓLIDOS EN LAS RIBERAS DEL RÍO "SAN PEDRO" MEDIANTE EL RECONOCIMIENTO DE OBJETOS CON UN DRON.

Trabajo de titulación previo a la obtención del
Título de Ingeniero Electrónico

AUTOR: Daniel Francisco Ortega Chulde

TUTOR: Carmen Johanna Celi Sánchez

Quito-Ecuador

2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Daniel Francisco Ortega Chulde con documento de identificación N° 1003850086 manifiesto que:

Soy autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 19 de febrero del año 2024

Atentamente,

Daniel Francisco Ortega Chulde
1003850086

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Daniel Francisco Ortega Chulde con documento de identificación Nº 1003850086; expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "Monitoreo de residuos sólidos en las riberas del río "San Pedro", mediante el reconocimiento de objetos con un dron", el cual, ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana

Quito, 19 de febrero del año 2024

Atentamente,

Daniel Francisco Ortega Chulde
1003850086

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Carmen Johanna Celi Sánchez con documento de identificación N°1717437808, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: MONITOREO DE RESIDUOS SÓLIDOS EN LAS RIBERAS DEL RÍO “SAN PEDRO” MEDIANTE EL RECONOCIMIENTO DE OBJETOS CON UN DRON, realizado por Daniel Francisco Ortega Chulde con documento de identificación N° 1003850086, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 19 de febrero del año 2024

Atentamente,

_____

Ing. Carmen Johanna Celi Sánchez Mgtr.
1717437808

DEDICATORIA Y AGRADECIMIENTO

A mi familia que desde mi infancia supo darme valores y consejos, para ser una buena persona, y siempre me apoyaron no solo en lo académico, sino también en todo momento, motivándome a culminar mis objetivos.

A mis maestros que me brindaron sus conocimientos para culminar mi camino universitario, por su gran apoyo ofrecido en este trabajo, por su tiempo compartido y por impulsar el desarrollo de mi formación profesional.

A mis amigos con los que nos hemos apoyado mutuamente en nuestra formación y que, hasta ahora, su amistad es señal de constancia y fraternidad.

INDICE DE CONTENIDOS

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE.....II
TITULACIÓN.....II
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA..... III
CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN IV
DEDICATORIA Y AGRADECIMIENTO.....V
ÍNDICE DE FIGURAS VIII
ÍNDICE DE TABLAS..... IX
RESUMEN X
ABSTRACT XI
CAPÍTULO 1 1
ANTECEDENTES 1
1.1 Planteamiento del problema..... 1
1.2 Justificación del proyecto 2
1.3 Objetivos..... 3
1.3.1Objetivo General 3
1.3.2 Objetivos Específicos 3
1.4 Delimitación..... 4
1.5 Glosario..... 5
CAPÍTULO 2 6
FUNDAMENTACIÓN TEÓRICA 6
2.1 Introducción..... 6
2.2 Inteligencia artificial..... 6
2.3 Visión artificial 6
2.4 Red neuronal..... 7
2.5 Neuronas 7
2.6 Open CV 7
2.7 Tensor Flow 7
2.8 Google Collab..... 8
2.9 Deep Learning..... 8
2.10 Detección de objetos..... 9
2.11 YOLO..... 9
2.12 Labellmg..... 9

2.13 Python 10
2.14 Dron 10
CAPÍTULO 3 11
3.1 Construcción de la base de datos y aprendizaje 11
3.2 Recolección de fotos usando un dron para crear la base de datos 11
3.3 Etiquetado de imágenes 13
3.4 Aumento de imágenes 19
3.5 Darknet 21
3.6 Entrenamiento usando Google Collab 24
CAPÍTULO 4 26
4.1 Prueba de software 26
CONCLUSIONES 35
RECOMENDACIONES 37
REFERENCIAS BIBLIOGRÁFICAS 38
ANEXOS 40

ÍNDICE DE FIGURAS

Figura 1 Ubicación del proyecto. 4
Figura 2 Dron DJI AIR 2S..... 11
Figura 3 Posición del dron para la recolección de archivos multimedia. 13
Figura 4 Archivos multimedia obtenidos de la memoria del dron. 13
Figura 5 Programa Labellmg. 14
Figura 6 Basura recurrente encontrada en el río, en donde resalta contenerdores de comida, envases tetrapak y botellas. 14
Figura 7 Contenedores de comida. 15
Figura 8 Envases Tetrapak. 15
Figura 9 Botellas PET. 16
Figura 10 Opciones en Labellmg. 16
Figura 11 Ventana para elegir la clase a la que pertenece. 17
Figura 12 Etiquetado de imágenes de varios artículos. 18
Figura 13 Bloc de notas obtenido por cada imagen. 18
Figura 14 Archivo generado por Labellmg. 19
Figura 15 Imagen original para aplicar aumento de imágenes. 20
Figura 16 Deformación elástica. 20
Figura 17 Se le aplica adición de ruido y desplazamiento. 21
Figura 18 Archivos para empezar el entrenamiento. 22
Figura 19 Entorno de Google Colab. 23
Figura 20 Archivos pertenecientes a Darknet. 23
Figura 21 Google Colab entrenando con Darknet. 25
Figura 22 Archivo (weights) obtenido. 26
Figura 23 Línea de código. 27
Figura 24 Archivo mp4. 27
Figura 25 Porcentaje de predicción obtenido de la consola. 28
Figura 26 Fotograma obtenido del video analizado. 28
Figura 27 Imagen del primer entrenamiento con un fondo homogéneo. 29
Figura 28 Imagen para el nuevo entrenamiento con cambio de escena y aumento de objetos. 29
Figura 29 Imágenes preparadas para el entrenamiento. 30
Figura 30 Fotograma obtenido del video analizado. 30
Figura 31 Porcentaje de detección en contenedores de comida es superior 94%. 31
Figura 32 Porcentaje de detección en botellas 94% y Tetrapak 80 %. 32

ÍNDICE DE TABLAS

Tabla 1 Capacidad de vuelo de la aeronave 12
Tabla 2 Gráfica de porcentaje obtenido durante los entrenamientos 33
Tabla 3 Gráfica comparativa entre las distancias en metros a la que es reconocido un objeto34

RESUMEN

El presente proyecto ha sido diseñado para realizar el reconocimiento de residuos sólidos en las riberas del río “San Pedro”. Muchos desechos a lo largo del tiempo se han ido acumulando en las orillas de dicho río. Explorando posibles aplicaciones de las nuevas tecnologías para abordar el problema, se considera la implementación de inteligencia artificial. Esta tecnología se encargaría de identificar tres categorías de residuos específicas: contenedores de comida (como tarrinas, platos y vasos desechables), envases Tetrapak y botellas PET.

Con el fin de cumplir este propósito, se empleó un dron para acceder aéreamente a distintas zonas del río. Desde esa perspectiva, se observaron y capturaron archivos multimedia de los residuos en las orillas, los cuales fueron posteriormente procesados mediante software de reconocimiento.

Al terminar el procesamiento de los archivos multimedia con el software de inteligencia artificial, se obtuvo un promedio de 93 % de efectividad en el reconocimiento de las tres clases de residuos.

Palabras claves: YOLO, neurona, aprendizaje, imagen.

ABSTRACT

The present project has been designed to recognize solid waste along the banks of the "San Pedro" river. Over time, a considerable amount of waste has accumulated along the riverbanks. Exploring potential applications of new technologies to address the issue, the implementation of artificial intelligence is being considered. This technology would be responsible for identifying three specific categories of waste: food containers (such as containers, disposable plates, and cups), Tetrapak packaging, and PET bottles.

In order to achieve this purpose, a drone was employed to access various areas of the river from the air. From this perspective, multimedia files of the waste along the banks were observed and captured, which were subsequently processed using recognition software.

Upon completion of the multimedia file processing with the artificial intelligence software, an average effectiveness of 93% was achieved in recognizing the three classes of waste.

CAPÍTULO 1

ANTECEDENTES

1.1 Planteamiento del problema

La contaminación ambiental por residuos sólidos es uno de los principales problemas que afectan a diferentes ecosistemas acuáticos. Es así Soliz et al, (2020) explica que en el Ecuador uno de los sitios recurrente en las grandes ciudades y centros poblados del país, para aliviar la carga de residuos son las cuencas hidrográficas, realidad que afecta el desarrollo de cualquier actividad humana.

Según Sánchez y Tello (2019) la contaminación de ríos y otros causes naturales se asume desde varios motivos, por ejemplo, el crecimiento demográfico, la falta de un adecuado manejo de residuos sólidos, la explotación minera y petrolera han influenciado notablemente en el grado o nivel de afectación de diferentes fuentes dentro del territorio ecuatoriano.

Un ejemplo de un problema con residuos sólidos está en el río “San Pedro”, que atraviesa los valles orientales de Quito, pues sufre de un alto grado de contaminación; de acuerdo a Pilalumbo (2020) esta cuenca hídrica que cuenta con una extensión de 8, 24 Km^2 ha sido contaminada debido a aguas servidas de la población en general, los efluentes de las industrias, actividad minera, sector hidrocarburíferas, los plaguicidas utilizados en la agricultura, entre los principales desencadenantes. Asimismo, concluyó sobre la necesidad de mantener en constante tratamiento la mencionada fuente hídrica, pues es fuente de consumo humano, animal, además de ser utilizada en ganadería.

Con base en la información anterior se desarrolló el presente proyecto, que, mediante la captura y procesamiento de imágenes a través de recursos tecnológicos de visión artificial, utilizando un dron, permitió identificar tipos de residuos sólidos sobre la ribera del río San Pedro, tales como envases tetra pack, botellas, fundas de basura, envolturas de productos y contenedores. Lo cual a posterior permitirá desarrollar iniciativas con respecto a la limpieza y mantenimiento del cauce del mencionado río.

1.2 Justificación del proyecto

Entre las innumerables ventajas de los avances tecnológicos en la actualidad, se reconoce la capacidad de procesar información gráfica, mediante recursos de visión artificial; por ejemplo, la capacidad de detectar, cartografiar y rastrear desechos sólidos dentro del entorno ambiental. Esta aplicación se vuelve evidente al referir la problemática de la contaminación ambiental que afectan todos los ecosistemas de la geósfera, especialmente al referir la hidrosfera, donde debido a las condiciones del líquido vital, este se vuelve susceptible de ser contaminado, afectando directamente su consumo y restringiendo su aplicación en distintas esferas de la actividad humana.

La envergadura del problema, mencionado anteriormente, es tan vasta que determinar por dónde comenzar a abordarlo resulta desafiante. Una solución potencial radica en la implementación de drones equipados con cámaras de alta definición. Estos vehículos aéreos no tripulados podrían ser utilizados para mapear con precisión la ubicación de los agentes contaminantes. Posteriormente, las imágenes obtenidas, se someten a un proceso previo en software computacional, que emplearía redes neuronales, para llevar a cabo el reconocimiento de los elementos visuales y así identificar los residuos sólidos o la basura, junto con su cantidad. Los datos resultantes de este proceso se aplicarían en el planteamiento de acciones o posibles soluciones, pertinentes a las autoridades responsables o empresas de limpieza interesadas en abordar dicha problemática.

1.3 Objetivos

1.3.1Objetivo General

- Implementar un sistema de IA para el reconocimiento de los residuos sólidos en las orillas del río "San Pedro", utilizando un dron y su respectiva cámara.

1.3.2 Objetivos Específicos

- Investigar la tecnología necesaria y los modelos de IA disponibles que efectúen el reconocimiento de objetos, tales como envases tetra pack, botellas PET, fundas de basura, envolturas de productos y contenedores de comida.

- Construir una base de datos que incluya las imágenes necesarias para el entrenamiento de la IA y efectuar el etiquetado de los ítems correspondientes.

- Entrenar un modelo de red neuronal para el reconocimiento de los residuos sólidos en la ribera del río “San Pedro”.

- Realizar las pruebas de funcionamiento con las imágenes obtenidas por el dron para posteriormente evaluar los resultados de la inteligencia artificial entrenada.

1.4 Delimitación

El sistema está enfocado para realizar reconocimiento sobre las orillas del río en donde existe arena, roca y vegetación.

No se recomienda realizarlo en lugares en donde la luz del sol tiene alta intensidad debido a que refleja directamente en la cámara y los archivos multimedia pueden salir quemados, de igual manera cuando la luz es mínima, se recomienda realizar la captura de datos hasta las 18:00 horas.

Figura 1 Ubicación del proyecto.

Fuente: Captura de imagen de Google maps (-0.2065499, -78.4187666)

1.5 Glosario

Neurona artificial. Las redes neuronales artificiales están basadas en el funcionamiento de las redes de neuronas biológicas. Las neuronas que todos tenemos en nuestro cerebro están compuestas de dendritas, el soma y el axón: Las dendritas se encargan de captar los impulsos nerviosos que emiten otras neuronas. Estos impulsos, se procesan en el soma y se transmiten a través del axón que emite un impulso nervioso hacia las neuronas contiguas (García Colmenero, 2018).

Inteligencia artificial (IA). Se refiere a sistemas o máquinas que imitan la inteligencia humana para realizar tareas y pueden mejorar iterativamente a partir de la información que recopilan. La IA se manifiesta de varias formas (Romero, 2007).

Visión artificial. Es parte de la IA que se centra en el desarrollo y perfeccionamiento de técnicas que permiten ver, identificar y procesar imágenes de la misma manera que lo hace la visión humana (Bodla N, 2017).

CAPÍTULO 2

FUNDAMENTACIÓN TEÓRICA

2.1 Introducción

En este capítulo se describe los temas que fueron abarcados en el proyecto para comprender el funcionamiento de la inteligencia artificial y los campos en los cuales puede dar soluciones el uso de la misma. También se describe los elementos que fueron utilizados en el desarrollo del software para proyecto.

2.2 Inteligencia artificial

La inteligencia artificial es la capacidad que un computador puede tener para aprender y realizar acciones a través del conocimiento adquirido a través de información, las cuales requiere de un análisis como la realiza la inteligencia humana (Endara y Sumba, 2021, p. 8).

El término ha sido aplicado frecuentemente en la elaboración de proyectos provistos con características propias de seres humanos con la facultad de analizar, explorar, razonar o aprender de acciones o experiencias a partir de datos erróneos o acertados, pero a diferencia de la inteligencia humana, una inteligencia artificial puede manejar grandes cantidades de datos a la vez (Barahona, 2019, p. 12).

La inteligencia artificial tiene un crecimiento vertiginoso en la actualidad, en donde se desarrolla aplicaciones en veneficio de los seres humanos a realizar acciones complejas, peligrosas o que pueden ser repetitivas, pudiendo resolver problemas en casi cualquier campo y resultando más eficientes (Rouhiainen, 2018, p.15).

2.3 Visión artificial

Es parte de la IA que se centra en el desarrollo y perfeccionamiento de técnicas que permiten ver, identificar y procesar imágenes de la misma manera que lo hace la visión humana (Bodla, 2017, p.5).

2.4 Red neuronal

Las redes neuronales artificiales están basadas en el funcionamiento de las redes de neuronas biológicas. Las neuronas que todos tenemos en nuestro cerebro están compuestas de dendritas, el soma y el axón: Las dendritas se encargan de captar los impulsos nerviosos que emiten otras neuronas. Estos impulsos, se procesan en el soma y se transmiten a través del axón que emite un impulso nervioso hacia las neuronas contiguas (Garcíaet al., 2018, p.25).

2.5 Neuronas

Son los elementos más pequeños de una red neuronal, los cuales son los procesadores de la información con limitada capacidad de procesamiento, son los encargados de procesar la información de las diferentes señales de entradas de una red neuronal y convertir en una única salida consolidada resultante, así mismo este resultado puede ser señal de entrada para otras neuronas. Cada neurona tiene un valor de importancia para su conexión y activación dentro de una red la cual se los denomina pesos (Díaz et al., 2023, p. 15).

2.6 Open CV

Open CV es una biblioteca de código abierto desarrollada por Intel, que dispone de una cantidad importante de librerías para el desarrollo de programación orientada a la visión artificial, sus algoritmos son usados principalmente para el procesamiento de imágenes que ayudan en la clasificación, identificación de objetos, caras, formas, patrones etc. Es un software multiplataforma que tiene soporte en C++, Python, Java y Matlab (Segura, 2019, p.28).

2.7 Tensor Flow

Es una plataforma de código abierto integral para el aprendizaje automático que se enfoca en la simplicidad y la facilidad de uso, con actualizaciones como ejecución entusiasta, API intuitivas de alto nivel y creación de modelos flexibles en cualquier plataforma (Falconí, 2021, p. 15).

Esta plataforma integral permite el aprendizaje automático. Es compatible con:

- Cálculo numérico basado en matrices multidimensionales
- GPU y procesamiento distribuido
- Diferenciación automática
- Construcción, capacitación y exportación de modelos.

2.8 Google Collab

También conocido como “Colaboratory”, es un entorno interactivo que permite escribir, ejecutar y compartir código dentro de Google Drive, en esta plataforma TensorFlow ya está preinstalado cuando se crea un nuevo cuaderno en colab.research.google.com. Simplemente se utiliza la librería “import tensorflow as tf ”, y comience a codificar (Cervera et al., 2022, p. 36).

Igualmente se debe destacar que los cuadernos de Google Colab son como documentos y hojas de cálculo de Google, pues se almacenan en Google Drive, se pueden compartir, editar y comentar de forma colaborativa, para lo cual se utiliza la opción “Compartir” dispuesta en la parte superior derecha del interfaz del cuaderno generado. De manera predeterminada, las notebooks de Colab se ejecutan en la CPU (Bodero et al., 2020, p.52).

2.9 Deep Learning

Según Pérez (2021) Deep Learning forma parte del campo del aprendizaje automático que engloba la inteligencia artificial, lo cual permite entrenar una máquina para realizar tareas como lo harían seres humanos como son reconocimiento del habla, reconocimiento de imágenes, predicciones.

Las técnicas de Deep Learning permiten sustraer características más importantes automáticamente de datos mediante el uso de redes neuronales organizadas en capaz. Las características son extraídas en un orden jerárquico, siendo adquiridas de menor complejidad a mayor grado de complejidad, estas últimas características complejas serán usadas para generar la salida al modelo con lo cual se busca dar solución a un problema (Pérez, 2021, p.42).

2.10 Detección de objetos

La detección de objetos es una tecnología de aprendizaje profundo (Deep Learning), que permite detectar objetos en imágenes y videos. Según Bodla (2017) basa en la identificación de un objeto con un cuadro delimitador en la imagen; lo cual permite categorizar (clasificar) y verificar si un objeto se visualiza en una determinada imagen en términos de probabilidad. La diferencia entre la localización de objetos y la detección de objetos es sutil. Simplemente, la localización de objetos tiene como objetivo ubicar el objeto principal (o más visible) en una imagen, mientras que la detección de objetos intenta encontrar todos los objetos y sus límites (Bodla, 2017, p.9).

Igualmente se debe considerar que, mediante un algoritmo de localización de objetos, se generan las coordenadas de ubicación con respecto a la imagen. Para visualizar dicho resultado, la forma más práctica es utilizar cuadros delimitadores que permiten establecer las coordenadas de ubicación del objeto (Bodla, 2017, p.10).

2.11 YOLO

YOLO, "You Only Look Once" es un algoritmo propuesto por Redmond en un artículo de investigación publicado en la IEEE/CVF "Conference on Computer Vision and Pattern Recognition (CVPR)", en comparación con el enfoque adoptado por los algoritmos de detección de objetos antes de YOLO, que reutilizan los clasificadores para realizar la detección; dicha herramienta propone el uso de una red neuronal de extremo a extremo, que hace predicciones de cuadros delimitadores y probabilidades de clase al mismo tiempo. Siguiendo un enfoque fundamentalmente diferente para la detección de objetos.

Con lo cual logra resultados de última generación superando a otros algoritmos de detección de objetos en tiempo real por un amplio margen, corriendo hasta 45 FPS (fotogramas por segundo) (Buitrón, 2023, p. 15).

2.12 Labellmg

LabelImg es una herramienta de anotación de imágenes gráficas. Está elaborado en lenguaje Python y usa Qt para su interfaz gráfica. Las anotaciones se guardan como archivos

XML en formato PASCAL VOC, el formato utilizado por ImageNet. Además, también es compatible con los formatos YOLO y CreateML (Layme et al., 2023, p.23).

2.13 Python

Es un lenguaje de programación de alto nivel de código abierto, el cual es muy popular en el desarrollo científico. Se destaca por su sencillez y fácil codificación evitando errores frecuentes que se da en otros lenguajes. Permite desarrollar algoritmos sencillos como también muy complejos con un menor número de líneas de código, siendo también este uno de entre los más usados para el desarrollo de inteligencia artificial (Damiani, 2019, p.18).

2.14 Dron

Los drones, a veces denominados "vehículos aéreos no tripulados" (UAV), están destinados a realizar diferente tipo de tareas teledirigidas. Su mecanización y aplicabilidad abarca un sinnúmero de tareas definidas por el usuario, considerando características específicas en su proceso de construcción y programación. Aunque inicialmente fueron concebidos para uso en la industria militar y aeroespacial, los drones han encontrado su camino en la corriente principal gracias a los niveles mejorados de seguridad y eficiencia que ofrecen. Estos vehículos aéreos no tripulados operan sin requerir la presencia de un piloto a bordo, y pueden variar en términos de su grado de autonomía (Gómez et al.,2021, p. 23).

Entre las características específicas de dichos dispositivos se considera la capacidad de viajar a diferentes alturas y distancias, que generalmente van desde los 4 km hasta los de 50 km, dependiendo de la especificidad de la tarea y condiciones de fabricación y programación del dispositivo (Del Río et al.,2019, p. 22).

CAPÍTULO 3

3.1 Construcción de la base de datos y aprendizaje

En esta sección se detalla el diseño del software a considerar en el proceso de detección de tres clases de residuos comunes en la cuenca del río San Pedro:

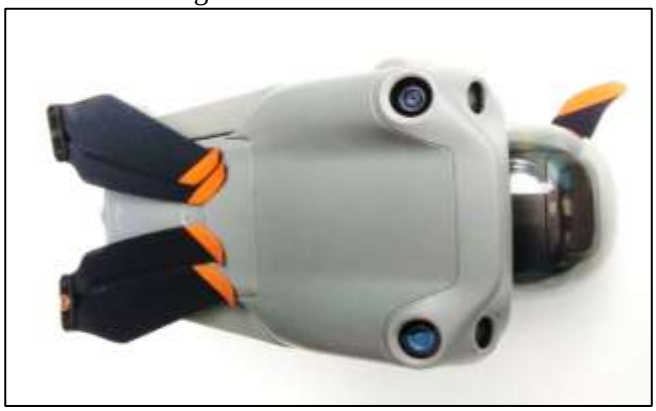
- Contenedores de comida (Tarrinas de plástico, tarrinas de poliestireno, platos desechables)
- Botellas PET
- Envases Tetrapack.

3.2 Recolección de fotos usando un dron para crear la base de datos

Para el desarrollo de este proyecto técnico, se optó por utilizar como medio de captura digital, una cámara incorporada al dron, como fuente primaria de obtención de archivos multimedia. Posteriormente, los archivos obtenidos se procesaron, utilizando un equipo informático con especificaciones necesarias, para desarrollar el reconocimiento y categorización.

El dron seleccionado para el desarrollo del proceso investigativo, fue un dispositivo marca DJI, mismo que presentó especificaciones indispensables como contar con sensores de aproximación integrados, sensores infrarrojos, además se consideró el tamaño del sensor integrado a la cámara. A continuación, se describe el dispositivo seleccionado, para el desarrollo del presente proyecto.

Figura 2 Dron DJI AIR 2S.



Elaborado por: Daniel Ortega

Tabla 1

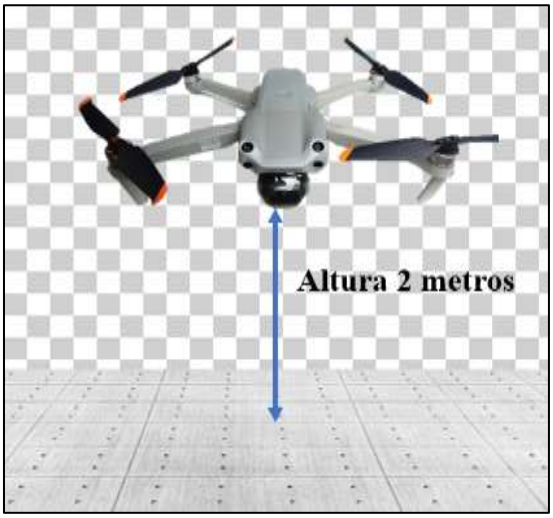
Capacidad de vuelo de la Aeronave

Techo máximo de servicio sobre el nivel del mar	5000m
Tiempo máximo de vuelo (sin viento)	31 minutos
Tiempo máximo de vuelo estacionario (sin viento)	30 minutos El tiempo de vuelo se midió en un entorno de prueba controlado. Las condiciones de prueba específicas son las siguientes: sin viento, a nivel del mar, velocidad de vuelo constante de 32,4 kph, APAS apagado, Aírense apagado, parámetros de la cámara configurados en 1080p/24 fps, modo de video apagado y vuelo con batería del 100 % a batería del 0 %. Los resultados pueden variar según el entorno, el escenario de uso y la versión de firmware.
Distancia máxima de vuelo (sin viento)	10,5 kilómetros
Velocidad máxima de vuelo (cerca del nivel del mar, sin viento)	19 m/s (modo S) 15 m/s (modo N) 5 m/s (modo C)
Resistencia máxima a la velocidad del viento	10,7 m/s

Fuente: (Manual de usuario del dron, 2021)

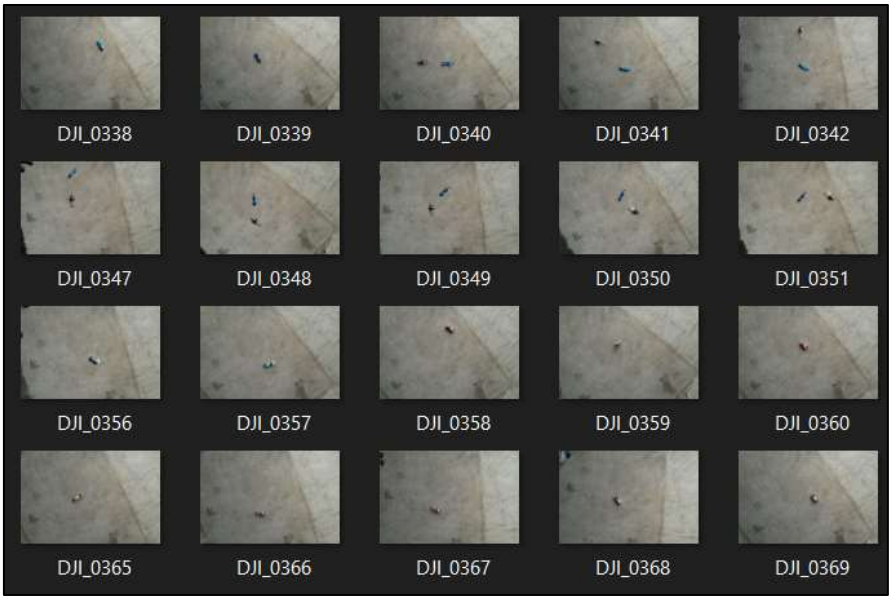
Una vez montado y preparado el dispositivo, se inició el proceso de obtención de archivos multimedia, para lo cual considerando las especificaciones técnicas se procedió a recolectar un promedio de 500 imágenes de las tres clases de objetos, previamente descritos y referenciando una altura promedio de dos metros.

Figura 3 Posición del dron para la recolección de archivos multimedia.



Elaborado por: Daniel Ortega

Figura 4 Archivos multimedia obtenidos de la memoria del dron.

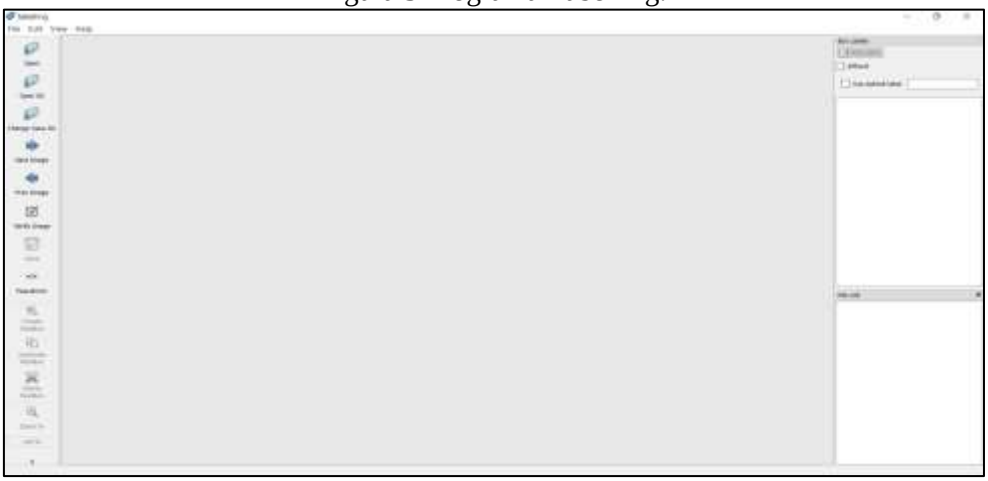


Elaborado por: Daniel Ortega

3.3 Etiquetado de imágenes

Para realizar el etiquetado utilizamos la aplicación de Labellmg, lo cual permitió registrar la información, para posteriormente clasificarla de acuerdo al tipo de archivo generado.

Figura 5 Programa Labellmg.



Elaborado por: Daniel Ortega

Con las imágenes obtenidas por el dron, se clasificó las imágenes en 3 carpetas de acuerdo a especificaciones previamente descritas:

- Contenedores
- Tetrapak
- Botellas

Figura 6 Basura recurrente encontrada en el río, en donde resalta contenerdores de comida, envases tetrapak y botellas.

Elaborado por: Daniel Ortega

Primera clase: Contenedores de comida

En esta carpeta se almacenó imágenes pertenecientes a contenedores, tales como envases de goma y espuma, tarrinas de plástico en varios tamaños, además de platos desechables.

Figura 7 Contenedores de comida.

Elaborado por: Daniel Ortega

Segunda clase: Envases Tetrapak

En la segunda carpeta, se registró imágenes de envases de leche, de jugo, de yogurt y de avena.

Figura 8 Envases Tetrapak.

Elaborado por: Daniel Ortega

Tercera clase: Botellas PET

En la tercera y última carpeta de recipientes se usó botellas de polietileno tereftalato de distintas marcas, por ejemplo, de coca cola, Güitig, Gatorade, V220, etc.

Figura 9 Botellas PET.

Elaborado por: Daniel Ortega

En cada carpeta se almacenó un promedio de 250 fotos obtenidas del dron, seguidamente se procedió a abrir mediante Labellmg. El mencionado programa se caracteriza por una interfaz intuitiva, lo cual permitió navegar y gestionar los archivos creados de forma eficiente. Es así que, para iniciar con el etiquetado, operación que consiste en marcar manualmente cada imagen con su respectiva clase, se predeterminó que para la mencionada operación se realice a través del programa YOLO como se muestra en la figura 10.

Figura 10 Opciones en Labellmg.

Elaborado por: Daniel Ortega

Posteriormente se procedió a etiquetar cada imagen con su respectiva clase, para desarrollar este paso previamente se asignó a cada clase un número, por lo tanto, para contenedores de comida se asignó el número 0, para envases Tetrapak se asignó el número 1 y para botellas PET, el número 2 de esta manera se constituyó una base de datos de las tres clases. Al crear el cuadro para etiquetar cada artículo, se generó un cuadro para registrar la clase a la que pertenece el objeto captado, lo cual se evidencia en la figura 11.

Figura 11 Ventana para elegir la clase a la que pertenece.

Elaborado por: Daniel Ortega

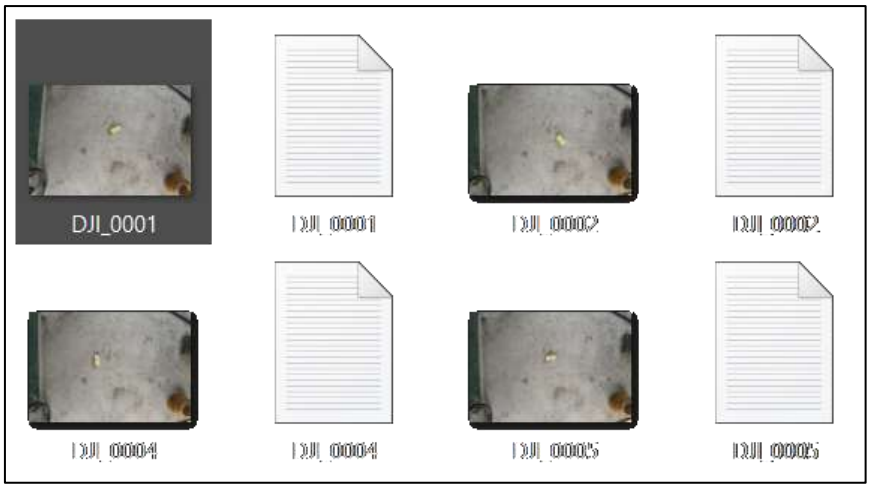
Además, cabe resaltar que, dentro del proceso de etiquetado, una imagen puede registrar más de un tipo de registro.

Figura 12 Etiquetado de imágenes de varios artículos.

Elaborado por: Daniel Ortega

Una vez que se etiquetaron todas las imágenes en las tres carpetas correspondientes, el programa genera un bloc de notas para cada imagen como se muestra en la figura 13.

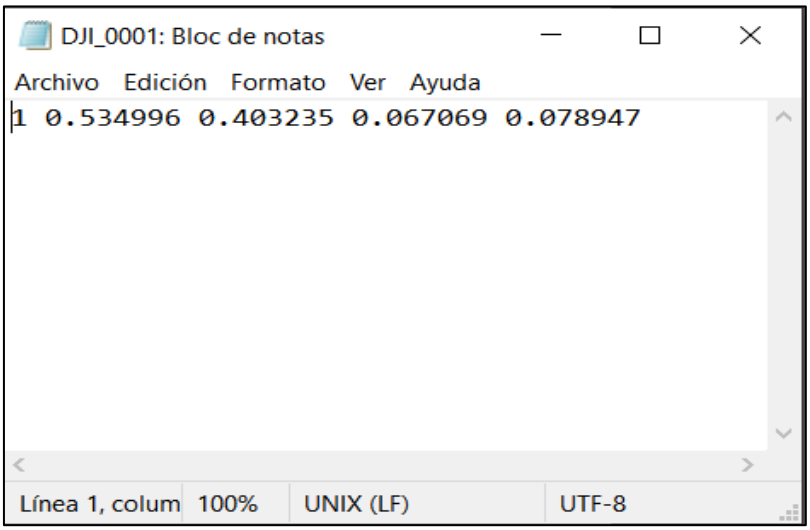
Figura 13 Bloc de notas obtenido por cada imagen.



Elaborado por: Daniel Ortega

El programa de Labellmg generó un bloc de notas en el cual se se registró información numérica de cada artículo marcado en la imagen, además de referenciar coordenadas que delimitan la ubicación del desecho fotografiado, clasificado y registrado previamente.

Figura 14 Archivo generado por Labellmg.



Elaborado por: Daniel Ortega

3.4 Aumento de imágenes

El propósito del aumento radica en generar más archivos con el fin de mejorar el proceso de entrenamiento. Esta estrategia de ampliación, se emplea de manera extensa en el entrenamiento de modelos de inteligencia artificial, particularmente en la tarea de reconocimiento de objetos. La ampliación de imágenes se refiere al uso de transformaciones y ajustes en las imágenes existentes, con el propósito de generar variaciones realistas. Esta práctica enriquece el conjunto de datos y fortalece la capacidad del modelo, para generalizar y reconocer objetos en diversas situaciones.

Algunas técnicas comunes de ampliación de imágenes incluyen:

- Rotación: Girar la imagen en un ángulo específico, para simular diferentes perspectivas.
- Escalamiento y recorte: Ajustar el tamaño y recortar regiones de interés en la imagen.
- Desplazamiento y traslación: Mover la imagen dentro de su espacio para cambiar su posición relativa.
- Cambio de iluminación: Modificar el brillo, el contraste o la saturación de la imagen para simular condiciones de iluminación variables.
- Flipping o espejado: Voltear horizontal o verticalmente la imagen.

- Ruido: Agregar ruido aleatorio a la imagen para simular condiciones más realistas.
- Deformación elástica: Aplicar deformaciones localizadas para simular deformaciones en la imagen.

Para generar el mencionado proceso de entrenamiento, se aplicó el desplazamiento, deformación elástica y adición de ruido. Tal como se evidencia en las figuras 15, 16 y 17.

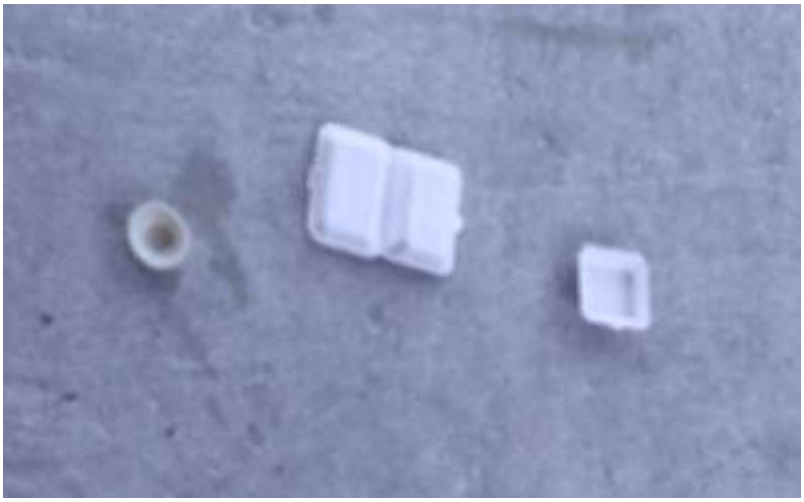
Figura 15 Imagen original para aplicar aumento de imágenes.

Elaborado por: Daniel Ortega

Figura 16 Deformación elástica.

Elaborado por: Daniel Ortega

Figura 17 Se le aplica adición de ruido y desplazamiento.



Elaborado por: Daniel Ortega

3.5 Darknet

Es un software el cual se utiliza para crear y entrenar sistemas de inteligencia artificial que pueden ver y reconocer objetos en imágenes y videos. Es especialmente conocido por su enfoque rápido y eficiente en la detección de objetos en tiempo real. El propósito de utilizar este software es direccionar al hardware, para el reconocimiento de diferentes objetos, como contenedores de comida, Tetrapak y botellas, en imágenes.

Mientras que Darknet permite crear una red neuronal, la cual gestiona imágenes generadas y etiquetadas previamente, permitiendo reconocer nuevas imágenes. Este software tiene la implementación de YOLO (You Only Look Once), un método que permite detectar y clasificar objetos, considerando registros previos por imagen. Darknet además se utiliza, para otras tareas de visión por computadora, como reconocimiento de rostros, textos en imágenes y segmentación de objetos en imágenes.

Para iniciar el proceso de entrenamiento, se procedió a descargar una versión actualizada de Darknet compatible con YOLO desde el repositorio oficial en GitHub, generando un archivo ZIP.

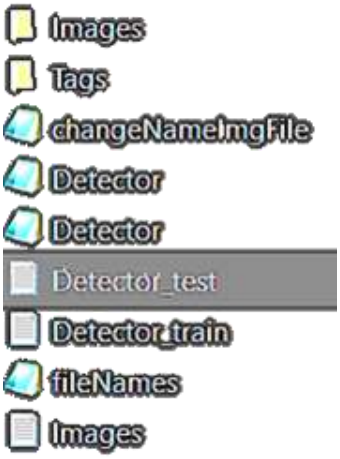
Pasos para utilizar el software:

- **Preparar los datos de entrenamiento**: Se recopiló las imágenes obtenidas con el dron, todas etiquetadas con un cuadro que contengan los objetos a reconocer mediante el modelo

preestablecido. Asimismo, se consideró que las imágenes contengan anotaciones sobre las coordenadas de los objetos visualizados en cada imagen.

- **Configurar el archivo de configuración**: Darknet requiere un archivo de configuración que define los parámetros de entrenamiento, como la arquitectura de la red neuronal, los directorios de datos y las clases de objetos que se están entrenando.

Figura 18 Archivos para empezar el entrenamiento.



Elaborado por: Daniel Ortega

- **Descargar los pesos pre entrenados**: Para optimizar el rendimiento y agilizar el proceso de entrenamiento, es factible obtener pesos preentrenados desde una extensa base de datos. Dichos pesos iniciales servirán como punto de partida para el entrenamiento con nuestros propios datos. Se empleó Google Drive para cargar los archivos respectivos en Darknet, junto con las imágenes etiquetadas, las cuales representaron un peso aproximado de 12 gigabytes.

- **Configurar Google Colab** Se procedió a abrir Google Colab y crear un nuevo cuaderno. Para lo cual se subió el código fuente de Darknet y los datos de entrenamiento a Google Colab o a su vez clonar el repositorio desde GitHub directamente en Colab.

- **Configurar el entorno de ejecución**: A continuación, se configuró Colab para utilizar una GPU. Dicho procedimiento se realizó mediante la opción "Entorno de ejecución" en la barra de menú de Colab.

Figura 19 Entorno de Google Colab.

Elaborado por: Daniel Ortega

- **Ejecutar los comandos de entrenamiento:** Utiliza los comandos de Darknet para iniciar el proceso de entrenamiento. Esto implica cargar los pesos preentrenados, configurar los directorios de datos y clases, para comenzar el entrenamiento con el conjunto de datos etiquetados preparados previamente.

Figura 20 Archivos pertenecientes a Darknet.

Elaborado por: Daniel Ortega

- **Monitorear el entrenamiento y ajustar los parámetros:** Durante el proceso de entrenamiento, se monitoreó las métricas de rendimiento y ajustó los parámetros, según fue necesario para obtener los mejores resultados. Es importante destacar que el entrenamiento de un modelo con Darknet en Google Colab llevó tiempo y requirió una cantidad significativa de recursos computacionales.

3.6 Entrenamiento usando Google Collab

Los modelos de aprendizaje profundo generalmente se basan en redes neuronales artificiales, que están compuestas por varias capas de neuronas interconectadas. Estas neuronas pueden ser del tipo perceptrón, unidad recurrente, unidad LSTM (memoria a largo plazo) u otras variantes, según la arquitectura y el tipo de modelo que se esté entrenando. Considerando lo anterior se desarrolló un proceso de preparación que constó de los siguientes pasos:

1. **Preparación del entorno:** En primer lugar, se configuró un entorno de Google Colab para utilizar una GPU, puesta esta acción aceleró significativamente el proceso de entrenamiento.

2. **Carga y preparación de datos**: Se cargó los datos de entrenamiento en el entorno de Google Colab. Esto tarea permitió subir imágenes, archivos de anotaciones y otros datos necesarios para el entrenamiento. Además, se realizó transformaciones adicionales sobre los datos antes de comenzar el entrenamiento.

3. **Configuración del modelo y entrenamiento**: Utilizando la biblioteca o el framework de aprendizaje profundo de su elección (como Tensor Flow, Pitarch o Darknet), se seleccionaron las funciones de pérdida adecuadas y se configuraron los hiperparámetros de entrenamiento. Luego, se inició el proceso de entrenamiento, utilizando los datos que se han preparado.

4. **Monitoreo y ajuste:** A medida que se desarrolló el modelo de entrenamiento, se monitoreó su rendimiento y ajustó los hiperparámetros. Además, se hizo un seguimiento de las métricas de rendimiento, como la precisión o la pérdida, y realizó iteraciones

adicionales de entrenamiento para mejorar los resultados. Una vez culminado el entrenamiento del programa se generó los pesos finales.

El formato en el que se guardó el modelo depende del framework utilizado. Por ejemplo, en TensorFlow, el modelo se guarda típicamente en un archivo con extensión .h5 (Hierarchical Data Format 5), mientras que en PyTorch, se guarda en un archivo con extensión. pth (PyTorch model file). Estos archivos contienen la información necesaria para recrear y utilizar el modelo entrenado en futuras aplicaciones o tareas de inferencia.

Guardar el modelo entrenado es beneficioso por varias razones. Permitió reutilizar el modelo en diferentes momentos sin tener que volver a entrenarlo desde cero, lo que ahorró tiempo y recursos computacionales. Además, el modelo pudo ser compartido con otros investigadores o desarrolladores, para que lo utilicen en sus propias aplicaciones.

Una vez guardado el modelo, este pudo ser cargado nuevamente en el entorno de desarrollo, para realizar inferencia en nuevos datos, es decir, hacer predicciones basadas en el conocimiento aprendido durante el entrenamiento.

Figura 21 Google Colab entrenando con Darknet.

Elaborado por: Daniel Ortega

CAPÍTULO 4

PRUEBAS Y RESULTADOS

4.1 Prueba de software

Al terminar el proceso de entrenamiento Google Colab, el cual duró un aproximado de 45 horas, se obtuvieron los pesos finales y sesgos, parámetros internos del modelo que determinan cómo se realizarán las predicciones y clasificaciones; considerando que el archivo generado contiene los valores finales que representa el conocimiento adquirido por el modelo durante el entrenamiento.

Figura 22 Archivo (weights) obtenido.

Elaborado por: Daniel Ortega

Para probar el funcionamiento se utilizó el software de anaconda, la cual es una plataforma de distribución de Python y R que facilitó la instalación, así como gestión de paquetes y entornos de desarrollo, pues proporcionó el gestor de paquetes "conda" además de una amplia selección de librerías y herramientas populares, para la programación científica.

Igualmente se trabajó en el entorno de Spyder, el cual está diseñado específicamente para el desarrollo y la programación en Python, y se enfoca en ser un entorno amigable para científicos de datos y analistas.

Posterior a lo cual se añadió los pesos al nuevo programa generado, en la línea de código agregamos el nombre del archivo “Detector_final. weights”.

Figura 23 Línea de código.

```
70  cv2.dnn.readNetFromDarknet('model\Detector.cfg','model\Detector_final.weights')
```

Elaborado por: Daniel Ortega

Y a continuación se procedió a ejecutar el código en el entorno de Spyder, generando una ventana que expone la opción de buscar una imagen o video.

Dependiendo del tamaño del archivo se definió el tiempo en analizarlo, cabe notar que para desarrollar las pruebas se utilizó video. Asimismo, destacar que el tamaño del archivo multimedia generado es de 56 MB en formato mp4.

Figura 24 Archivo mp4.

Elaborado por: Daniel Ortega

Una vez que el archivo ha sido cargado en nuestro software, se procede a hacer clic en el botón de "analizar". En la consola de Spyder, se puede visualizar el porcentaje con el cual el sistema está detectando los objetos.

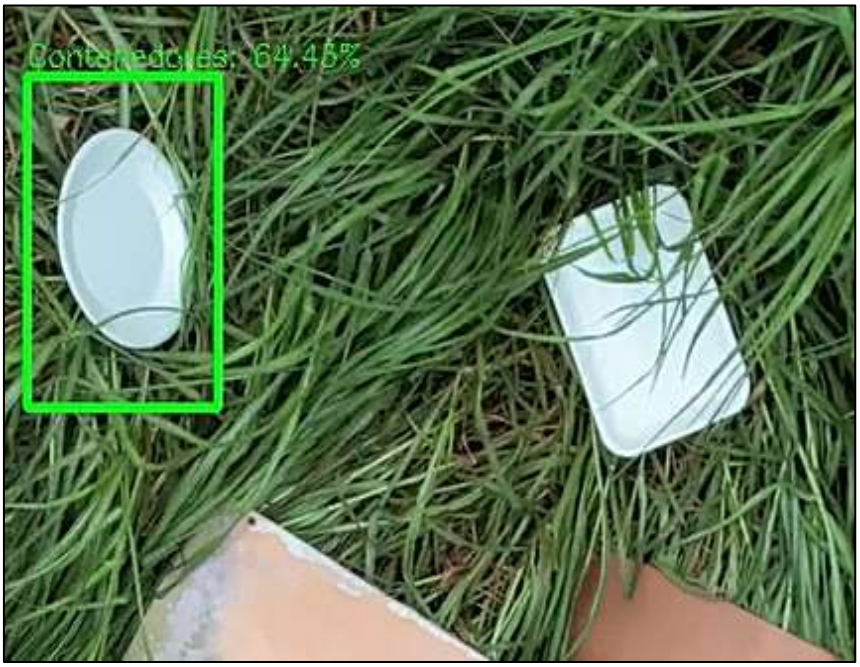
Figura 25 Porcentaje de predicción obtenido de la consola.

```
predicted object Botellas: 74.44%
predicted object Contenedores: 64.43%
predicted object Tetrapack: 58.75%
```

Elaborado por: Daniel Ortega

Al observar el video obtenido del análisis, se identificó que algunos objetos no fueron reconocidos, por lo que se precisó el modelo de predicción necesita ser mejorado.

Figura 26 Fotograma obtenido del video analizado.



Elaborado por: Daniel Ortega

Para mejorar el porcentaje de predicción, se procedió a mezclar las clases de elementos y aumentar su número en cada imagen, también se seleccionó un espacio con contraste de colores semejantes a los espacios naturales propios de cuencas hídricas, a diferencia del anterior compilado de imágenes capturadas con un fondo monocromático.

Figura 27 Imagen del primer entrenamiento con un fondo homogéneo.

Elaborado por: Daniel Ortega

Figura 28 Imagen para el nuevo entrenamiento con cambio de escena y aumento de objetos.

Elaborado por: Daniel Ortega

Para desarrollar el siguiente entrenamiento, se aumentó el número de imágenes a 1000, es decir el cuádruple del anterior, y se realizó el mismo proceso de etiquetado, para seguidamente cargarlo a Google Drive y entrenarlo con Google Colab.

Figura 29 Imágenes preparadas para el entrenamiento.

Elaborado por: Daniel Ortega

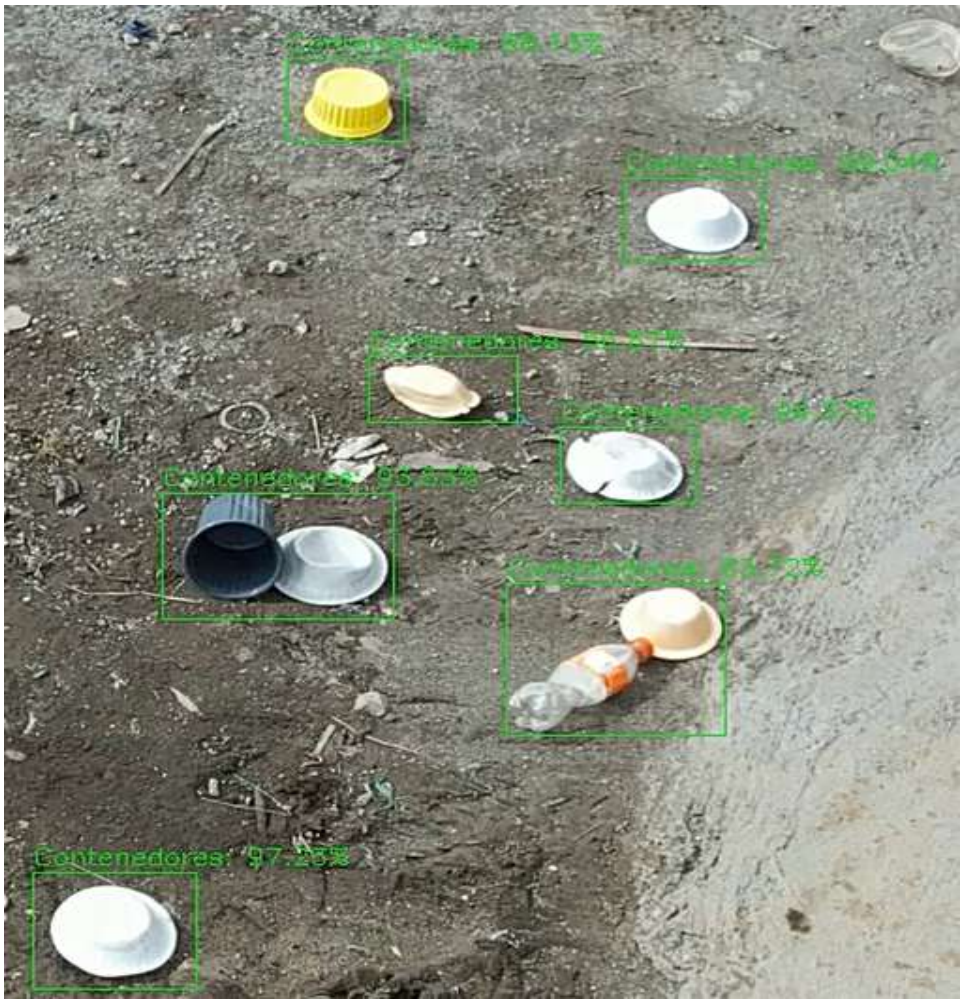
La suma de archivos para el entrenamiento resultó un aproximado de 18 gigas debido al aumento de imágenes. El tiempo que Google Collab tardó en entrenar a Darknet fue de 80 horas. Se obtuvieron los nuevos pesos, con el archivo generado, se realizó el mismo paso de cargarlo en el código y procedemos a ejecutarlo.

Figura 30 Fotograma obtenido del video analizado.

Elaborado por: Daniel Ortega

Como se identificó en el análisis del nuevo entrenamiento. el porcentaje de detección aumentó en un 98% para botellas y 80% para contenedores de comida.

Figura 31 Porcentaje de detección en contenedores de comida es superior 94%.



Elaborado por: Daniel Ortega

Figura 32 Porcentaje de detección en botellas 94% y Tetrapak 80 %.

Elaborado por: Daniel Ortega

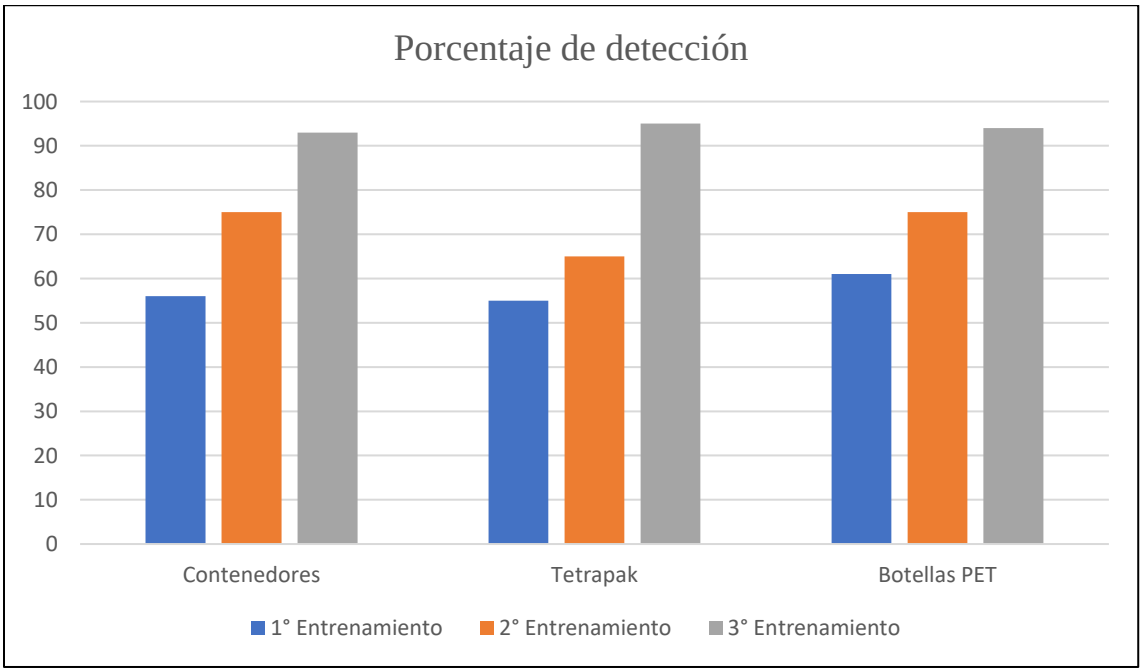
Algo muy importante a recalcar es que el porcentaje de acierto es proporcional a la distancia a la que se encuentre el objeto a reconocer. La toma de imágenes para el entrenamiento se llevó a cabo a 2 metros de altura. Debido a esta razón, si la cámara está lejos de los objetos, tiende a no reconocerlos; en cambio, si está cerca, el porcentaje aumenta considerablemente.

Para este proyecto, se realizaron tres entrenamientos. En el segundo entrenamiento, se utilizó el archivo (. weights) del primer intento, el cual contribuyó a optimizar el segundo. Además, se añadieron imágenes con aumento, lo que resultó en un incremento del porcentaje, promediando entre el 70% y el 80% en cada clase de imagen procesada.

Durante las pruebas, se observó que el reconocimiento solo se lograba si el lugar donde se encontraban los objetos era uniforme, por ejemplo, un patio de cemento. Debido a esta razón, se procedió a realizar un tercer entrenamiento, incorporando imágenes que mostraban distintos entornos, como tierra, césped, plantas y objetos que no pertenecen al entorno. Con lo cual el programa alcanzó un porcentaje superior al 90% en todas las clases, dependiendo de la distancia a la que se capturó el video.

Tabla 2

Gráfica de porcentaje obtenido durante los entrenamientos

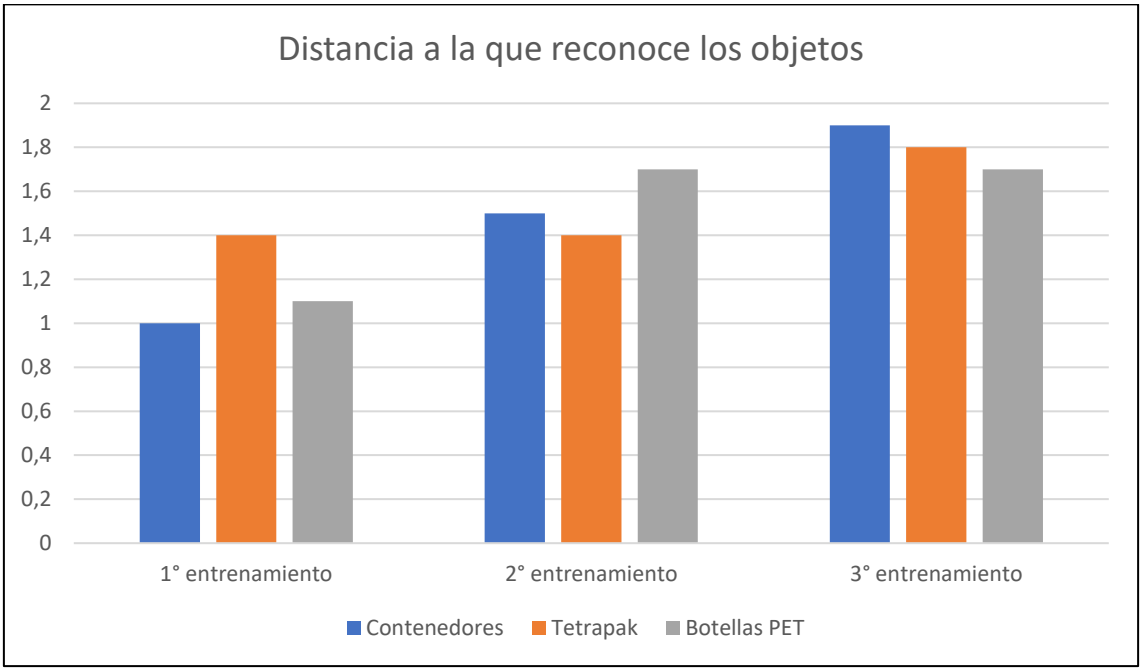


Elaborado por: Daniel Ortega

Igualmente, un elemento que influye significativamente es la iluminación con la que se obtienen las capturas. Por lo tanto, si a un objeto la luz que recibe (ya sea natural o artificial) es demasiado intensa, el programa tendrá inconvenientes para detectar esos objetos sobreexpuestos a la luz.

Tabla 3

Gráfica comparativa entre las distancias en metros a la que es reconocido un objeto



Elaborado por: Daniel Ortega

CONCLUSIONES

- La inteligencia artificial tiene la capacidad de reconocer objetos en un porcentaje superior al 90%, siempre y cuando estos no se encuentren a más de 2 metros de distancia.

- El dron permite el acceso a rincones del río que serían inaccesibles sin un barco o equipo para escalada y descenso.

- Las escenas en las que se realizó la toma de imágenes para el entrenamiento son muy importantes, pues en el momento en que estas sean ingresadas al programa, decidirán el porcentaje de efectividad en el análisis de datos.

- La visión artificial posibilitó una detección precisa y confiable de los residuos. Utilizando algoritmos y técnicas de procesamiento de imágenes, el sistema pudo identificar y distinguir diferentes clases de desechos con gran precisión, incluso en entornos complejos o con condiciones de iluminación variables.

- Al emplear un sistema de visión artificial, se automatiza el proceso de detección de basura, lo que implica que no se requiere la intervención humana directa para realizar la tarea de reconocimiento, ahorrando tiempo y recursos.

- Con el empleo de un sistema de visión artificial, se posibilita la detección de basura en un área determinada. Esto permite una limpieza más eficiente y efectiva, pues los esfuerzos pueden concentrarse en los lugares específicos, donde se detecta basura en lugar de realizar una limpieza generalizada.

- YOLO es reconocido por su capacidad para lograr un equilibrio entre precisión y velocidad en comparación con otros enfoques de detección de objetos. Al implementar YOLO, es posible obtener resultados precisos en tiempo real, adecuándolo para aplicaciones en tiempo real y de alta velocidad.

- El éxito en el reconocimiento de objetos con YOLO se encuentra fuertemente influenciado por la disponibilidad y calidad de los datos de entrenamiento. La creación de un conjunto de datos de entrenamiento etiquetado y representativo resulta fundamental para alcanzar un rendimiento óptimo del modelo. Asimismo, la realización de un entrenamiento adecuado y la optimización de los hiperparámetros son elementos esenciales para obtener resultados óptimos.

RECOMENDACIONES

- Al construir la base de datos, compuesta por imágenes y archivos de bloc de notas, se recomienda mantener un procesamiento adecuado que permita asegurar coherencia y organización. Esto evita la pérdida o confusión de archivos durante el entrenamiento.

- Asimismo, se considera necesario sugerir una adecuada definición de archivo y rutas de acceso de las imágenes, así como archivos de bloc de notas. Esta práctica contribuirá a mantener la consistencia y facilitará la recuperación de datos en caso de ser necesario.

- Durante el entrenamiento, se recomienda aplicar técnicas de aumento de datos. Esto implica generar imágenes de entrenamiento adicionales, mediante transformaciones como rotación, cambio de escala, desplazamiento, cambio de brillo, entre otras. El aumento de datos contribuye a incrementar la diversidad de las imágenes y mejora la capacidad del modelo, para generalizar de manera más efectiva.

- Igualmente, con base en la experiencia de registro de la información, se recomienda que durante el proceso de grabación, para su posterior análisis, es importante mantener constante la intensidad de la luz, no sea excesiva, ya que esto podría causar errores en el programa de reconocimiento de objetos.

- Es esencial tener en cuenta las condiciones climáticas al capturar archivos multimedia con el dron. Fuertes vientos o, en todo caso, lluvias podrían ocasionar daños a la aeronave o provocar su estrellamiento.

REFERENCIAS BIBLIOGRÁFICAS

[1] Barahona Guamani, E. S. (2019). *Navegación autónoma basada en maniobras bajo estimación de posturas humanas para un robot omnidireccional kuka youbot* (Bachelor's thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Carrera de Ingeniería Electrónica y Comunicaciones).

[2] Boden, M. A. (2017). *Inteligencia artificial*. Turner.

[3] BODERO, E. M., LOPEZ, M. P., CONGACHA, A. E., CAJAMARCA, E. E., & MORALES, C. H. (2020). Google Colaboratory como alternativa para el procesamiento de una red neuronal convolucional. *Revista Espacios*, *41*(07).

[4] Buitrón Tandalia, I. A. (2021). *Análisis de rendimiento de you only look once, retinanet y single shot detector aplicado a la detección y conteo vehicular* (Master's thesis, Quito: EPN, 2021.).

[5] Damiani, L. (2019). Optimización estocástica acelerada con aplicación a la ingeniería de procesos.

[6] Del Río-Santana, O., Espinoza-Fraire, T., Sáenz-Esqueda, A., & Córtes-Martínez, F. (2019). Levantamientos topográficos con drones. *Revista Ciencia*, *1*.

[7] Endara Sumba, C. D., & Maigua Yánez, E. J. (2021). *Desarrollo de un algoritmo de trayectoria para un robot seguidor de línea destreza de competencia mediante visión e inteligencia artificial* (Bachelor's thesis).

[8] García-Sanz, M. P., Belmonte, M. L., & Galián, B. (2022). Barreras invisibles: validación de un instrumento para valorar la actitud hacia las personas con discapacidad intelectual. *Electronic Journal of Research in Education Psychology*, *20*(56), 151-176.

[9] Gómez-Zurdo, R. S., Martín, D. G., González-Rodrigo, B., Sacristán, M. M., & Marín, R. M. (2021). Aplicación de la fotogrametría con drones al control deformacional de estructuras y terreno. *Informes de la Construcción*, *73*(561), e379-e379.

[10] Layme, D. G., Hinostroza, H. L., & Aquise, R. S. (2023). Análisis granulométrico de agregado para el concreto a través de un algoritmo basado en redes neuronales (Deep Learning). *Revista Ingeniería de Construcción*.

[11] Pérez Zaldívar, L. (2021). *Algoritmos de aprendizaje automático para la predicción de la productividad de la confección textil* (Master's thesis, Universidad Internacional de Andalucía).

[12] Pilalumbo Armas, J. J. (2020). *Estudio de la Calidad de Agua del Río San Pedro, ubicado dentro del Distrito Metropolitano de Quito en el periodo 2013-2019* (Bachelor's thesis, Ecuador, Latacunga: Universidad Técnica de Cotopaxi UTC.).

[13] Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*, 20-21.

[14] Sánchez, A. A., & Tello, L. L. G. (2019). La contaminación ambiental en los acuíferos de Ecuador. *Revista Visión Contable*, (19), 64-101.

[15] Segura Medranda, O. D. (2019). *Diseño y construcción de un sistema electrónico para personas no videntes como ayuda para el cruce de las calles urbanas basado en el procesamiento de imágenes* (Bachelor's thesis, Escuela Superior Politécnica de Chimborazo).

[16] Soliz Torres, M. F., Durango Cordero, J. S., Solano Peláez, J. L., & Yépez Fuentes, M. A. (2020). *Cartografía de los residuos sólidos en Ecuador, 2020*. Quito, EC: Universidad Andina Simón Bolívar, Sede Ecuador/INEC/VLIR-UOS/GAIA/Alianza Basura Cero Ecuador/Acción Ecológica.

ANEXOS

Anexo 1

Dron utilizado para este proyecto

Anexo 2

Mando a distancia utilizado para controlar el dron.

Anexo 3

Características del dron

DJI Air 2	
Dimensiones	Plegado: 180×97×77 mm (largo × ancho ×alto) Desplegado: 183×253×77 mm (largo × ancho × alto)
Peso Longitud diagonal	595 gramos 302mm
Velocidad máxima de ascenso	6 m/s (modo S) 6 m/s (modo N)
Velocidad máxima de descenso	6 m/s (modo S) 6 m/s (modo N)
Ángulo de inclinación máximo	35° (Modo S) Frente: 30°, Atrás: 20°, Izquierda: 35°, Derecha: 35° (Modo N)
Velocidad angular máxima	250°/s (Modo S) 90°/s (Modo N) 60°/s (Modo C)
Rango de temperatura de funcionamiento	0° a 40°C (32° a 104°F)
Frecuencia de operación	2,4 GHz 5,8 GHz
Potencia de transmisión (EIRP)	FCC: ≤30 dBm CE: ≤20 dBm SRRC: ≤20 dBm MIC: ≤20 dBm FCC: ≤30 dBm CE: ≤14 dBm
Rango de precisión de vuelo estacionario	Vertical: ± 0,1 m (con posicionamiento por visión) ± 0,5 m (con posicionamiento por GNSS) Horizontal: ± 0,1 m (con posicionamiento por visión) ± 1,5 m (con posicionamiento por GNSS)

Hélices	Liberación rápida, bajo nivel de ruido, plegable.
Brazos de la nave	De tipo plegable
GNSS	GPS+GLONASS+GALILEO
Brújula	Brújula única
IMU	IMU individual
Almacenamiento interno	8GB

Anexo 4

Dron sobrevolando el río “San Pedro” para realizar las pruebas correspondientes

Anexo 5

Prueba de detección realizada a dos metros de altura en un entorno controlado

Anexo 6

Prueba de detección realizada a cuatro metros de altura en un entorno controlado

Anexo 7

Prueba de detección realizada a seis metros de altura en un entorno controlado

Anexo 8

Interfaz del programa

Anexo 9

Código

```
import numpy as np
import cv2
# Obtener el archivo de video del dron
file_video_stream = cv2.VideoCapture('images/testing/DJI_0530.mp4')
fourcc = cv2.VideoWriter_fourcc(*'MJPG')
frame_width = int(file_video_stream.get(3))
frame_height = int(file_video_stream.get(4))
size = (frame_width, frame_height)
out = cv2.VideoWriter('output.avi', fourcc, 20.0, size)
start_frame_number = 0
file_video_stream.set(cv2.CAP_PROP_POS_FRAMES, start_frame_number)
length = int(file_video_stream.get(cv2.CAP_PROP_FRAME_COUNT))
print(length)
c = 1
# Crear un bucle while
while c <= length:
    # Obtener el fotograma actual del flujo de video
    ret, current_frame = file_video_stream.read()
    # Utilizar el fotograma actual del video en lugar de la imagen
    img_to_detect = current_frame
    print(c * 100 / length)
    img_height = img_to_detect.shape[0]
    img_width = img_to_detect.shape[1]
    if c % 15 == 0:
        # convertir a blob para pasar al modelo
        img_blob = cv2.dnn.blobFromImage(img_to_detect, 0.003922, (320, 320), swapRB=True,
crop=False)
        # recomendado por los autores de YOLO, el factor de escala es 0.003922=1/255, el
ancho y alto del blob es 320,320
        # tamaños aceptados son 320×320,416×416,609×609. Más tamaño significa más precisión,
pero menos velocidad
        class_labels = ["Contenedores", "Tetrapack", "Botellas"]
        # Declarar solo un color
        class_colors = ["0,255,0", "0,0,255", "255,0,0", "255,255,0", "0,255,255",
"255,0,255", "155,155,155",
                        "155,0,155", "155,155,0", "50,50,50"]
        class_colors = [np.array(every_color.split(",")).astype("int") for every_color in
class_colors]
        class_colors = np.array(class_colors)
        class_colors = np.tile(class_colors, (3, 1))
        # Cargar el modelo personalizado de detección
        # obtener las predicciones de detección por el modelo usando el método forward()
        yolo_model = cv2.dnn.readNetFromDarknet('model\Detector.cfg',
'model\Detector_final.weights')
        # Obtener todas las capas de la red yolo
        # bucle y encontrar la última capa (capa de salida) de la red yolo
        yolo_layers = yolo_model.getLayerNames()
        yolo_output_layer = [yolo_layers[yolo_layer - 1] for yolo_layer in
yolo_model.getUnconnectedOutLayers()]
        # ingresar blob preprocesado al modelo y pasar a través del modelo
        yolo_model.setInput(img_blob)
        # obtener las capas de detección al pasar hasta la capa de salida
        obj_detection_layers = yolo_model.forward(yolo_output_layer)
        ############### Cambio NMS 1 ###############
        # inicialización para la supresión no máxima (NMS)
        # declarar lista para [id de clase], [centro de la caja, ancho y alto de la caja[]],
[confianzas]
        class_ids_list = []
```

```
        boxes_list = []
        confidences_list = []
        ############## Cambio NMS 1 FIN ###########
        # recorrer cada una de las salidas de capas
        for object_detection_layer in obj_detection_layers:
            # recorrer las detecciones
            for object_detection in object_detection_layer:
                # obj_detections[1 to 4] => tendrán los dos puntos centrales, ancho de la
caja y alto de la caja
                # obj_detections[5] => tendrá puntajes para todos los objetos dentro de la
caja delimitadora
                all_scores = object_detection[5:]
                predicted_class_id = np.argmax(all_scores)
                prediction_confidence = all_scores[predicted_class_id]
                # tomar solo predicciones con confianza mayor al 20%
                if prediction_confidence > 0.20:
                    # obtener la etiqueta de clase predicha
                    predicted_class_label = class_labels[predicted_class_id]
                    # obtener las coordenadas de la caja delimitadora para la imagen real
desde el tamaño de la imagen redimensionada
                    bounding_box = object_detection[0:4] * np.array([img_width, img_height,
img_width, img_height])
                    (box_center_x_pt,    box_center_y_pt,    box_width,    box_height)    =
bounding_box.astype("int")
                    start_x_pt = int(box_center_x_pt - (box_width / 2))
                    start_y_pt = int(box_center_y_pt - (box_height / 2))
                    ############## Cambio NMS 2 ###############
                    # guardar id de clase, inicio x, y, ancho y alto, confianzas en una lista
para el procesamiento de NMS
                    # asegurarse de pasar la confianza como flotante y el ancho y alto como
enteros
                    class_ids_list.append(predicted_class_id)
                    confidences_list.append(float(prediction_confidence))
                    boxes_list.append([start_x_pt,        start_y_pt,        int(box_width),
int(box_height)])
                    ############## Cambio NMS 2 FIN ###########
        ############## Cambio NMS 3 ###############
        # Aplicar el NMS devolverá solo los ids de valor máximo seleccionados mientras suprime
las cajas delimitadoras superpuestas no máximas (débiles)
        # la supresión no máxima se establece en 0.5 y el umbral de supresión máxima para NMS
se establece en 0.4 (ajustar y probar para un mejor rendimiento)
        max_value_ids = cv2.dnn.NMSBoxes(boxes_list, confidences_list, 0.5, 0.4)
        # recorrer el conjunto final de detecciones restantes después de NMS y dibujar la
caja delimitadora y escribir el texto
        for max_valueid in max_value_ids:
            max_class_id = max_valueid
            box = boxes_list[max_class_id]
            start_x_pt = box[0]
            start_y_pt = box[1]
            box_width = box[2]
            box_height = box[3]
            # obtener la etiqueta de clase predicha y el texto
            predicted_class_id = class_ids_list[max_class_id]
            predicted_class_label = class_labels[predicted_class_id]
            prediction_confidence = confidences_list[max_class_id]
        ############## Cambio NMS 3 FIN ###########
            end_x_pt = start_x_pt + box_width
            end_y_pt = start_y_pt + box_height
            # obtener un color de máscara aleatorio del array numpy de colores
            box_color = class_colors[predicted_class_id]
            # convertir la matriz numpy de color en una lista y aplicar al texto y la caja
            box_color = [int(c) for c in box_color]
            # imprimir la predicción en la consola
```

```
            predicted_class_label     =     "{}:     {:.2f}%".format(predicted_class_label,
prediction_confidence * 100)
            print("objeto predicho {}".format(predicted_class_label))
            # dibujar el rectángulo y el texto en la imagen
            cv2.rectangle(img_to_detect,  (start_x_pt,  start_y_pt),  (end_x_pt,  end_y_pt),
box_color, 3)
            cv2.putText(img_to_detect,  predicted_class_label,  (start_x_pt,  start_y_pt-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, box_color, 1)
            if c > (length - 2):
                out.release()
    # cv2.imshow("Resultado de la Detección", img_to_detect)
    out.write(img_to_detect)
    c = c + 1
    # terminar el bucle while si se presiona la tecla 'q'
    if cv2.waitKey(1) & 0xFF == ord('o'):
        break
# cerrar todas las ventanas de OpenCV
# file_video_stream.release()
out.release()
# cv2.destroyAllWindows()
```