Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Detección y Clasificación de Contaminantes Plásticos en Entornos Acuáticos

Autor: Pablo Fernández Barrera

Tutor: Daniel Gutiérrez Reina

Dpto. de Ingeniería de Electrónica

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Detección y Clasificación de Contaminantes Plásticos en Entornos Acuáticos

Autor:
Pablo Fernández Barrera

Tutor:
Daniel Gutiérrez Reina
Profesor titular

Dpto. de Ingeniería Electrónica

Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado: Detección y Clasificación de Contaminantes Plásticos en Entornos Acuáticos

Autor: Pablo Fernández Barrera
Tutor: Daniel Gutiérrez Reina

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla, 2024

Agradecimientos

Tras mucho esfuerzo he logrado terminar este proyecto. Este documento no es sólo un proyecto más, sino el broche final a una de las etapas más bonitas de mi vida, llena de días de penas y alegrías y de noches en vela y de diversión. Por eso quiero dedicar este primer capítulo a agradecer el todo el apoyo ofrecido durante todos estos años.

En primer lugar, me encantaría agradecerle a mi familia, por haber tenido una inagotable paciencia conmigo y por haberme apoyado incondicionalmente cada día de esta gran etapa Por quererme y ayudarme a progresar hasta el día de hoy y los que queden.

Por supuesto tengo que darles las gracias a mis amigos, que me han ayudado en este camino, a progresar tanto en lo académico como en lo personal. Por ser los apoyarme en los malos momentos y hacerme reír en os malos. Especial agradecimiento a mis amigos del grado, Elena, Carmen, Miguel y Sales, sin vosotros todo esto no hubiera sido posible. Ha costado, pero hemos terminado por fin.

Por último, darles las gracias a todos mis profesores del grado y, en especial, a Daniel Gutiérrez, por haberme dejado participar en un proyecto tan innovador y con impacto social siendo capaz de aportar a la gran disciplina de la Inteligencia Artificial que tanto me apasiona.

Gracias, os estoy eternamente agradecido.

Pablo Fernández Barrera

Sevilla, 2024

Resumen

Los océanos se están convirtiendo en auténticos vertederos gigantes. Se estima que entre el 60-80-% son plásticos y se prevé que vaya en aumento. Es en esta problemática donde se enmarcar este proyecto.

Con la finalidad de desarrollar un vehículo autónomo (ASV) que recoja la basura, en su mayoría plásticos, flotando en el agua, se pretende desarrollar un algoritmo de visión artificial que, mediante una cámara instalada en el vehículo, permita detectar la basura existente en la superficie oceánica.

Para ello haremos uso del algoritmo YOLOv5 que se especializa en la detección de objetos en imágenes. Este algoritmo será reentrenando con una fusión de distintos conjuntos de imágenes de basura en medio acuático con el objetivo de especializar el algoritmo de detección de objetos genérica al ámbito que interesa. De este modo, se pretende conseguir un algoritmo de detección de basura en imágenes lo suficientemente robusto como para ser ejecutado de manera autónoma a bordo de un ASV.

Abstract

The oceans are turning into giant landfills. It is estimated that 60-80-% of this is plastic and it is expected to increase. It is in this context that this project is framed.

With the aim of developing an autonomous vehicle (ASV) that collects rubbish, mostly plastics, floating in the water, we intend to develop an artificial vision algorithm that, by means of a camera installed in the vehicle, allows us to detect the rubbish on the ocean surface.

To do this, we will make use of the YOLOv5 algorithm, which specialises in detecting objects in images. This algorithm will be re-trained with a fusion of different sets of images of rubbish in the aquatic environment with the aim of specialising the generic object detection algorithm to the field of interest. In this way, the aim is to achieve an image-based object detection algorithm robust enough to be run autonomously on board an ASV.

Índice abreviado

Índice

1 INTRODUCCIÓN

En este capítulo introductorio, se dará una visión general del proyecto además de presentar tanto los objetivos de este como sus bases a fin de conocer los hitos y el alcance de este

1.1 Motivación

La contaminación de recursos hídricos es un problema con que el ser humano ha tenido que lidiar desde el principio de los tiempos. Las primeras evidencias que se tienen de contaminación del agua datan de hace 7000 años en rio Faynan en Jordania la cual se atribuye al trabajo del cobre en plena edad de los metales [1]. El agua ha jugado en papel fundamental en el desarrollo del ser humano tanto para la creación de recursos, como es el caso de la agricultura, como para la evacuación de desechos usando los ríos como gran cloaca en las ciudades. Así, no fue hasta milenios más tarde, con el surgimiento de la cultura musulmana donde se prohibió arrojar basura a los ríos de las ciudades, dándole al agua la protección que se merecía. Sin embargo, aunque los estados combatieron la contaminación acuática de manera individual, no fue hasta mediados del siglo XX que, debido a los residuos vertidos a los océanos por parte de las grandes potencias industriales, se empieza a percibir este problema como un asunto global y de vital importancia para la supervivencia y el bienestar de la humanidad en general [2].

En cuanto a los principales residuos a los que se debe hacer frente para lograr una mejora en las condiciones de los recursos hídricos, se pueden clasificar el residuos sólidos y líquidos. Los residuos sólidos tales como botellas, latas y basura en general, representan la parte más visible de este problema llegando a formar grandes islas artificiales del tamaño de países formadas por basura en descomposición. El problema de los residuos sólidos no solo se limita a ser un problema desde el punto de vista de la biodiversidad provocando asfixias y dificultad para alimentarse a la fauna marina, sino que representa un gran factor de riesgo para la salud de las personas a nivel global. La basura presente en el agua no solo genera suciedad y residuos que afectan a la calidad del agua, siendo una de las principales causas de contaminación de agua potable provocando enfermedades como el cólera, sino que la propia descomposición de la basura representa un riego incluso mayor.

Figura 1.1: Isla de basura en el Pacífico Norte. Fuente: *revistahyc.com*

Esta basura presente en los recursos hídricos se descompone en micropartículas, normalmente micro plásticos, cuya presencia en el agua genera una gran cantidad de problemas. En primer lugar, estos micro platicos, son ingeridos por los peces que habitan los océanos y mares que luego ingerimos, además de estar presentes en el agua que bebemos, de modo que pasan a nuestro torrente sanguíneo de manera fácil y continua en el tiempo. Además, este tipo residuos no sólo se limitan a los recursos hídricos, sino que llegan incluso a transmitirse a campos de cultivo y a animales terrestres muy alejados de los océanos por medio de la evaporación de agua contaminada con este compuesto y su posterior precipitación en forma de lluvia.

Figura 1.2: Micro plásticos encontrados en la orilla. Fuente: *paho.org*

La segunda gran categoría de residuos presentes en el medio acuático es la de los residuos líquidos, en esta categoría se agrupan todas las sustancias contaminantes que se usan en la agricultura, como pesticidas o fertilizantes, así como fármacos o productos de limpieza como legía o detergentes y desechos humanos como heces. Este tipo de desechos, aunque más indistinguibles a la vista que los anteriormente mencionados, representan un riesgo para la salud de los ecosistemas y las personas debido a que la presencia de estos productos en el agua representa y grave riego para la potabilidad del agua y hacerla inservible la agricultura y ganadería.

En cuanto a la eliminación de estos residuos, la solución más usada es la recolección de estos, pues así se elimina la presencia de residuos de gran tamaño y se evita que se descompongan en el agua formando micro prácticos. Esta parece ser la práctica más prometedora para la eliminación de estos residuos microscópicos ya que debido a su tamaño resulta muy costoso filtrarlos una vez presentes en el agua. Sin embargo, mayor parte de estas soluciones pasan por el uso de buzos que recolecten la basura a mano, lo que convierte a este tipo de limpiezas en una tarea costosa económicamente e ineficiente. En cuanto a los residuos líquidos, la solución más usada hoy en día pasa por tratar el agua afectada usando productos químicos y mediante la detección temprana de estas aguas contaminadas minimizar evitar su uso y minimizar sus daños en la medida de lo posible.

En este marco surge el proyecto ECOPORT [3], que trata de desarrollar un vehículo autónomo que recoja residuos de la superficie del agua mediante un sistema multirobot Este sistema consta de un ASV de detección de basura por medio de un algoritmo de Deep Learning aplicado a las imágenes de una cámara presente en el vehículo, que, una vez reconocidos los residuos, manda su localización a un segundo vehículo que procederá a su recolección.

Figura 1.3: *Prototipo de ASV*

1.2 Objetivos del trabajo

El principal objetivo del proyecto que integra a este trabajo es el desarrollo de un vehículo autónomo que se mueva sobre la superficie del agua y sea capaz de detectar la basura existente en la misma, proporcionando un desarrollo y una respuesta rápidos para evitar la propagación de agentes contaminantes en el mar. Además de este, existen otros tipos de objetivos asociados al proyecto, como la medición de parámetros que indiquen la calidad del agua, pudiendo determinar así la presencia de contaminantes líquidos en tiempo real y proporcionando una alerta temprana de este tipo de residuos.

Las principales características de la plataforma propuesta son la detección en tiempo real y alerta temprana, donde los ASV estarán equipados con sensores para medir las principales variables relacionadas con la contaminación del agua (nitratos, pH, conductividad, entre otros) y con cámaras para detectar derrames de petróleo y plásticos. El sistema permitirá la detección temprana de residuos en la superficie y una rápida actuación. Asimismo, se busca la robustez del sistema mediante el diseño y la implementación de estimadores y controladores avanzados y de alto rendimiento, permitiendo a la flota de ASV operar con seguridad en puertos marítimos donde haya vientos, pequeñas olas y otros barcos, utilizando las cámaras para evitar obstáculos fijos y móviles.

En este contexto, el objetivo de este trabajo es el desarrollo de un algoritmo de visión artificial usando el modelo YOLOv5 que sea capaz de llevar a cabo las tareas de detección de basura en tiempo real, usando como fuente de información las imágenes tomadas por la cámara existente en el vehículo. Como requisitos, este algoritmo deberá tener la suficiente precisión como para que el vehículo puede realizar sus tareas sin supervisión humana sin comprometer su funcionamiento, y además deberá de ser lo suficientemente óptimo como para ejecutarse en un equipo con recursos computacionales limitados como es el hardware del vehículo a una velocidad suficiente que permita analizar todas las imágenes provenientes de la cámara en tiempo real.

1.3 Estructura del trabajo

Para finalizar este apartado, se explicará de manera detallada la estructura del proyecto, destacando los elementos esenciales en el desarrollo del proyecto y posteriormente explicando cómo se estructura la memoria y dando una descripción de cada parte a modo de sinopsis.

Este proyecto se puede dividir en dos grandes bloques interconectados, la creación de un dataset y el reentrenamiento de YOLO. La composición de un conjunto de imágenes de basura en la superficie acuática se llevará a cabo fusionando varios datasets de basura en un medio acuático de modo que las imágenes seleccionadas sean lo más similares posibles a las tomadas por la cámara del vehículo. En cuanto al reentrenamiento de YOLO, una vez configurado este conjunto de imágenes se tratará de especializar a YOLO en reconocimiento de la basura indiciada. Después de este entrenamiento se llevará a cabo tanto pruebas con videos tomados en condiciones de funcionamiento similares a las del vehículo como en imágenes reales proporcionadas por pruebas realizadas en el mismo. En cuanto a la organización de la memoria, se pueden apreciar los siguientes capítulos.

1. Introducción. En este capítulo se describe brevemente las motivaciones que llevaron a la elección de este tema de trabajo, se establecen los hitos a cumplir y se presenta la estructura de la memoria.

2. Estado del arte. En este apartado se explorará las soluciones previas que se han propuesto para el mismo problema además de explicar donde se posiciona este proyecto con respecto a sus semejantes.

3. Metodología. En este capítulo se expondrá que procedimientos se han seguido a la hora de desarrollar el proyecto, además de explicar en detalle que herramientas y procedimientos utilizados.

4. Resultados. En este capítulo, se explicará de manera cronológica como se siguió el proceso descrito con anterioridad y cuáles fueron los resultados que se obtuvieron de su aplicación.

5. Conclusiones y futuros trabajos. Finalmente, aquí se expondrán las conclusiones inferidas de los resultados del proyecto y se explorarán sus limitaciones además proponer posibles mejoras para futuros desarrollos.

2 ESTADO DEL ARTE

Ante el inminente problema de la contaminación los recursos naturales , no son pocos los proyectos e investigaciones que han surgido basados en la utilización de la visión por computador para conseguir clasificar y detectar la basura de manera correcta. Así, este capítulo estará dedicado a explorar algunas de las soluciones más interesantes que existen ante este problema.

2.1 Detección y clasificación de residuos en general

En cuanto a la detección de residuos, se pueden encontrar diversos estudios y propuestas que abordan el problema de distinta manera y con distintos objetivos. Entre todos estos enfoques se destacan los siguientes.

En la ~~referencia~~ [4], se estudia el caso de uso de YOLOX para la detección de basura doméstica, desarrollando el algoritmo llamado i-YOLOX. En este estudio se puede apreciar los buenos resultados que arroja el reentrenamiento de YOLO, ~~ente~~ ante este tipo de tareas de detección. Se puede apreciar que se consigue una detección muy precisa sobre un conjunto de basura muy heterogéneo y en situaciones muy distintas que van desde zapatos tirados por el suelo, hasta botes de jabón distribuidos por un cuarto de baño. Aunque este proyecto cuenta con un enfoque similar al de este trabajo en cuanto al reentrenamiento y especialización del algoritmo de visión por computador YOLO, las principales diferencias residen en que, por un lado no está pensado para ejecutarse en un ASV de manera que no se explora el funcionamiento del algoritmo es una situación de manejo real de un ASV, por otro lado , el dataset usado para entrenamiento y validación contiene sólo basura tirada por el suelo , donde no existen problemas como la presencia de reflejos en el agua de luz o del propio residuo.

Algorithm	P/%	R/%	F1/%	mAP/%	Parameters/M	GFLOPs	FPS
Faster-RCNN [14]	54.67	**86.14**	66.67	81.70	137.04	370.14	11.79
YOLOv4 [40]	84.34	58.68	67.74	75.13	64.02	60.07	16.98
SSD [17]	84.15	77.15	80.25	82.08	25.88	62.45	**40.13**
YOLOv5 [32]	83.30	54.49	64.47	75.63	47.06	115.91	24.55
YOLOX [31]	85.78	80.06	82.58	85.68	8.94	26.79	26.37
i-YOLOX (**ours**)	**87.72**	81.96	**84.67**	**87.15**	**6.86**	**17.28**	37.01

Figura 2.1: Comparativa del rendimiento de i-YOLOX con otros modelos de detección. Fuente: [4]

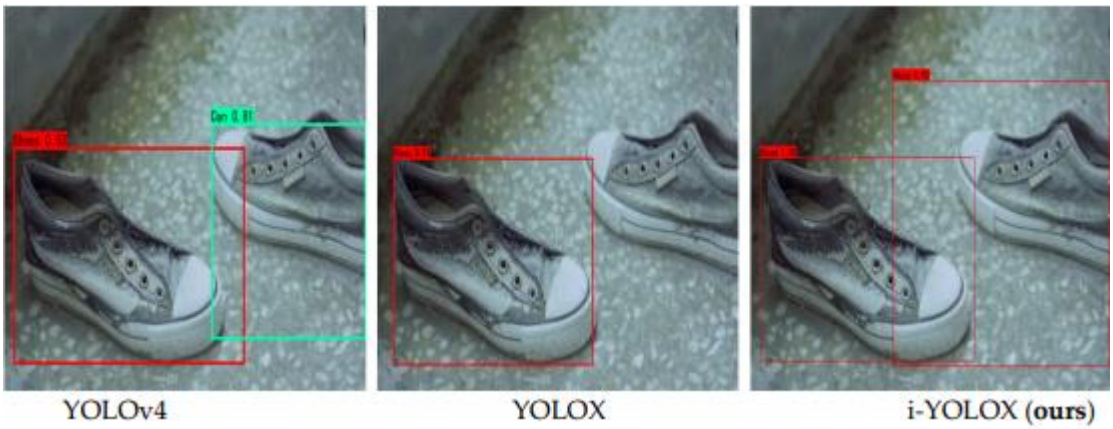

Figura 2.2: Comparativa en la detección de i-YOLOX con otras implementaciones de YOLO. Fuente: [4]

Siguiendo el enfoque del anterior proyecto, en [5] se presenta un desarrollo de un modelo de detección y clasificación que separa la basura urbana en categorías como papel, platicó u orgánica, con el objetivo de poder clasificarla para su futuro reciclaje. En este estudio, se fusionan una gran cantidad de dataset de residuos conocidos, TrashNet dataset [6] o TACO [7] para reentrenar el modelo EfficientDet-B2[8] y conseguir una detección y clasificación de la basura con resultados sorprendentemente precisos dada la diversidad de los objetos clasificados. Al igual que el proyecto anterior, las principales diferencias con este trabajo residen en que las imágenes de la basura no están tomadas desde un ASV ni son únicamente de residuos acuáticos. Además, este proyecto hace uso de un algoritmo de visión diferente que no está enfocado en la detección en tiempo real.

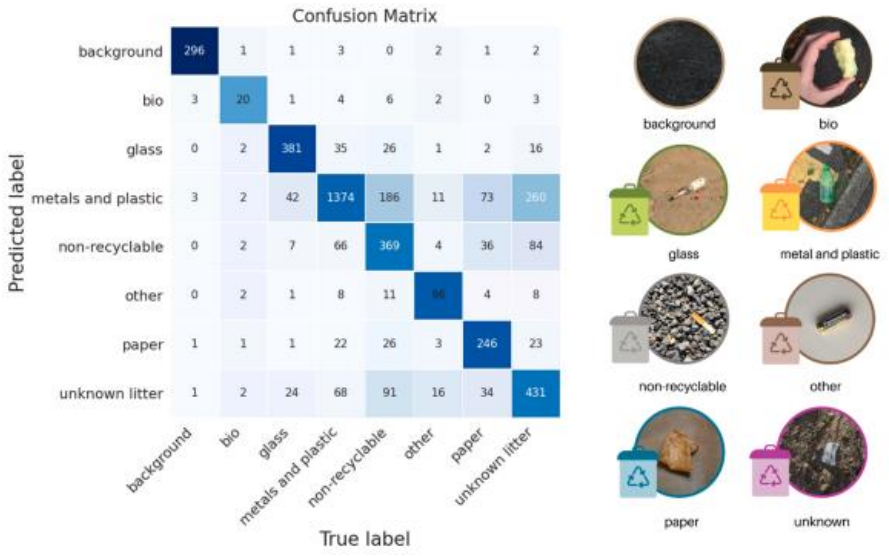

Figura 2.3: Matriz de confusión y categorías para clasificar en la detección.
Fuente: [5]

Como último proyecto que explora la detección de basura en general, [9] es un estudio enfocado a implementar un algoritmo de visión artificial que, al ser ejecutado a bordo de un camión de la basura, sea capaz de detectar los residuos presentes en las calles. Este proyecto se basa en la recolección de imágenes con el camión para formar posteriormente los datasets d entrenamiento y validación. Aunque este proyecto es similar a este trabajo en que se pretende ejecutar el algoritmo de visión en una estación móvil y realizar las tareas de clasificación y detección en tiempo real , hay dos diferencias sustanciales , la primera reside en que no se ha decido implementar YOLO , sino que se usa el modelo Overfeat-GoogLeNet [10][11] , y la segunda es que debido a las condiciones en las que se han tomado las imágenes de entrenamiento , aparecen casi exclusivamente colillas y hojas como residuos a clasificar. Aun así, los autores subrayan que este proyecto abre la posibilidad obtener una clasificación de basura más genérica si se consiguiera tomar imágenes con residuos más variados.

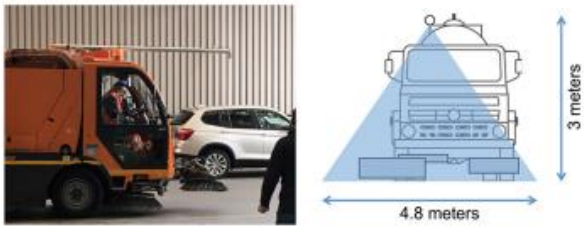

Figura 2.4: Camión de basura donde se corre el algoritmo. Fuente: [9]

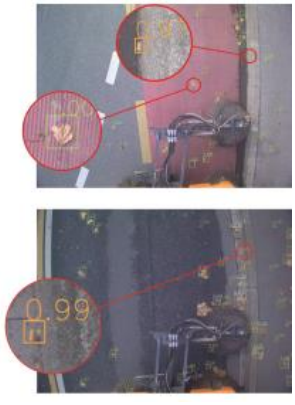

Figura 2.5: Ejemplo de detección de hoja y de colilla. Fuente: [9]

2.2 Detección de residuos en entornos acuáticos

Una vez explorados algunos desarrollos de dedicados a la detección y clasificación de la basura en entornos urbanos, es momento de analizar algunas soluciones ya existentes a las tareas de detección en entornos exclusivamente acuáticos.

La primera de ellas es [12], en donde se desarrolla un algoritmo basado en el reentrenamiento de YOLOv5 para la detección de basura subacuática. En este desarrollo, se hace uso del dataset ICRA19-Trash [13]que cuenta con un total de 7634 imágenes de basura en entornos acuáticos para reentrenar YOLO consiguiendo unos resultados bastante buenos aún en condiciones de baja visibilidad como baja luminosidad o alta turbidez en el agua. Al igual que con proyectos anteriormente mencionados, la principal diferencia con el desarrollo propuesto en este trabajo es que en ningún momento se prueban esos excelentes resultados en una implementación de ASV real, de modo que la gran precisión que demuestra sobre el dataset de validación, no se puede extrapolar de manera directa a un entorno real de funcionamiento.

Algorithm	mAP (%)	FPS (GPU)	FPS (CPU)	Inference (ms)	GFLOPS	Parameter
YOLOv5s	97.9	**35**	6	158.5	16.3	7059304
Proposed	97.5	**35**	**15**	**66.4**	3.0	826557

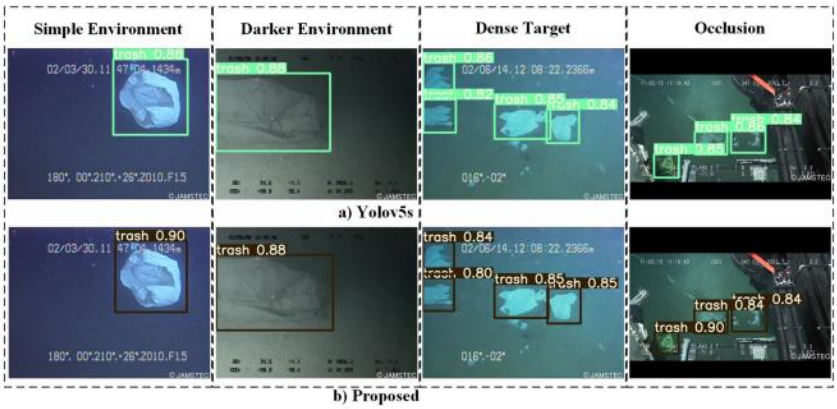

Figura 2.6: Comparativa de YOLOv5s por defecto y del modelo reentrenado.
Fuente: [12]

Por último, en el proyecto desarrollado en [14] explora la detección de basura en canales de agua, al igual que en este trabajo, mediante el uso de un algoritmo basado en una red neuronal llamada UNet [15]. Esta investigación se enfoca en la creación de un dataset de más de 13500 imágenes recolectadas mediante un amplio sistema de dispositivos IOT dispuestos en distintos canales de agua urbanos. Así, más que de desarrollar un modelo adecuado para la detección de basura, esta investigación pretende desarrollar en profundidad las problemáticas más importantes asociadas a los dataset de basura en recursos hídricos como pueden ser de pequeño tamaño muchos de los residuos o la baja precisión a la hora detectar objetos parcialmente sumergidos.

Figura 2.7: Extracto de casos problemáticos en el dataset.
Fuente: [14]

3 METODOLOGIA

Una vez definidos los objetivos y el contexto en el que sitúa la solución que aporta el proyecto, es necesario definir y explicar las herramientas y los procesos que se llevarán a cabo para cumplir con los objetivos propuestos. Así, este en este capítulo, se expondrán tanto los principales elementos usados, datos, algoritmo y hardware, como los procesos que se han seguido para integrar todas las partes.

3.1 Detección y clasificación de objetos

La detección y clasificación de objetos son dos tareas esenciales para cualquier sistema que busque interactuar eficazmente con un entorno real. Estas tareas se fundamentan en el procesamiento de la información obtenida del entorno en el que se encuentra el sistema, permitiéndole llegar a conclusiones y tomar decisiones. A continuación, se explorará los dos enfoques principales para la detección de objetos: mediante sensores tradicionales y mediante visión artificial.

Para detectar la presencia de un objeto mediante el uso de sensores el enfoque más habitual es la implementación de sensores de proximidad con LIDAR o Ultrasonido, que miden la distancia entre el objeto y el sistema de medida. Estas medias son precisas y robustas incluso en condiciones adversas, sin embargo, ofrecen información únicamente de un ámbito en concreto de la realidad, lo que puede ser suficiente para asegurar la detección de la presencia del objeto, pero suelen ser insuficiente al tratar de realizar una clasificación. Esto se debe a que para realizar tareas de clasificación en general se necesita información diferentes magnitudes físicas, como la forma, el color, o el material para llegar a una conclusión, lo que se traduce en una gran cantidad y variedad de sensores que deben funcionar de manera conjunta y en algoritmos muy complejos.

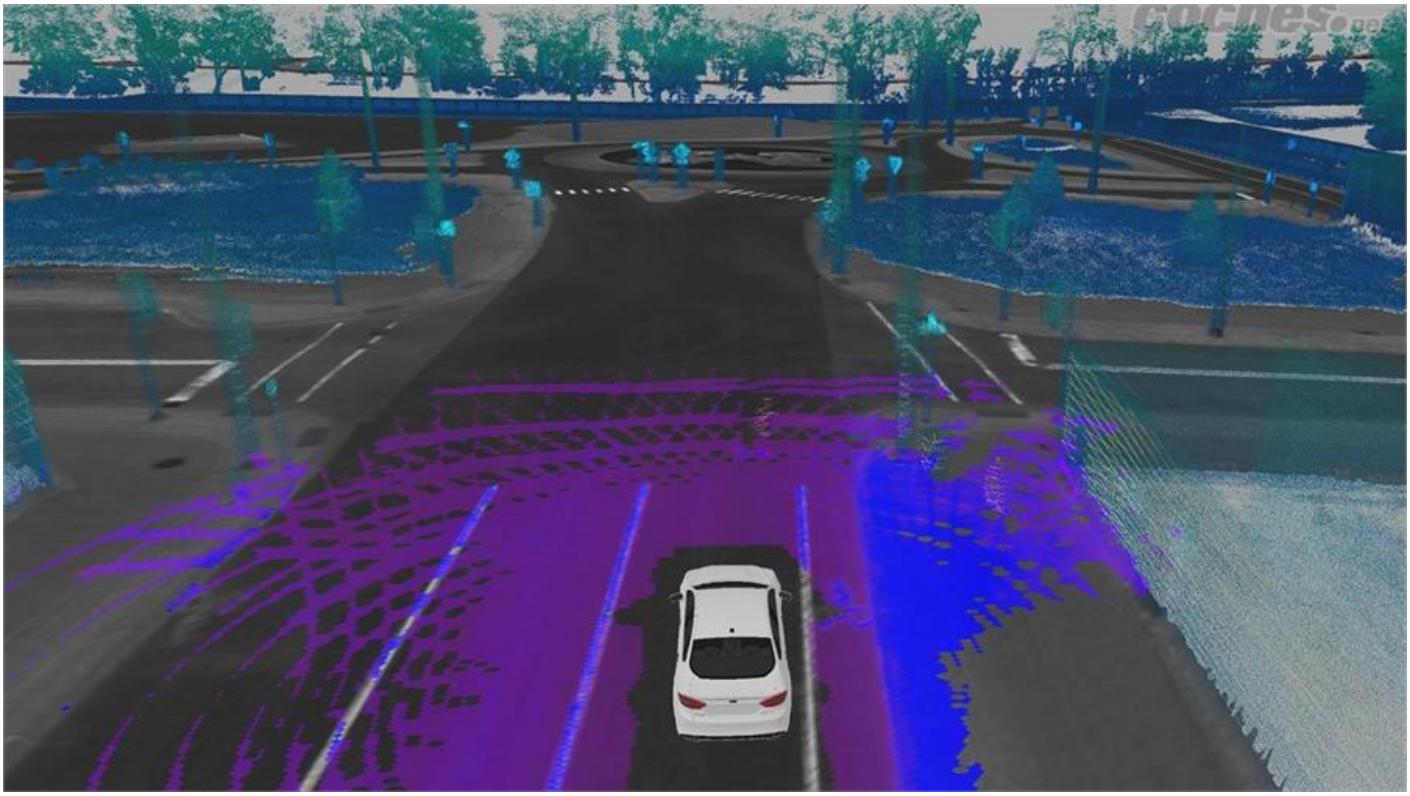

Figura 3.1: Ejemplo de uso de LIDAR en detección de obstáculos durante la conducción. Fuente: *coches.net*

Ante la problemática de tener que integrar gran cantidad de sensores para poder llevar a cabo tareas de detección y clasificación, surge la tecnología de visión artificial como la principal solución. La visión artificial o visión por computador es un método de detección y clasificación basado la aplicación de algoritmos de procesamiento a las imágenes obtenidas del medio físico. Estas imágenes permiten obtener información variada como el color o la forma desde un único sensor y algoritmo de procesamiento de información. Este enfoque ha permitido que esta solución se alce como la más prometedora en los últimos años, donde gracias al aumento de la capacidad y la velocidad de cómputo en los dispositivos, es posible llevar a cabo la gran cantidad de cómputo que requiere procesar imágenes.

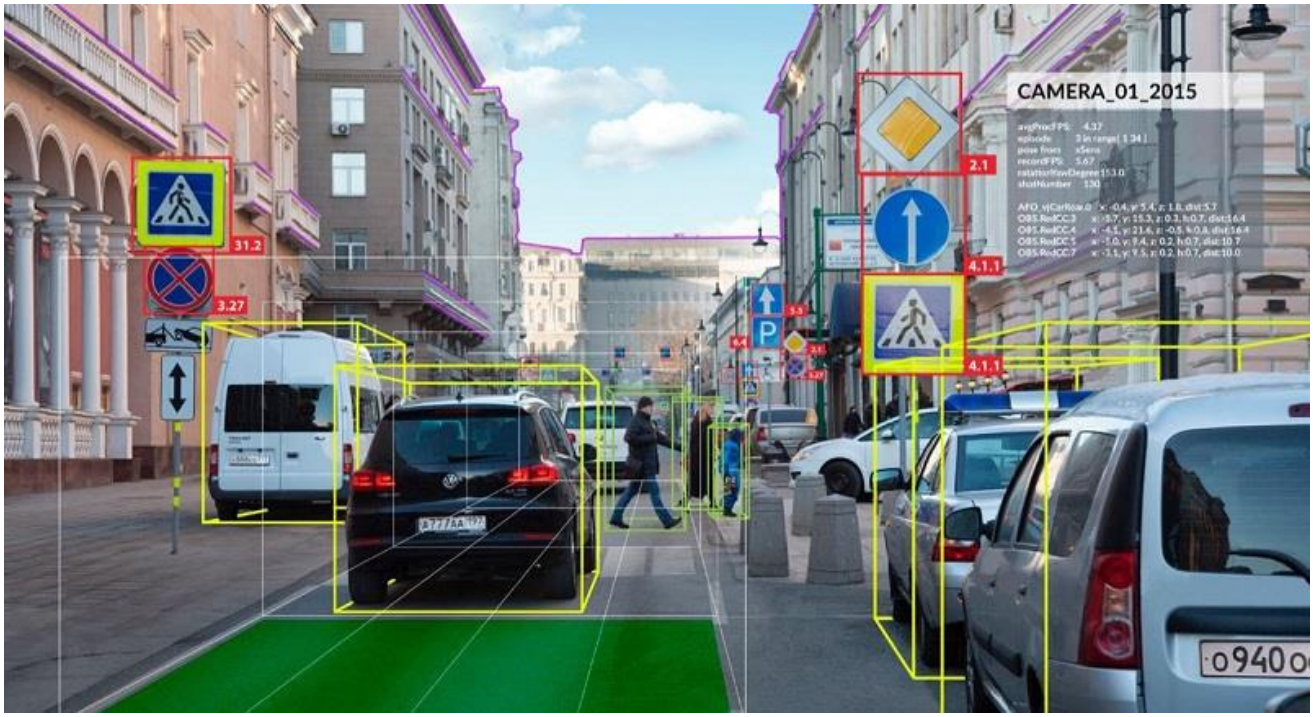

Figura 3.2: Ejemplo de uso de visión artificial durante la conducción. Fuente: *ignaciogavilan.com*

3.1.1 Algoritmos de Visión Artificial

Una vez analizadas las ventajas e inconvenientes del uso de sensores tradicionales y del uso de la visión artificial, es momento de explorar los distintos tipos de algoritmos de visión. Esta disciplina ha evolucionado significativamente, abarcando desde métodos tradicionales basados en técnicas de procesamiento de imágenes hasta enfoques modernos impulsados por inteligencia artificial. En este apartado, se explorarán tanto los métodos tradicionales de visión artificial como aquellos basados en el uso de Inteligencia Artificial.

Los métodos tradicionales de visión artificial se basan en técnicas de procesamiento de imágenes y análisis de características. Estas técnicas incluyen transformaciones de imagen, detección de bordes, segmentación y extracción de características. Herramientas matemáticas como las transformadas de Fourier y las transformadas de Hough son comúnmente utilizadas para analizar las imágenes y extraer información relevante. Por ejemplo, la detección de bordes se emplea para identificar los límites de los objetos dentro de una imagen, mientras que la segmentación divide una imagen en regiones significativas para facilitar su análisis. Estos métodos requieren un diseño cuidadoso de algoritmos específicos para cada tarea y suelen depender de parámetros ajustados manualmente.

Figura 3.3: Ejemplo de extracción de bordes con Python. Fuente: *kipunaec.com*

Con el avance de la inteligencia artificial, los métodos de visión artificial han incorporado técnicas de aprendizaje automático que permiten a las máquinas aprender directamente de los datos. Algoritmos como las Máquinas de Soporte Vectorial (SVM), los Bosques Aleatorios y los Modelos de Mezcla de Gaussianas han demostrado ser efectivos para tareas como la clasificación de imágenes y el reconocimiento de patrones. Estos enfoques aprenden a partir de grandes conjuntos de datos etiquetados, permitiendo una mayor precisión y adaptabilidad en comparación con los métodos tradicionales. Además, técnicas como el aprendizaje profundo han revolucionado el campo, permitiendo a las máquinas aprender representaciones de alto nivel directamente desde los datos en bruto. Gracias a este tipo de técnicas de Aprendizaje profundo se consiguen algoritmos capaces de llevar a cabo tareas de detección y clasificación de objetos basándose en ejemplos previos analizados y consiguiendo así algoritmos que detectan y clasifican conjuntos de objetos con características de forma o color muy diversos como es el caso de la basura.

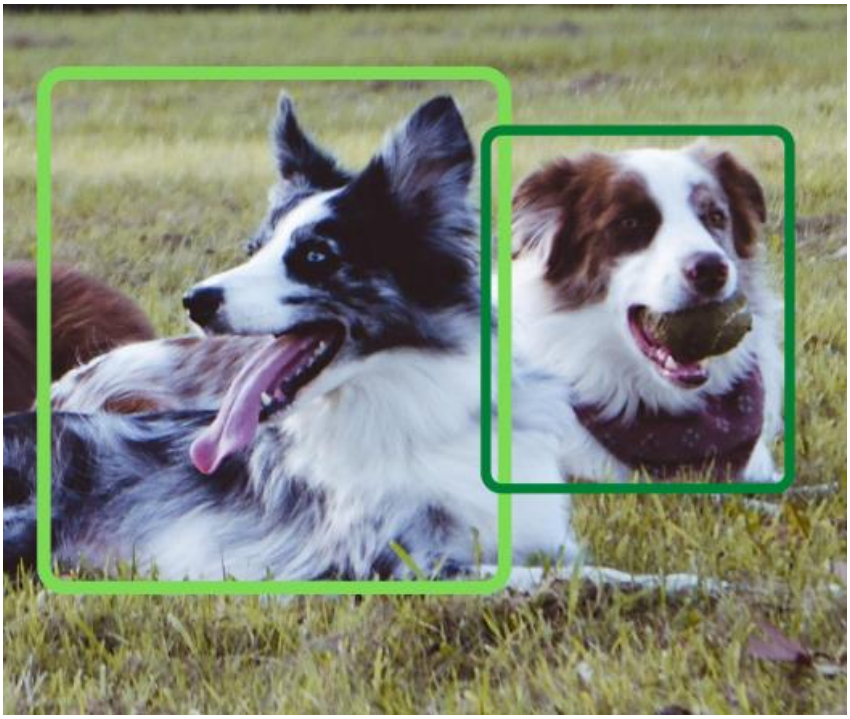

Figura 3.4: Ejemplo de detección y clasificación de “Perro” usando Deep Learning. Fuente: *aprendemachinelearning.com*

3.1.2 Aprendizaje profundo

El Aprendizaje profundo o *Deep Learning*, una subdisciplina del aprendizaje automático se basa en la idea de emular la estructura y funcionamiento del cerebro humano mediante redes neuronales artificiales. En el corazón de estas redes se encuentran las neuronas artificiales, que son la unidad básica de procesamiento. Inspiradas en las neuronas biológicas, cada neurona artificial recibe múltiples entradas, las procesa mediante una combinación ponderada, y aplica una función de activación para producir una salida. Matemáticamente, una neurona puede ser representada como (1), donde x_i son las entradas, w_i los pesos que modulan la importancia de cada entrada, b el sesgo que ajusta la salida, y f la función de activación que introduce no linealidad en el modelo, permitiendo a la red neuronal aprender y representar relaciones complejas en los datos.

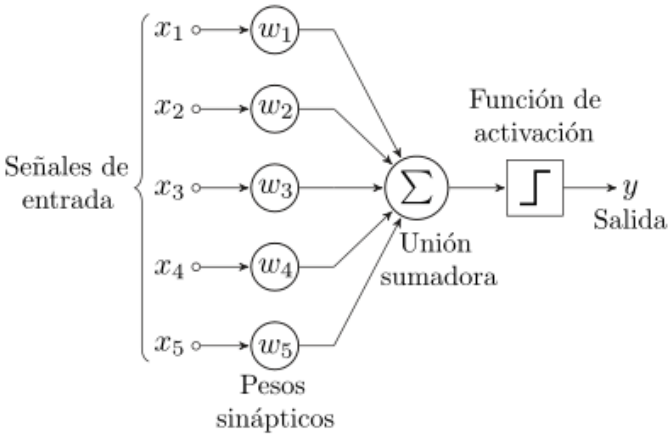

Figura 3.5: Esquema de la estructura de una neurona. Fuente: *ibm.com*

$$y = f(\sum x_i w_i + b)$$

Ecuación 3.1: Función de una neurona (1)

Las redes neuronales están compuestas por capas de estas neuronas, organizadas en una arquitectura específica. La primera capa, conocida como capa de entrada, recibe los datos brutos. A continuación, una o más capas ocultas realizan el procesamiento intermedio, y finalmente, la capa de salida produce el resultado final. Las conexiones entre neuronas de diferentes capas están determinadas por los pesos sinápticos, que se ajustan durante el proceso de entrenamiento para minimizar el error de predicción. Este ajuste se realiza a través de algoritmos de optimización como el descenso de gradiente.

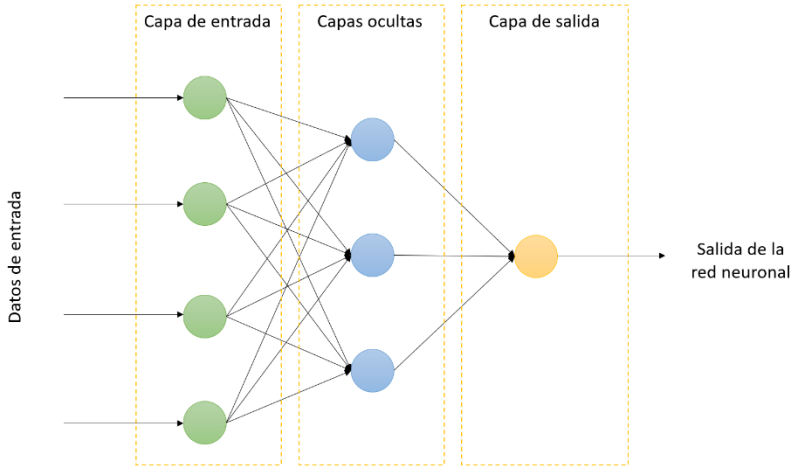

Figura 3.6: Esquema de red neuronal. Fuente: *iteractivechaos.com*

Un componente crucial en el entrenamiento de redes neuronales es la función de pérdida, también conocida como función de costo. Esta función cuantifica la discrepancia entre las predicciones de la red y los valores reales esperados. El objetivo del entrenamiento es minimizar esta pérdida, ajustando los pesos y sesgos de la red para mejorar su capacidad predictiva. Existen varias funciones de pérdida, dependiendo del tipo de problema, las más usadas son el error cuadrático medio para problemas de regresión y la entropía cruzada categórica es frecuente en tareas de clasificación.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Ecuación 3.2: Error Cuadrático Medio

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

Ecuación 3.3: Entropía cruzada categórica

Para asegurar que las redes neuronales puedan capturar patrones complejos, se emplean funciones de activación no lineales como ReLU (*Rectified Linear Unit*), Sigmoid y Tanh. Estas funciones permiten que las redes neuronales aprendan relaciones no lineales en los datos, lo que es esencial para resolver problemas complejos de reconocimiento de patrones y toma de decisiones. Sin funciones de activación no lineales, las redes neuronales serían equivalentes a un modelo lineal, limitando significativamente su capacidad para modelar problemas del mundo real.

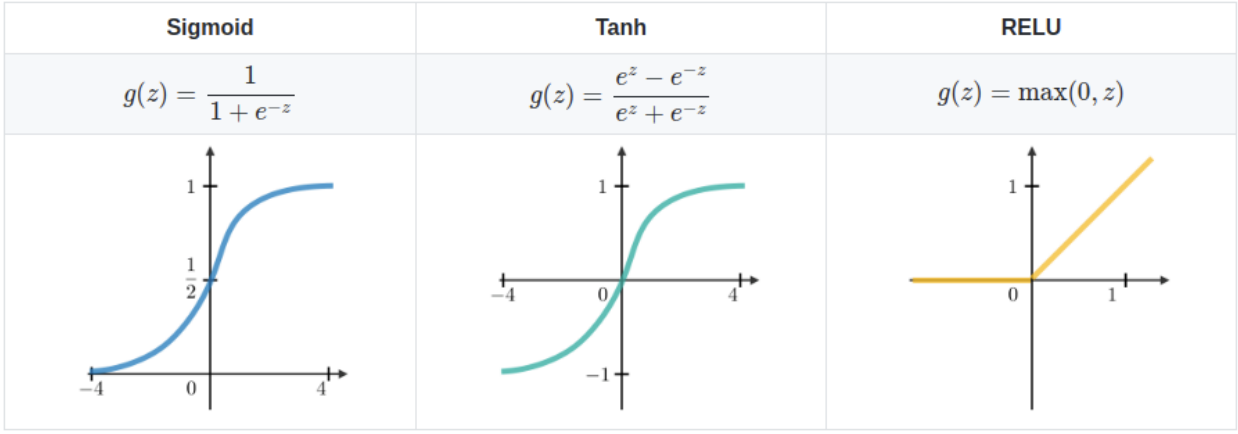

Figura 3.7: Ejemplo de funciones de activación más comunes. Fuente: *studymachinelearning.com*

En resumen, el Deep Learning se basa en la interacción compleja de neuronas artificiales organizadas en redes profundas, donde cada neurona aplica funciones de activación a combinaciones ponderadas de entradas. El entrenamiento de estas redes implica la minimización de una función de pérdida mediante algoritmos de optimización, permitiendo que la red aprenda y generalice a partir de datos. Este enfoque no sólo ha revolucionado campos como la visión artificial, sino que también se aplica en muchos otros campos como es el procesamiento de lenguaje natural.

3.1.2.1 Redes Neuronales Convolucionales

Dentro de las redes neuronales, las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) son una clase de redes neuronales especialmente efectivas para procesar datos con estructura de cuadrícula, como imágenes. Las CNN han revolucionado el campo de la visión por computadora debido a su capacidad para capturar características espaciales y patrones jerárquicos en los datos. A diferencia de las redes neuronales totalmente conectadas, donde cada neurona de una capa está conectada a todas las neuronas de la capa siguiente, las CNN emplean capas convolucionales que utilizan filtros para explorar y detectar características locales en las entradas.

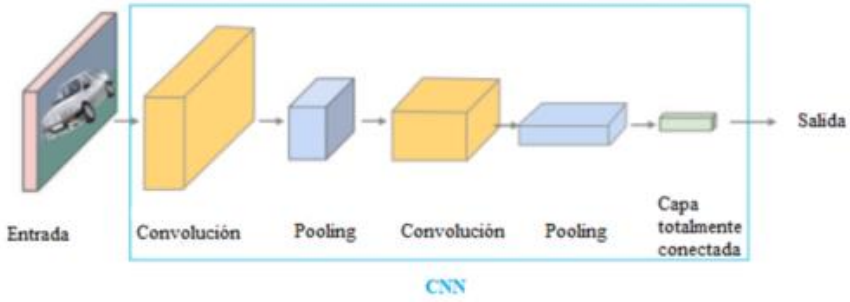

Figura 3.8: Esquema de red neuronal convolucional. Fuente: *researchgate.net*

En una CNN, la capa convolucional es fundamental. En esta capa se aplican una serie de filtros (o *kernels*) sobre la entrada, realizando operaciones de convolución para producir mapas de características. Cada filtro de esta capa es responsable de detectar características específicas, como bordes, texturas o colores. A medida que se aplican múltiples filtros, la red puede aprender a reconocer combinaciones de características básicas que forman patrones más complejos. La operación de convolución preserva la relación espacial entre los píxeles al considerar solo pixeles cercanos, lo que resulta esencial para mantener la estructura de la imagen. Otro componente clave de las CNN es la capa de *pooling*, que reduce las dimensiones espaciales de los mapas de características mediante operaciones como el *max-pooling* o el *average-pooling*. El *max-pooling* toma el valor máximo de una región de la entrada, reduciendo así el tamaño del mapa de características y permitiendo una forma de invariancia a la traslación. Este proceso no solo disminuye la cantidad de parámetros y el costo computacional, sino que también ayuda a la red a ser más robusta frente a pequeñas variaciones y desplazamientos en las imágenes. Las CNN también incluyen capas totalmente conectadas al final de la arquitectura, que toman las características extraídas y reducidas por las capas convolucionales y las utilizan para realizar la clasificación final. Estas capas funcionan de manera similar a las neuronas en una red neuronal tradicional, donde cada neurona recibe entradas de todas las neuronas de la capa anterior, permitiendo una combinación ponderada de características para realizar la decisión final. La salida de estas capas pasa por una función de activación haciendo posible calcular la probabilidad de que cierta clase exista en una imagen dada.

El entrenamiento de las CNN sigue el mismo principio de minimización de una función de pérdida mediante algoritmos de optimización como el descenso de gradiente. Sin embargo, debido a la naturaleza convolucional y las operaciones de *pooling*, las CNN son capaces de aprender características jerárquicas y espaciales de manera más eficiente y efectiva que las redes totalmente conectadas. Esto ha permitido a las CNN alcanzar resultados sobresalientes en tareas de reconocimiento de objetos, detección de rostros, segmentación de imágenes y muchas otras aplicaciones en visión por computadora.

3.1.3 Métodos análogos a YOLO

Una vez descrito y analizado el funcionamiento de las redes neuronales y cómo es posible usarlas en tareas de visión por computador, es momento de analizar diferentes arquitecturas de redes neuronales alternativas a YOLO con el fin de poder establecer una comparativa entre las distintas herramientas existentes para llevar a cabo tareas de detección y clasificación de objetos en imágenes.

En primer lugar, la arquitectura R-CNN (*Region-based Convolutional Neural Networks*) [16] destaca por el empleo de algoritmo externos como *Selective Search* para generar regiones candidatas que podrían contener objetos en la imagen. Cada región candidata se redimensiona y pasa a través de una CNN preentrenada para extraer las características más relevantes lo que permite a R-CNN y sus variantes alcanzar una alta precisión en la detección de objetos. Sin embargo, este enfoque es generalmente más lento que YOLO ya en lugar de realizar una detección en una sola etapa, R-CNN genera primero propuestas de regiones candidatas y luego aplica una red convolucional para clasificar cada región. Versiones de esta arquitectura como Fast R-CNN [17] mejoran la eficiencia compartiendo cómputo de convolución, y Faster R-CNN [18] introduce una red de propuestas de regiones (RPN) que genera regiones con alta probabilidad de contener la clase deseada para acelerar el proceso de cómputo aún más, pero aun así no alcanza la velocidad de YOLO en aplicaciones de tiempo real. Por lo tanto, si se necesita una detección rápida y eficiente, YOLO sigue siendo una opción superior.

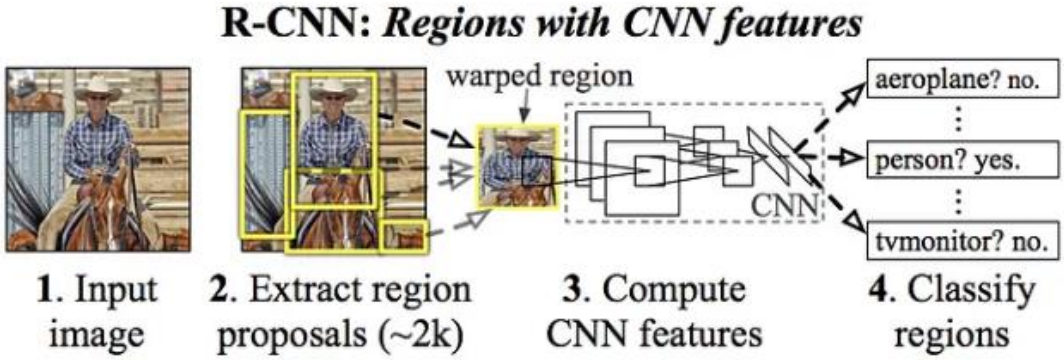

Figura 3.9: Esquema del funcionamiento de R-CNN. Fuente: *damavis.com*

La siguiente arquitectura a considerar es SSD (Single Shot MultiBox Detector) [19], que, al igual que YOLO, realiza detección de objetos en una sola etapa sin necesidad de generar propuestas de regiones por separado. Esto permite a SSD operar a velocidades comparables a YOLO, siendo también adecuado para aplicaciones en tiempo real. SSD divide la imagen en una cuadrícula y realiza predicciones de clases y cajas delimitadoras directamente para cada celda. Una de las ventajas de SSD sobre YOLO es su capacidad para manejar múltiples escalas de características, lo que mejora la detección de objetos de diferentes tamaños. Aunque SSD y YOLO tienen rendimientos similares en términos de velocidad, la arquitectura de múltiples escalas de SSD puede ofrecer una ligera ventaja en precisión para ciertos conjuntos de datos. A pesar de esto, para muchas aplicaciones prácticas, la simplicidad y eficiencia de YOLO lo hacen más atractivo.

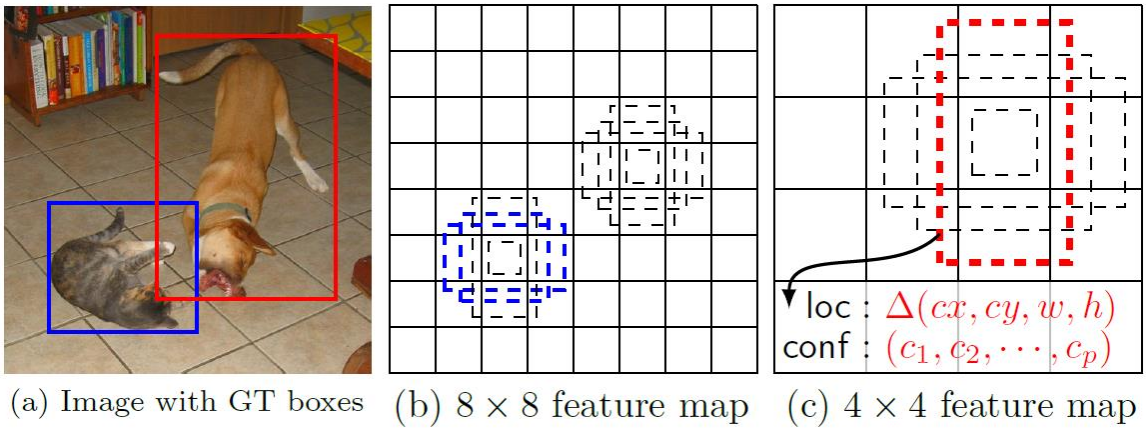

Figura 3.10: Ejemplo de creación de cuadrículas con SSD. Fuente: *towardsdatascience.com*

Por último, EfficientDet [20], es un enfoque más reciente que se basa en el modelo EfficientNet conocido por su eficiencia en términos de precisión y velocidad. Comparado con YOLO, EfficientDet utiliza una arquitectura de red llamada BiFPN (*Bidirectional Feature Pyramid Network*) que facilita la fusión de características a diferentes resoluciones, mejorando tanto la precisión como la eficiencia computacional. BiFPN permite la combinación bidireccional de características, lo que significa que las características se combinan de manera tanto ascendente como descendente en términos de abstracción, mejorando la capacidad de la red para detectar objetos de diferentes tamaños y contextos con mayor precisión. EfficientDet ofrece un equilibrio excelente entre rendimiento y velocidad, lo que puede superar a YOLO en términos de eficiencia y precisión en algunos casos. Esto hace que EfficientDet sea particularmente adecuado para aplicaciones móviles y otras situaciones con recursos limitados, donde es crucial optimizar el uso de recursos computacionales. Aun así, YOLO sigue siendo la opción más usada debido a su implementación más sencilla y su rendimiento probado en tiempo real.

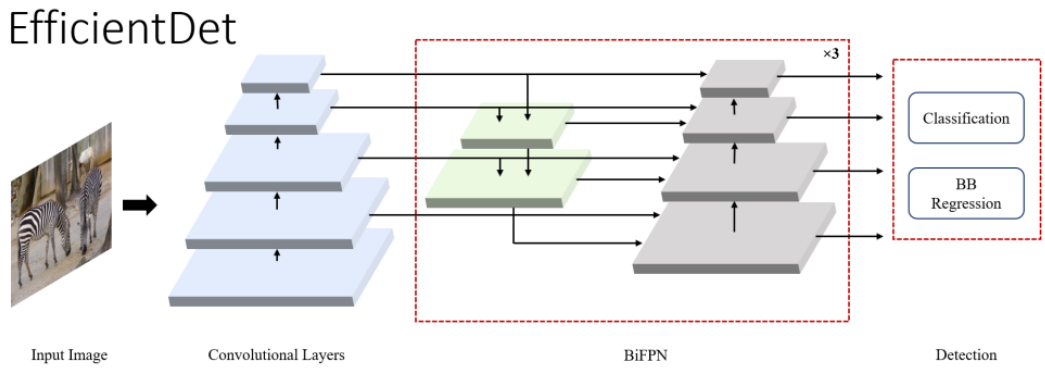

Figura 3.11: Arquitectura EfficientDet. Fuente: *gdpicture.com*

3.2 YOLO como algoritmo detección

Una vez analizadas las diferentes alternativas para la creación de un algoritmo de detección y clasificación de objetos, es momento de explicar en detalle el funcionamiento de la que se usará en este proyecto que es YOLO (You Only Look Once). Esta herramienta hace uso de una red neuronal convolucional entrenada para la detección de objetos en general, la cual especializaremos para la detección de residuos en recursos hídricos mediante un reentrenamiento. El enfoque de YOLO, que se basa en la localización y detección en un solo modelo, lo hace extremadamente eficiente y apto para tareas que requieran el procesado de las imágenes en tiempo real, como es este caso.

3.2.1 Origen de YOLO

YOLO surge en 2015 como un algoritmo revolucionario en el campo de la visión por computador debido a su enfoque de una sola pasada, que le permite realizar tareas de detección y clasificación con una combinación de eficacia y rapidez nunca vista. Sin embargo, esta primera versión de YOLO ~~flaqueaba~~ a la hora de detectar objetos pequeños y era impreciso en la locación de los mismo.

En la segunda versión de YOLO de 2016, YOLOv2 se mejoró la arquitectura de red para permitir filtros más pequeños y *Bounding boxes* [21] variables, lo que mejoró mucho las debilidades anteriores. Siguiendo esta línea, en 2018 con YOLOv3 se logró un gran salto en la precisión y velocidad del modelo gracias a la implementación de un nuevo modelo llamado “Darknet-53” además del uso de la técnica FPN (Feature Pyramid Network) [22] que permitía capturar características con diversos niveles de detalle.

No fue hasta 2020 que se lanzó YOLOv4 el cual introdujo mejoras importantes en la precisión y velocidad gracias a técnicas como “Bag of Freebies” [23] y CSPDarknet53.Además ese mismo año se lanzó YOLOv5 que simplificada y optimizaba esta arquitectura.

Posteriormente los años siguientes se han desarrollado versiones como YOLOv6 y YOLOv8 que son de código abierto. Estas versiones son capaces de ofrecer mejores características de rendimiento y fiabilidad, aunque para este proyecto nos quedaremos con la versión YOLOv5 debido a su ampliamente demostrada robustez y rendimiento a lo largo de estos años.

3.2.2 YOLOv5

Al ser YOLOv5 la versión que se usará en este proyecto, se usará este apartado para explicar más a fondo las características de esta implementación. Como se dijo anteriormente, YOLOv5 se lanza en 2020 de la mano de Glen Jocher y Ultralytic y su principal diferencia con las anteriores versiones es el uso de PyTorch y de CSPDarknet53.Este algoritmo CSPDarknet53, consiste en una implantación de Darknet53 que usando *Cross Stage Partial* Network (CSPNet) [24] que divide los datos de entrada para que sean más fáciles de procesar para posteriormente combinar las partes formando una representación más extensa de los datos de entrada. Esta versión cuenta también con diferentes modelos preentreandos donde varía su velocidad y precisión en función del número de parámetros que tiene.

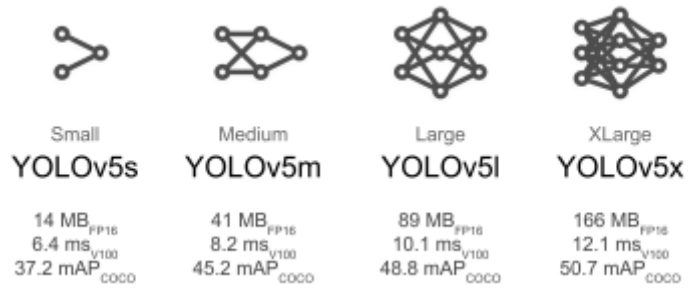

Figura 3.12: Modelos de YOLOv5

Como se puede observar, conforme los modelos se hacen más pesados (de 14MB a 166MB), su velocidad se reduce y su precisión aumenta. Para aplicaciones que se ejecuten en la nube se recomiendan modelos pesados que dan mejores resultados en la mayoría de los casos y para desarrollos en plataformas móviles se recomienda versiones como YOLOv5s o YOLOv5m [25]. Para este proyecto usaremos la versión YOLOv5s, que permite una buena precisión y sobre todo una gran velocidad y eficiencia en términos de recursos computacionales, que se adaptan a la perfección a las limitaciones y necesidades del ASV. En concreto se usará el checkpoint yolov5s6, que presenta mejoras significativas a la hora de reconocimiento en video de alta resolución. [26].

3.2.5 Funcionamiento de YOLO

El enfoque de YOLO se basa en la detección, clasificación y localización de objetos mediante el uso de una red neuronal convolucional (CNN) que permite por medio de un aprendizaje profundo (*Deep Learning*) conseguir un resultado preciso y rápido. Esta CNN se compone de tres partes fundamentales, *Backbone, Neck* y *Head*. El *Backbone* es el encargado extraer un mapa de características de la imagen que se da como entrada al sistema. Esto lo hace gracias a distintas capas convolucionales que aplican distintos filtros a las regiones de la imagen para conseguir capturar características de nivel más abstracto. Posteriormente, este mapa de características de más alto nivel pasa al *Neck*, donde de tratará de agregar la máxima información posible. Esta tarea se realiza mediate capas de aumento de resolución (*Upsample*) y técnicas ya mencionadas anteriormente como FPN (*Feature Pyramid Network*) y el CSP (*Cross Stage Partial)* siendo esta capa clave para la correcta detección de pequeños detalles y objetos que han podido perderse al subir el nivel de abstracción. Por último, el *Head* se encarga de inferir la clase y la región donde se encuentra el objeto basándose en el resultado del *Neck* [27].

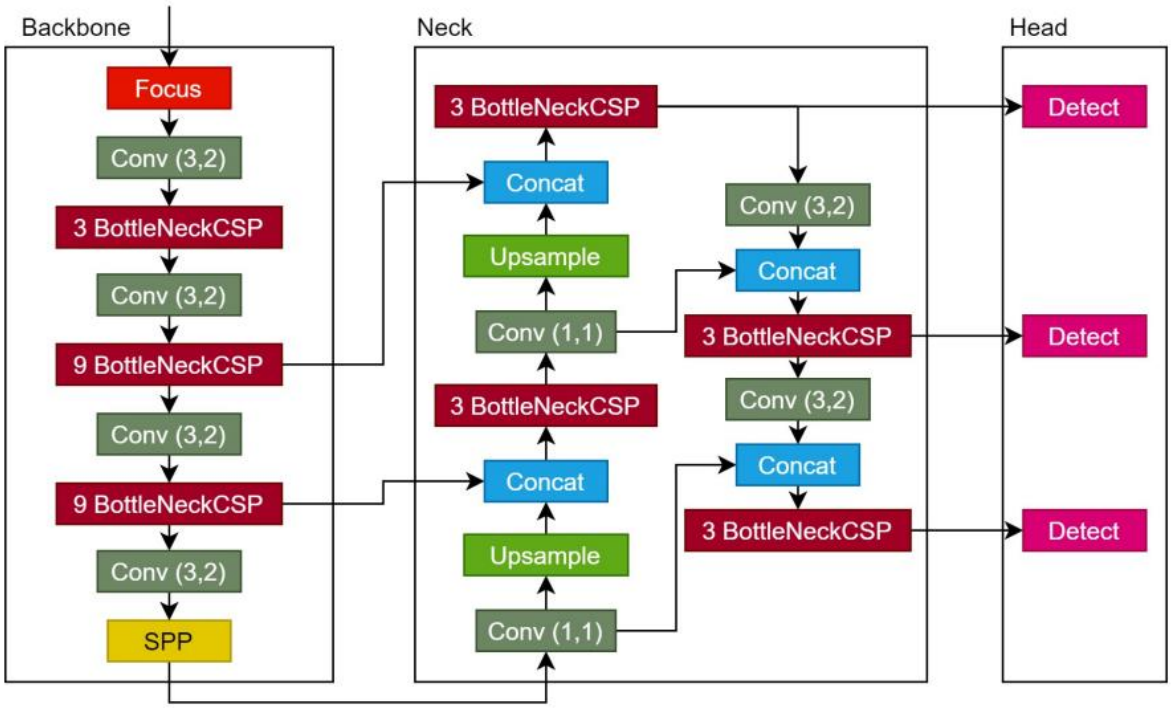

Figura 3.13: Red neuronal de YOLOv5 por defecto Fuente: *arxiv.org*

El proceso de detección comienza con una imagen como entrada de la red neuronal. Esta imagen será divida en pequeñas cuadrículas S x S y para cada celda se predicen varios rectángulos que delimitan la posición del posible objeto llamados *Bounding Boxes*. Estas celdas se caracterizan por medio de las coordenadas X e Y de su centro y ancho y alto en píxeles. Además, son en estas celdas las unidades mínimas para las que la red calcula las probabilidades de que un cierto objeto este o no en esa región. De este modo, se filtran todas las celdas que presenten una probabilidad de clase baja.

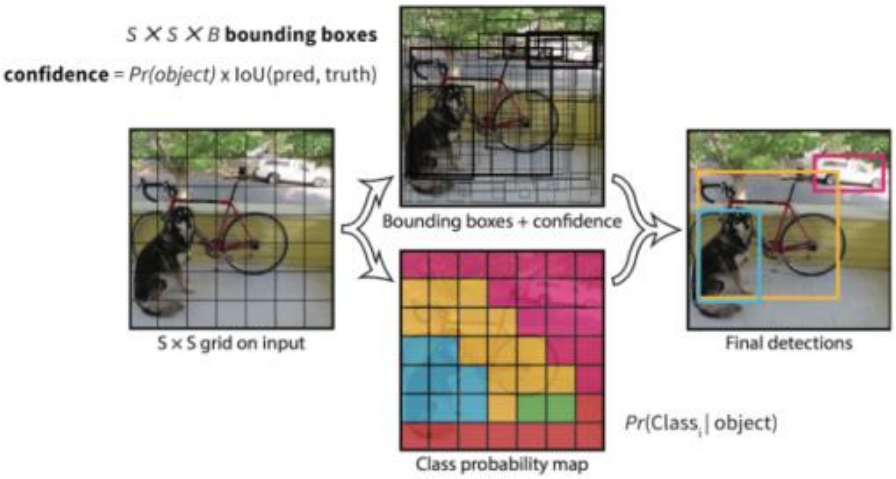

Figura 3.14: Esquema de detección por *Bounding boxes*. Fuente: *damavis.com*

Estas regiones de alta probabilidad presentarán muchas *Bounding boxes* solapadas, de manera que se filtrará cual es la región de mayor interés mediate la métrica *Intersection over Union* (IoU). Esta métrica calcula la relación que existe entre las áreas de 2 regiones y el área de unión de estas, de modo que un IoU alto indica que las celdas están muy solapadas entre ellas, indicando la localización más probable del objeto. Por último, se utiliza un filtro de *Non-maximum Suppression* (NMS) que elimina las redundancias y deja solo la celda que representa la detección más precisa.

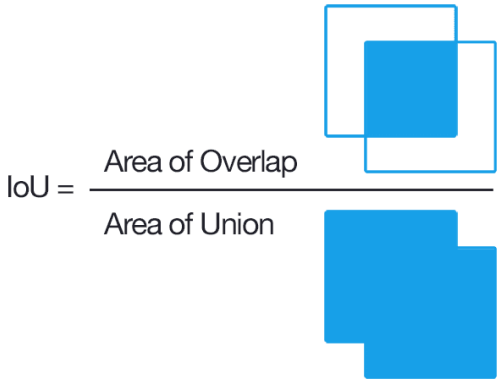

Figura 3.15: Fórmula y explicación gráfica de *Intersection Over Union*.
Fuente: *pyimagesearch.com*

Figura 3.16: Funcionamiento del NMS. Fuente: *damavis.com*

3.2.6 YOLO Data Augmentation

Data Augmentation hace referencia a un conjunto de técnicas que se aplican a los datasets usados en entrenamiento de modelos de AI para aumentar el tamaño y la diversidad del dataset sin tener que recopilar realmente más datos. En el caso de la visión por computador, este aumento se realiza aplicando técnicas de edición a las imágenes del dataset. Con el uso de estas técnicas se consigue crear nuevas imágenes a partir de las ya existentes mediante rotaciones, cambios de escala, cambios de luminosidad o contraste, recortes …Así, este proceso no solo permite aumentar el tamaño del dataset, sino que ayuda a mejor la diversidad mejorando los resultados del modelo, pues al tener imágenes más diversas en orientación, contraste o color, permite que el modelo puede inferir mejores conceptos de más alto nivel.

En el caso de YOLO, no es necesario realizar una data augmentation de manera aislada, pues el mismo modelo la realiza a partir del dataset que se le da como entrada [28]. Algunos de las técnicas que usa YOLO para realizar el aumento son las siguientes.

- Rotación: girar la imagen en un ángulo determinado.
- Escala: aumentar o reducir el tamaño de la imagen.
- Traslación: mover la imagen en una dirección específica.
- Volteo horizontal o vertical: reflejar la imagen horizontal o verticalmente.
- Recorte aleatorio: seleccionar una región de la imagen para ser recortada.

- Cambio de brillo, contraste o saturación: ajustar el brillo, contraste o saturación de la imagen.
- Añadir ruido: agregar ruido a la imagen para hacerla más robusta.
- Distorsión elástica: aplicar deformaciones elásticas a la imagen para aumentar la variabilidad de los datos.
- Añadir sombras o cambios de iluminación: simular cambios en la iluminación de la imagen.
- Añadir objetos o elementos de fondo: agregar objetos o elementos de fondo a la imagen para hacerla más compleja

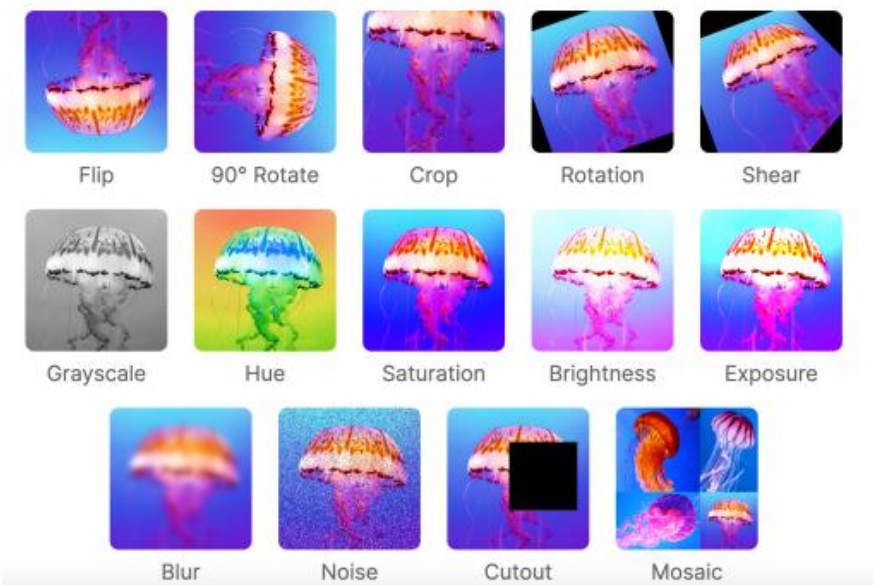

Figura 3.17: Ejemplo de data augmentation. Fuente: *roboflow.com*

3.2.7 Métricas

A la hora de hablar sobre YOLO y los algoritmos de detección es preciso desarrollar que métricas se van a tomar para expresar y cuantificar el rendimiento del modelo.

~~En el ámbito de la Inteligencia artificial,~~ es necesario tener en cuenta los siguientes conceptos básicos sobre cuantificación del rendimiento de una clasificación para poder construir métricas más complejas. El primero de ellos es Verdadero Positivo o *True Positive* (TP), que expresa que cuantas predicciones etiquetadas como verdadero, son de hecho verdadero. Junto con este el Falso Positivo o *False Positive* (FP) que expresa cuantas predicciones etiquetadas como positivas, no lo eran en realidad. La suma de estas dos métricas compone el total de positivos predichos por la red. Análogamente, Verdadero Negativo o *True Negative* (TN) se constituye con todas las predicciones etiquetadas como negativas que eran correctas, y el Falso Negativo o *False Negative* (FN) las que, etiquetadas como negativas, eran positivas en realidad. Al unir todas estas métricas en un matriz, surge la Matriz de confusión, que expresa como de acertada ha sido la clasificación del modelo.

De estas métricas simples, surgen dos más que resultan muy útiles, Precisión o *Precision* y Sensibilidad o *Recall*. La precisión expresa la relación entre los positivos predichos de manera correcta y el total de positivos predichos. Por otro lado, la Sensibilidad expresa la relación que existe entre los positivos predichos de manera correcta y el total de positivos que existían realmente. De este modo se puede determinar que el clasificar perfecto tendría tanto la Precisión como la Sensibilidad iguales a 1, ya tanto los FP como los FN serían cero.

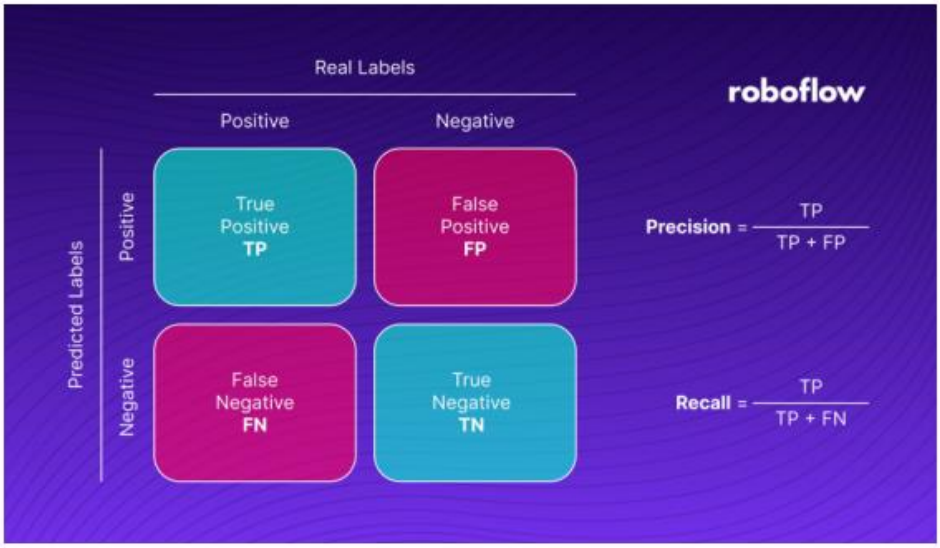

Figura 3.18: Matriz de confusión, Sensibilidad y Precisión. Fuente: *roboflow.com*

Además de la matriz de confusión, en el caso de YOLO se pueden evaluar la precisión y la sensibilidad mediante una curva de Precisión-Recall (PR). La curva PR permite visualizar cómo cambian estas dos métricas a medida que ajustamos el umbral de confianza que determina cuándo una predicción se considera positiva. Un modelo ideal tendría una alta precisión y sensibilidad simultáneamente, pero en la práctica, a medida que aumentamos el umbral de confianza, la precisión suele aumentar mientras que la sensibilidad disminuye, y viceversa. A través de esta curva, se puede identificar el umbral de confianza óptimo que balancea la precisión y la sensibilidad de manera efectiva, maximizando el desempeño del modelo en la detección de objetos.

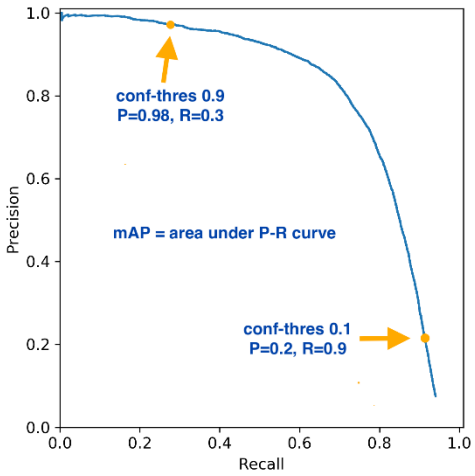

Figura 3.19: Ejemplo de Curva PR. Fuente: *github.com*

Otra métrica a considerar es mAP (Mean Average Precision) que expresa la relación entre precisión y sensibilidad para distintos niveles de confianza en las predicciones del modelo.En el caso de YOLO, se utilizan dos variantes principales de mAP: mAP@50 y mAP@50-95. La mAP@50 mide la precisión para un valor de IoU (Intersección sobre Unión) de 0.5. Un valor de IoU de 0.5 significa que al menos el 50% de la caja delimitadora predicha se superpone con la caja delimitadora real. Por otro lado, la mAP@50-95 calcula la precisión media para un rango de valores de IoU, desde 0.5 hasta 0.95, considerando distintos umbrales de superposición entre las cajas delimitadoras predichas y las reales. Esto proporciona una evaluación más exhaustiva del desempeño del modelo en diferentes niveles de precisión y permite una comprensión más completa de su capacidad para detectar objetos con precisión variable.

$$AP = \int_0^1 p(r)dr, \qquad p(r) \equiv Precision - Recall; \qquad mAP = \frac{1}{N}\sum_1^N AP_N, \qquad N \equiv Número\ de\ clases$$

Ecuación 3.4: Mean Average Precision

3.3 Creación del dataset

En el desarrollo de sistemas de Inteligencia Artificial, el primer proceso que se debe realizar es la creación de un conjunto de datos de calidad sobre el que realizar el entrenamiento. Este conjunto de datos además de ser amplio tiene que estar confeccionado para cumplir una serie de características que aseguren que los datos son lo suficientemente buenos como para ser usados como entrada para un proceso de entrenamiento [29].

3.3.1 Características de un buen dataset

Para conseguir que un dataset se pueda considerar como adecuado para realizar un entrenamiento, debe cumplir con los siguientes requisitos.

1. Representatividad de los datos. El dataset debe contener una variedad de ejemplos que reflejen todas las posibles combinaciones de inputs que le llegarán al modelo en la situación real. En este caso en particular, debe contener imágenes de una gran cantidad de basura distinta como botellas, latas o bolsas en distintos entornos acuáticos como ríos o mares para que el algoritmo sea capaz de generalizar en todos los contextos.

2. Calidad de los datos. Los datos incompletos, incorrectos o ruidosos pueden afectar negativamente el rendimiento del modelo. Por lo tanto, es crucial realizar un preprocesamiento exhaustivo para limpiar y preparar los datos antes de utilizarlos para el entrenamiento. En este caso, es necesario que las imágenes sean de la suficiente calidad y evitar imágenes poco relevantes que introducen ruido como pueden ser fotos de basura una la calle o en una piscina.

3. Tamaño del dataset. En general, un dataset más grande proporciona al modelo más ejemplos para aprender patrones y generalizar mejor a nuevas instancias [30]. En esta implementación, se usará un proceso de fusión de distintos datasets relevantes para asegurar un tamaño suficiente en el conjunto de datos.

3.3.2 Formato de los datos

Cuando se habla de datos para realizar un entrenamiento de una red neuronal, tan importante es la obtención de un dataset de calidad como la forma y el etiquetado de estos. En este caso de, los datos vendrán en forma de imagen con formato RGB (Red, Green and Blue) de tamaño 1280x1280, aunque es posible implementar otro tipo de tamaños de imagen también cuadrados como 720x720 o 640x640.En cuanto al tipo de etiquetado, YOLO admite varios formatos para esta tarea.

3.3.2.1 YOLO darknet

El formato que usaremos en esta implementación y el más utilizado el formato de YOLO darknet [31]. Este estándar de etiquetas consta un archivo en formato TXT donde se recoge la clase a la que pertenece el objeto detectado, las coordenadas x-y del centro del rectángulo que contiene al objeto y su ancho y altura. Este archivo debe tener el mismo nombre que la imagen a la que etiqueta para que sea reconocido a la hora del entrenamiento.

Figura 3.20: Imagen 000013.jpg del dataset de entrenamiento

class_id	center_x	center_y	width	height
0	0.478125	0.803125	0.1125	0.128125
0	0.352734	0.927343	0.1148	0.1453125
0	0.7765625	0.501562	0.04218	0.03125

Figura 3.21: Etiqueta 000013.txt del dataset de entrenamiento

3.3.3 Proceso de fusión de datasets

Para cumplir con los estándares de calidad y tamaño del dataset que se usará para reentrenar YOLO, es necesario llevar a cabo un proceso de fusión de varios conjuntos de datos distintos. Al no contar suficientes imágenes tomadas desde el ASV, el proceso de creación del dataset se basará en recolectar dataset relevantes (que contengan imágenes de basura en la superficie del agua) y fusionarlos en uno solo que será el que usé para el reentrenamiento definitivo.

El proceso comenzará al encontrar un dataset de imágenes de basura, posteriormente se analiza si cumple con los criterios de representatividad y calidad de manera que las imágenes sean nítidas y estén tomadas desde una perspectiva similar a la que habrá en ASV, no valen tomadas desde UAV, por ejemplo, y que además estén tomadas en ríos, lagos u océanos, no sería válidas las tomadas en piscinas o en las calles. En caso de no cumplir estos criterios se descartaría el dataset y se comenzaría la búsqueda de un nuevo dataset.

Una vez asegurada la calidad del dataset, se fusionará con los datasets de iteraciones anteriores para crear un solo dataset que combine ambos. Posteriormente, se procederá a la división del dataset en conjuntos de entrenamiento y de prueba. Este proceso implica separar aleatoriamente el dataset en un 80% para entrenamiento y un 20% para prueba. El conjunto de entrenamiento se utilizará para entrenar el modelo, mientras que el conjunto de prueba se mantendrá aparte y se utilizará para evaluar el rendimiento del modelo entrenado. Con el objetivo de hacer estos entrenamientos de prueba lo más rápidos posible, se reducirá al máximo el número de veces que durante el entrenamiento se da una pasada completa al dataset (llamadas estas pasadas épocas de entrenamiento).

En caso de que los resultados sean mejores que los previos a la inclusión del dataset, se debe determinar si son lo suficientemente buenos como para ejecutar un entrenamiento definitivo que va a consumir muchos recursos computacionales y tiempo. Es caso afirmativo, se realizará una búsqueda de hiperparámetros para mejorar aún más el desempeño del modelo y finalmente se realizará un entrenamiento con el dataset completo usando todos los recursos computacionales y de tiempo que se necesiten.

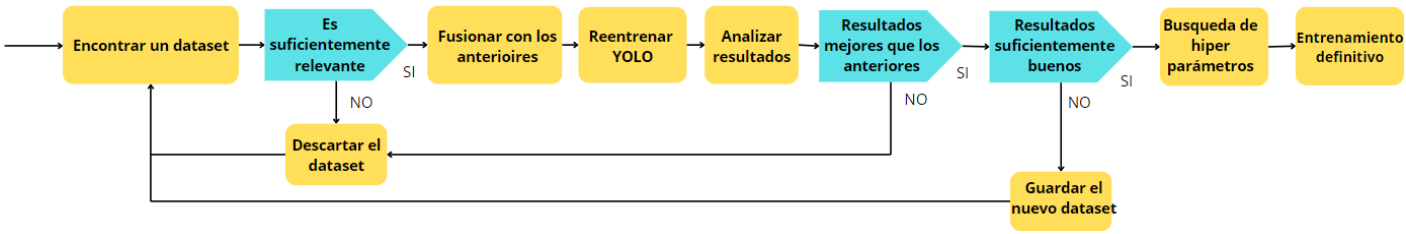

Figura 3.22: Esquema del proceso de selección de dataset y reentrenamiento de YOLO

3.4 Hardware

Una vez abordado las herramientas y la metodología que se usarán para desarrollar el software del ASV, en conveniente desarrollar de manera breve el hardware donde va a ejecutarse el algoritmo de YOLO ya que es vital conocer las limitaciones tecnológicas y de cómputo cuando se trata con un proyecto que ejecuta algoritmos de Inteligencia Artificial. Así, se presentarán de forma breve tanto la cámara usada como la plataforma de cómputo.

Por un lado, la cámara Zed 2i, es un sistema de cámara estéreo que mejora la consciencia espacial de la detección gracias su amplio ángulo de detección de 120º. Además, también da la posibilidad de calcular la distancia a los objetos por estereoscopia usando dos imágenes RGB. Este sistema también destaca por ofrecer la posibilidad de acceder a bajo nivel a los datos proporcionados con la cámara, lo que junto con su comunicación vía USB 3.1 permite una comunicación cómoda y eficiente con la plataforma de cómputo. Adicionalmente, la ZED 2i integra funcionalidades avanzadas como la detección y seguimiento de objetos en 3D en tiempo real, permitiendo monitorizar el movimiento de estos en el espacio. También es capaz de realizar reconstrucciones 3D detalladas del entorno, lo que la hace ideal para aplicaciones en robótica, realidad aumentada y virtual, y mapeo espacial. [32].

Por otro lado, la plataforma usada Jetson Xavier NX, destaca por ser una plataforma que permite el procesamiento conjunto de varias medidas proporcionadas por sensores como cámaras o sónares. Gracias a su arquitectura acelerada por GPU, esta plataforma es capaz de hacer tareas computacionalmente costosas en tiempo real como la utilización de técnicas de Inteligencia Artificial o planificación de rutas [33].

Figura 3.23: Zed 2i. Fuente: *stereolabs.com*

Figura 3.24: Jetson Xavier NX. Fuente: *nvidia.com*

4 RESULTADOS

Una vez explorada la metodología y las herramientas que se han usado para realizar este desarrollo, abordaremos en este capítulo los resultados del proceso de fusión de datasets y reentrenamiento de YOLOv5.Primero se hará un breve desarrollo de buenas prácticas que se recomiendan a la hora de reentrenar YOLO con un dataset específica para luego ir abordando de manera cronológica los resultados del proceso de fusión y reentrenamiento, para finalmente analizar ~~la~~ cual es el desempeño real del modelo al someterlo a pruebas realizadas por el ASV.

4.1 Kit de herramientas utilizadas

Antes de empezar a desarrollar como se ha ejecutado el proceso de reentrenamiento del modelo, es preciso mencionar y explicar las herramientas utilizadas durante el desarrollo del proyecto.Estas herramientas se pueden agrupar según el propósito de su uso en etiquetado de imágenes, entrenamiento del modelo y visualización de resultado.

En cuanto al etiquetado de imágenes, se ha utilizado la herramienta online CVAT [34], que permite la utilización de un software que mediante el dibujo de la *Bounding Boxes* con el ratón genera las etiquetas automáticamente con el formato especificado, en este caso Darknet. Además, se han usado programas propios para normalizar el formato de las etiquetas, así como para organizar las imágenes en los distintos directorios separando los datasets en entrenamiento y validación mediante el uso de scit-learn [35]. En cuanto al entrenamiento del modelo, se ha hecho uso del repositorio de github de ultralytics [36] para facilitar el proceso de entrenamiento de YOLO. Gracias a este repositorio, se puede llevar a cabo todo el proceso de entrenamiento únicamente mediante comandos de terminal (bash), aunque internamente los ejecutables están desarrollados en Python usando el framework especializado en Deep Learning PyToch [37]. Además de esto, se ha hecho uso de GoogleColab [38] para ejecutar los reentrenamientos de prueba gracias a que permite usar recursos computacionales en la nube que permiten reentrenar YOLO mucho más rápido que el un computador normal. ~~Por último, la visualización de resultados~~ se hecho ~~mediante una serie de programas propios que se adjuntarán como anexos.~~ Estos programas escritos en Python están dedicados a crear visualizaciones de los resultados de los reentrenamientos mediante librerías como Matplotlib [39] o Seaborn[40] además de otras como OpenCV [41] que se han usado para visualizar las imágenes con sus *Bounding Boxes*.

4.2 Recomendaciones para reentrenar YOLO

Para realizar este proceso de reentrenamiento y fusión de datos, se han seguido una serie de recomendaciones suministradas por ultralytics para asegurar que se alcanzan los mejores resultados posibles al entrenar con un dataset personalizado [42]. Las recomendaciones seguidas han sido las siguientes.

- Aportar una cantidad de imágenes por clase superior a 1500.En este caso al tener sólo una clase genérica de basura, cualquier imagen con basura es válida.

- Generar un conjunto de imágenes representativo, refriéndose a que hayan sido tomadas en un entorno similar de ejecución, con diferentes iluminaciones, diferente ángulo … etc. En este caso se consigue gracias en parte a fusionar varios datasets.

- Asegurar que las etiquetas están generadas de manera lo más precisa posible.

- Agregar imágenes de fondo que no contengan ningún objeto para reducir los Falsos Positivos al no "acostumbrar" al modelo a buscar un objeto en todas las imágenes. Se recomienda que sea entre un 0-10% del total del dataset.

- Tratar de entrenar 300 épocas como objetivo por defecto para ver el efecto del *overfitting*.

- En caso de presencia de objetos de tamaño reducido, entrenar y probar el modelo en resoluciones altas. En este caso se entrenará a 720 para las pruebas y a 1280 para el modelo final.

- Usar un *batch size* suficientemente grande a la hora de entrenar para evitar estadísticas deficientes al aplicar *batch normalization* [43].

- Usar los hiper parámetros por defecto que trae YOLOv5 para las pruebas y sólo al realizar el entrenamiento definitivo ejecutar previamente un proceso de evolución de hiper parámetros.

4.3 Análisis del dataset 1: FloW-Img

El primer dataset que se tratará será FloW-Img generado por la empresa ORCA-Uboat [44], que consiste en un conjunto de 2000 imágenes y 200 vídeos tomados desde un ASV de residuos en la superficie acuática con un total de 5271 objetos etiquetados.

El dataset se divide en 1200 imágenes para entrenamiento y 800 para validación. Estas imágenes están tomadas con una resolución tanto de 1280x720 como de 1280x640. Para este proyecto usaremos los videos como datos de test para probar el desempeño del modelo en un video simulando las condiciones reales de funcionamiento. En cuanto al etiquetado, está en formato PASCAL, por lo que es necesario transcribirlo al formato Darknet, ~~como se hace en el anexo I.~~

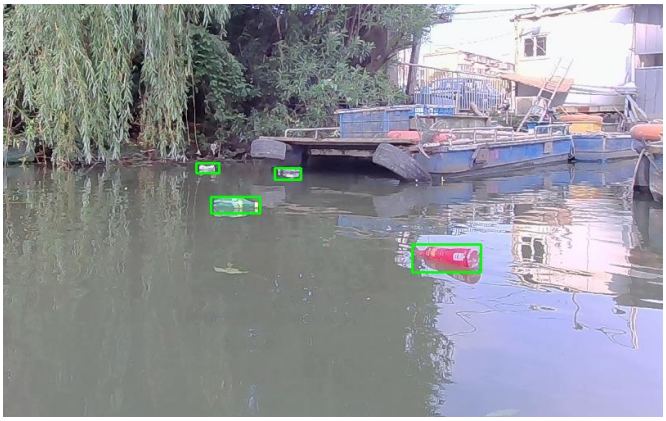

Figura 4.1: Imagen de FloW-Img etiquetada

Analizando el dataset al completo es importante resaltar que la mayoría de las imágenes contienen entre 2 y 3 piezas distintas de basura y que no existen imágenes sin basura. Como se expuso en las recomendaciones reentrenar YOLO, esto puede provocar una gran cantidad de Falsos Positivos. De modo que se espera que en las pruebas de validación su rendimiento sea correcto pues el subconjunto de validación tampoco posee imágenes de fondo, pero al ejecutar el test en los videos es esperable que existan FP cuando no hay basura presente.

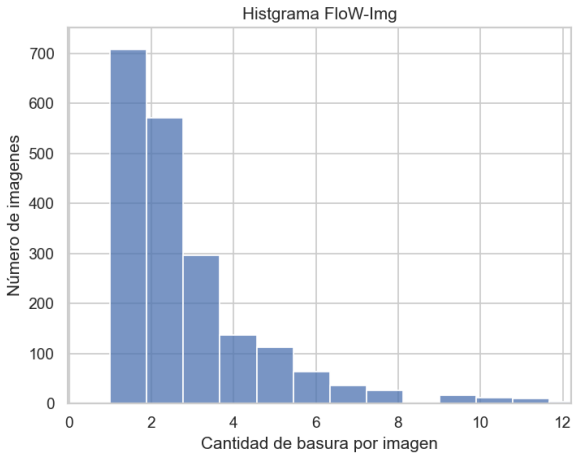

Figura 4.2: Histograma de distribución de basura del dataset 1

4.4 Reentrenamiento 1

Una vez analizado el primer dataset, se procederá a un entrenamiento de prueba para validar que los resultados que ofrece YOLO ante la detección de basura en recursos hídrico son los esperados.

Este entrenamiento al ser de prueba se realizará sobre un número de épocas reducido y a una resolución menor de la que se usará en ASV ya que se cuenta con recursos limitados de cómputo y tiempo. Así, este primer entrenamiento se realizará con una resolución de 640x640 y se entrenará 25 épocas.

Reentrenamiento	Resolución	Épocas	N.º de imágenes de entrenamiento	N.º de imágenes de validación
1	640x640	25	1800	200

Tabla 4.1: Características reentrenamiento 1

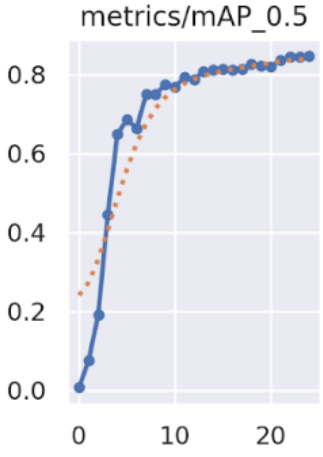

Figura 4.3: Evolución de mAP@0.5 en función de las épocas

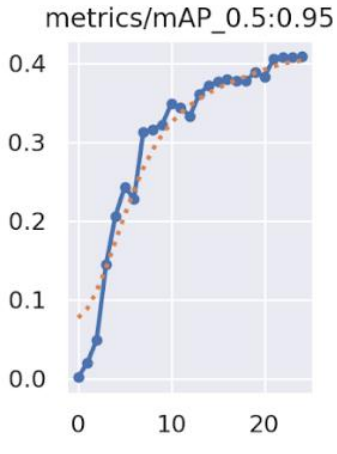

Figura 4.4: Evolución de mAP@0.5:0.95 en función de las épocas

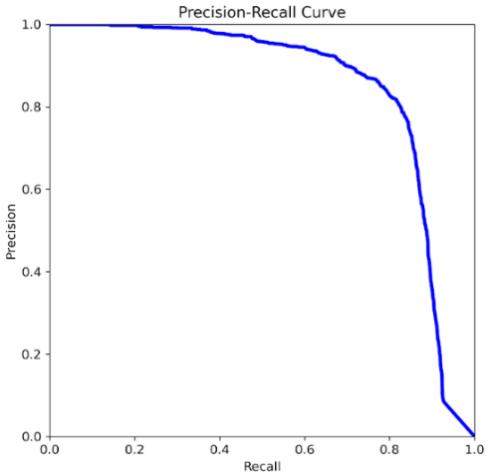

Figura 4.5: Curva de Precisión contra Sensibilidad

Reentrenamiento	mAP@0.5	mAP@0.5:0.95	Precision	Recall
1	0,84701	0.40883	0,8392	0,7933

Tabla 4.2: Métricas reentrenamiento 1

A partir de los resultados mostrados siendo el mejor mAP@50 obtenido y la combinación P-R que mejor rendimiento proporciona (Coordenadas del vértice de la curva P-R), se puede observar que la adaptación de YOLO a un dataset de basura flotando en entornos acuáticos es exitosa.

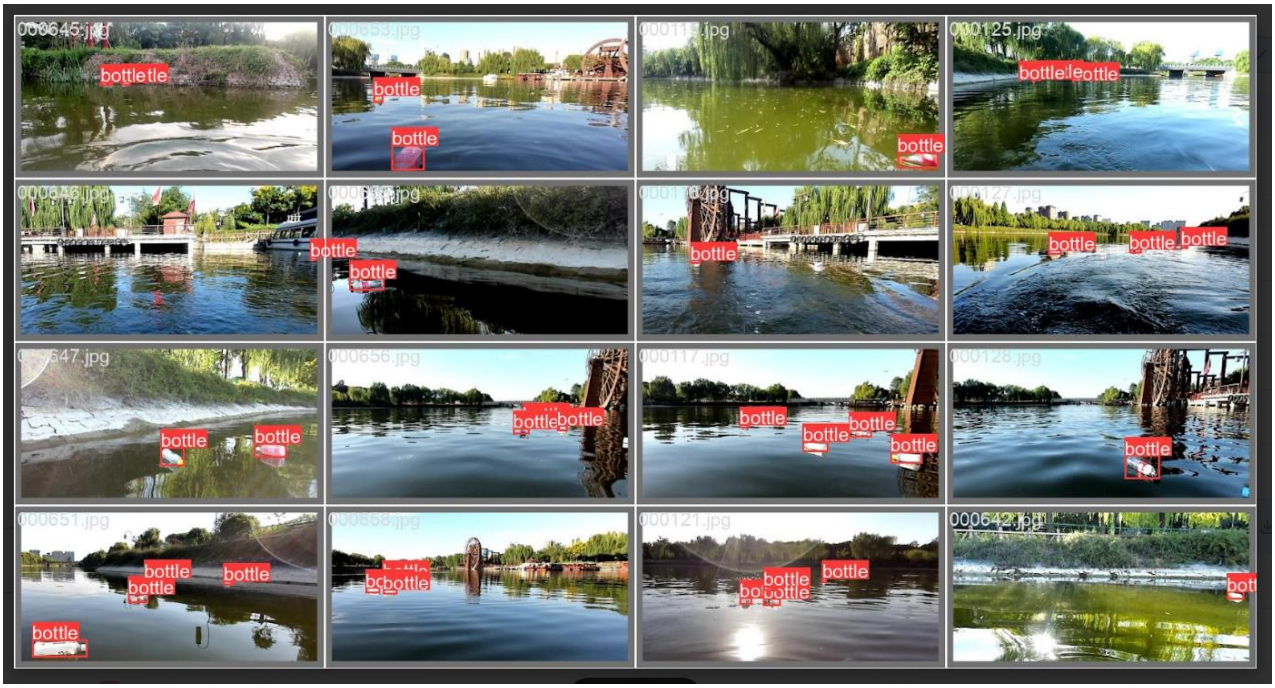

Figura 4.6: Conjunto de imágenes para validación etiquetadas entrenamiento 1

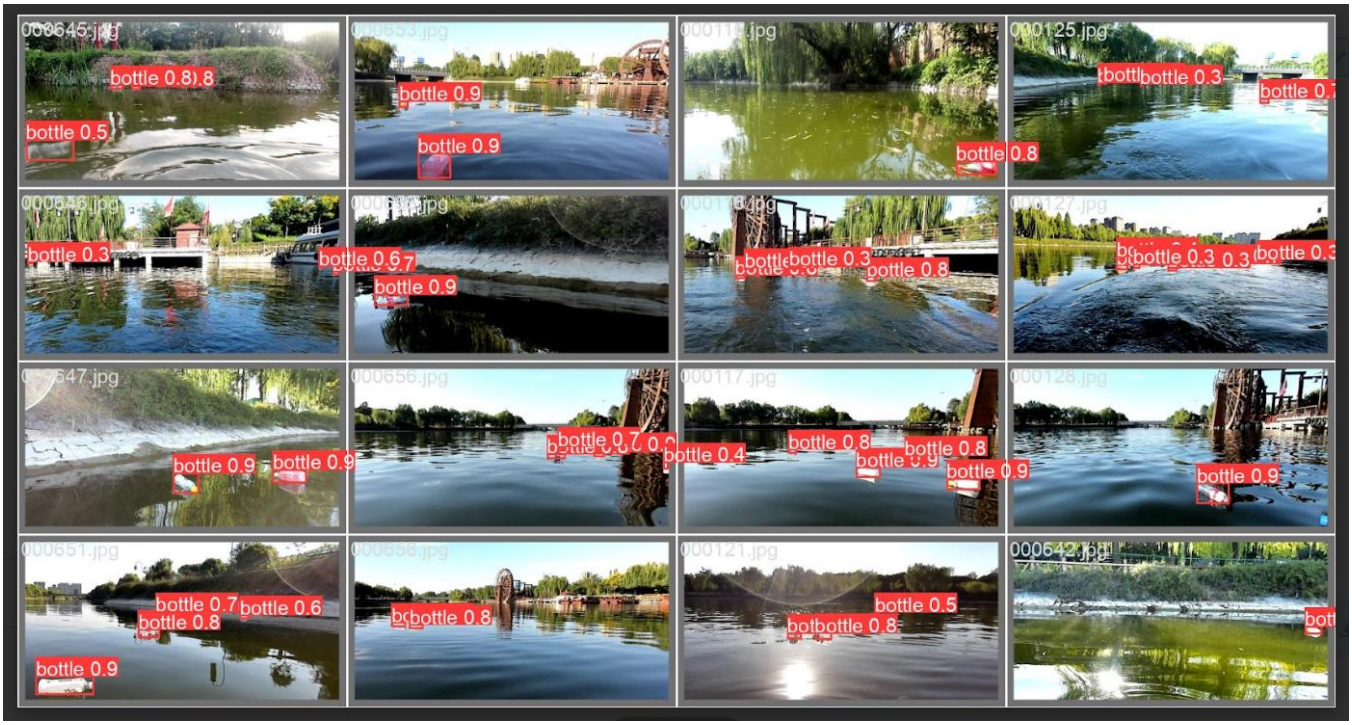

Figura 4.7: Resultados de validación entrenamiento 1

Figura 4.8: Extracto de video de test con falso positivo

Como se esperaba, las pruebas ejecutadas sobre el subconjunto de validación muestran un desempeño bastante positivo, pero al intentar detectar en un video real donde no siempre hay basura presente, se detectan falsos positivos de manera continua. Además, se puede apreciar como también detecta falsos positivos en algunos reflejos existentes en el agua (Figura 4.5 esquina superior izquierda).

En resumen, el primer ~~re~~entrenamiento finaliza con un resultado positivo. Sin embargo, presenta problemas de falsos positivos que se tratarán de solucionar en futuras iteración mediante la ampliación del dataset de entrenamiento incluyendo más imágenes en general para solucionar el problema de los reflejos e imágenes sin basura para mejorar el desempeño en videos reales.

4.5 Análisis del dataset 2 y fusión

El segundo dataset que se va a tratar de incluir en el entrenamiento será floating_litter [45]. Este dataset este compuesto por imágenes tomadas desde un ASV de basura y otros objetos flotando en la superficie del agua.

Este dataset lo componen 691 imágenes con una resolución de 640 x 320. En cuanto al etiquetado, el dataset está pensado para tareas de segmentación contando con 3 etiquetas posibles *trash*, *water* y *static*, haciendo referencia a la basura, el agua y el fondo. De estas etiquetas se usará únicamente la de *trash* para realizar el entrenamiento. Cabe destacar también que es posible descargar este dataset en formato Darknet entre otros, por lo que no es necesario realizar una traducción desde otro formato como sí ocurrió con el dataset anterior.

Figura 4.9: Imagen de floating_litter etiquetada

Figura 4.10: Imagen sin basura de floating_litter

Analizando el dataset se observa que existen una cantidad significativa de imágenes ausentes de basura y que la mayoría de las imágenes que lo componen tienen entre 0 y 3 piezas de basura. Estas características hacen que sea apto para fusionarlo con el dataset anterior, pues introduce imágenes de fondo y la frecuencia de basura por imagen es similar en ambos casos.

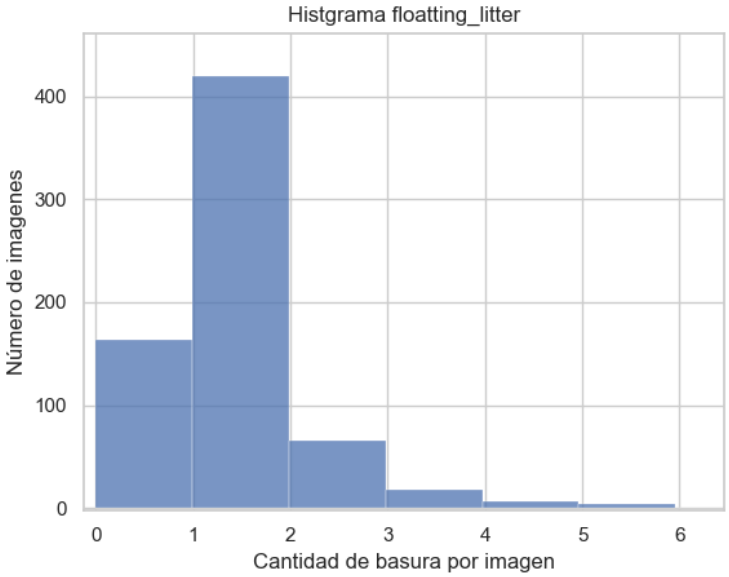

Figura 4.11: Histograma de distribución de basura del dataset 2

Una vez analizado el dataset floating_litter y habiendo determinado que es apto para fusionarlo con el anterior, es momento de realizar la unión. Este dataset formado de la fusión ~~lo que~~ se llamará Dataset 1_2 y estará compuesto por un total de 2691 imágenes procedentes de los datasets usados anteriormente. Destacar también que existen 164 imágenes de fondo constituyendo un 6% del total, situándose en el rango recomendado anteriormente. Con el reentrenamiento sobre este dataset se espera una mejoría en los resultados de validación ~~pues al tener más imágenes se podrá reentrenar durante más épocas~~ y una mejora significativa en la detección de falsos positivos cuando no haya basura presente.

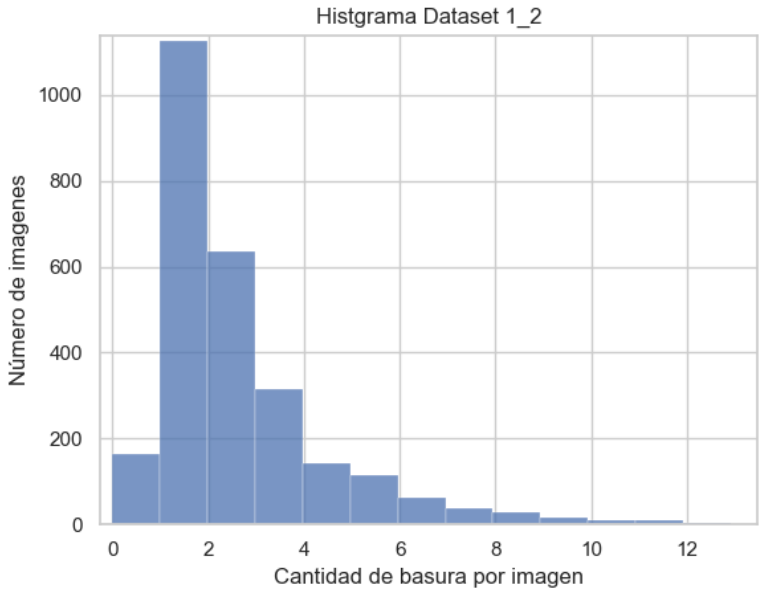

Figura 4.12: Histograma de distribución de basura en el dataset 1_2

4.6 Reentrenamiento 2

Ya con los datasets fusionados de manera satisfactoria, ~~es momento~~ de proceder con el segundo reentrenamiento de YOLO y analizar los resultados que arroja.

Este segundo entrenamiento se realizará sobre un total de 2152 imágenes para entrenamiento y 539 para validación un total de 30 épocas pues al haber aumentado el dataset, en principio es posible entrenar más épocas sin riesgo de sobreajuste. Este reentrenamiento se realizará sobre el checkpoint yolov5s6, que será el definitivo y a una resolución de 720 x720 para ir aproximando la resolución a los 1280 x 1280 que se quiere, pero consumiendo menos recursos ya que no deja de ser una prueba. Mencionar también que siguiendo las recomendaciones proporcionadas previamente se usará un tamaño de lote de 32, evitando usar así los inconvenientes de usar *batch-size* pequeños.

Reentrenamiento	Resolución	Épocas	N.º de imágenes de entrenamiento	N.º de imágenes de validación
1	640x640	25	1800	200
2	720x720	30	2152	539

Tabla 4.3: Características reentrenamiento 2

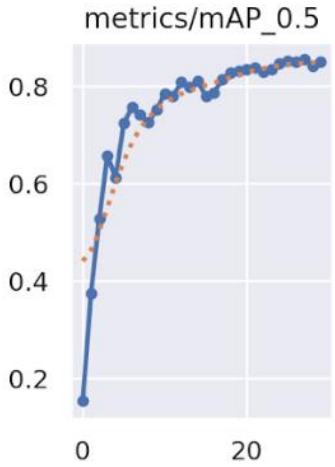

Figura 4.13: Evolución mAP@0.5 en función de las épocas

Figura 4.14: Evolución mAP@0.5:0.95 en función de las épocas

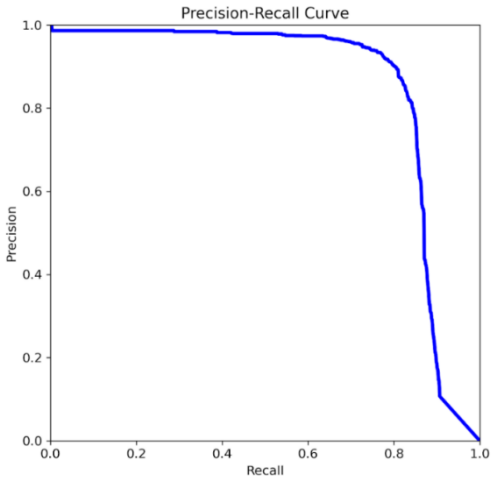

Figura 4.15: Curva Precisión contra Sensibilidad

Reentrenamiento	mAP@0.5	mAP@0.5:0.95	Precision	Recall
1	0,8470	0.4088	0,8392	0,7933
2	0,85115	0,48478	0,9017	0,8014

Tabla 4.4: Métricas entrenamiento 2

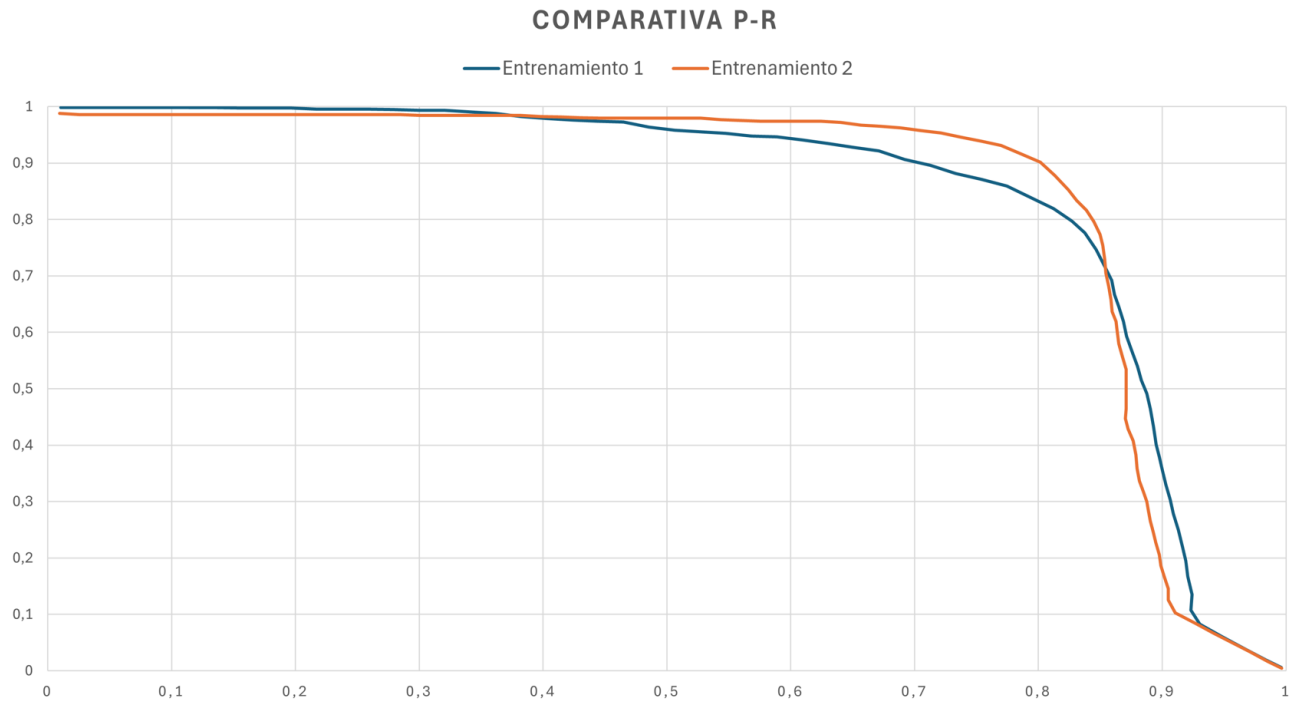

Figura 4.16: Comparativa P-R de los entrenamientos 1 y 2

Como se puede observar, con el segundo entrenamiento se obtiene una mejora de un 7% en la precisión y de un 18,5% en el mAP@0.5:0.95 indicando una reducción significativa en la detección de falsos positivos, pues la precisión mejora en mucha mayor proporción que la sensibilidad.

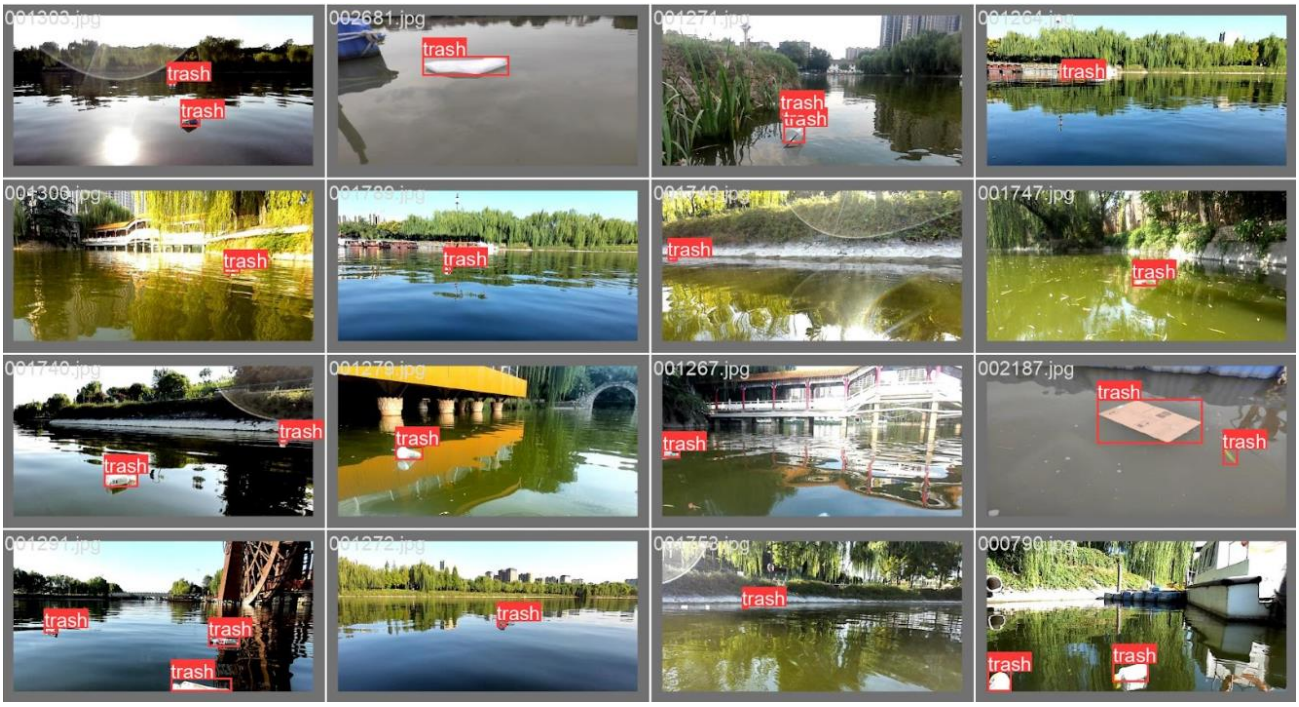

Figura 4.17: Subconjunto para validación entrenamiento 2

Figura 4.18: Resultados de validación entrenamiento 2

Figura 4.19: Extracto de video de test con falso positivo

Analizando el desempeño del modelo en los videos de test, podemos ver que, gracias a la mejora conseguida en el modelo en este segundo reentrenamiento, reconoce ambas botellas como basura, que en el caso anterior sólo reconocía la amarilla, y que, aunque existan falsos positivos en el disco del ASV donde se graba el video, esta vez son más esporádicos y con menor confianza. Esto indica que si se aplicara un filtrado de confianza alrededor del 0.5 se podrían eliminar la mayor parte de los falsos positivos.

4.7 Análisis del dataset 3 y fusión

Como último dataset se ha seleccionado mixed-env-raw del conjunto WCB5G [46]. Este dataset se compone de 6 datasets distintos donde se agrupan imágenes de residuos en distintos entornos acuáticos. Sin embargo, algunos de estos conjuntos están compuestos por imágenes tomadas en piscinas por lo que se han desechado por no ser suficientemente relevantes para el proyecto, pues la basura se encontrará en entornos naturales como ríos o lagos.

El dataset se compone de 844 imágenes de resolución 1280 x 1280. Este dataset permite la posibilidad de descargarlo el etiquetado en formato Darknet, siendo la única etiqueta es “trash”, por lo no será necesario realizar ningún cambio de formato.

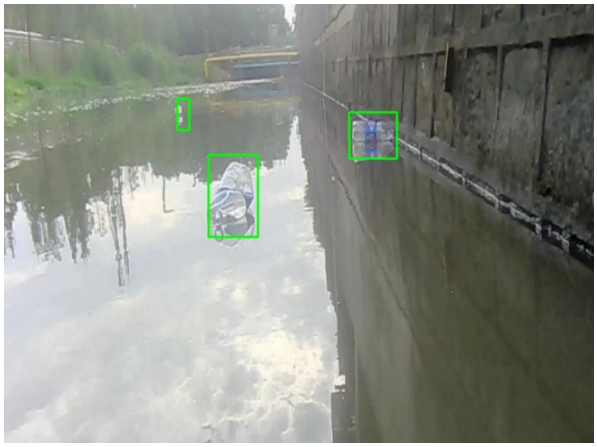

Figura 4.20: Imagen etiquetada de WCB5G

Al analizar este dataset podemos observar que sigue el patrón de los anteriores. Existen un 11% de imágenes de fondo y la gran mayoría de imágenes presenta entre 1 y 3 piezas de basura. Esto hace que este dataset sea muy adecuado para fusionarlo con los anteriores pues la distribución es similar.

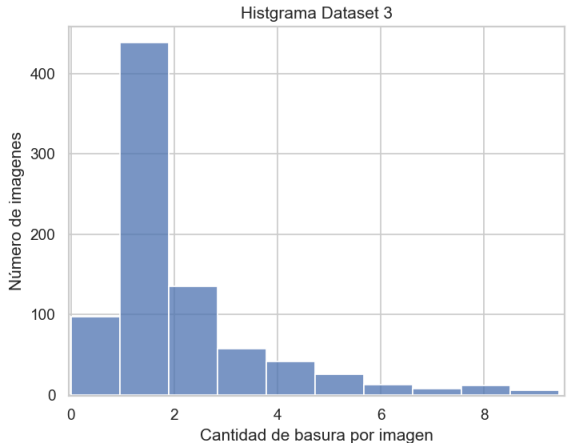

Figura 4.21: Histograma de distribución de basura dataset 3

Una vez determinado que el dataset 3 es apto para fusionarlo con los anteriores es momento de integrarlos todos. Este dataset_1_2_3, consta de 3535 imágenes con resoluciones variadas, 1280x1280 o 640x320. Además, se puede apreciar cómo, al igual que en casos anteriores, la mayor parte de las imágenes presentan entre 1 y 3 piezas de basura, presentando también un 7,3% de imágenes sin basura, cumpliendo así el criterio establecido anteriormente. Con el reentrenamiento de este dataset se espera una mejora en los parámetros como en caso anteriores y que gracias al aumento del 30% en la cantidad de imágenes, reduzcan los falsos positivos detectados.

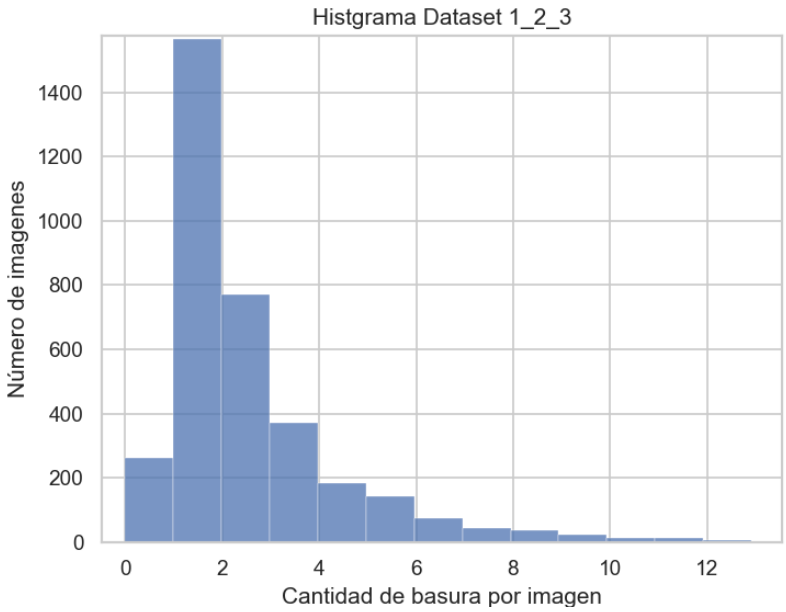

Figura 4.22: Histograma de distribución de basura por imagen del dataset_1_2_3

4.8 Reentrenamiento 3

Con el nuevo dataset se realiza el último del reentrenamiento de prueba, para comprobar que las métricas se mejoran como se espera.

Este reentrenamiento se realizará sobre un total de 2828 para entrenar y 707 de validación, con resolución de 720 x 720 como en el caso anterior para evitar el excesivo uso de recursos. Además, se aumentará el número de épocas 100 ya que la cantidad de imágenes ha aumentado de manera notable y permite entrenar al modelo en una cantidad mayor de épocas antes de llegar al sobreajuste del modelo. Además, este entrenamiento tan largo sirve a su vez para estimar la cantidad de épocas a entrenar en el entrenamiento definitivo.

Reentrenamiento	Resolución	Épocas	N.º de imágenes de entrenamiento	N.º de imágenes de validación
1	640x640	25	1800	200
2	720x720	30	2152	539
3	720x720	100	2828	707

Tabla 4.5: Características reentrenamiento 3

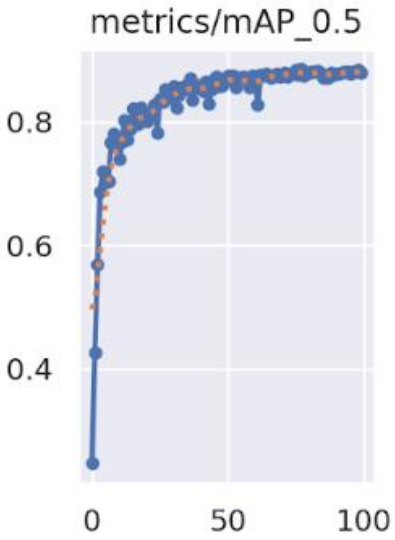

Figura 4.23: Evolución de mAP@0.5 en función de las épocas entramiento 3

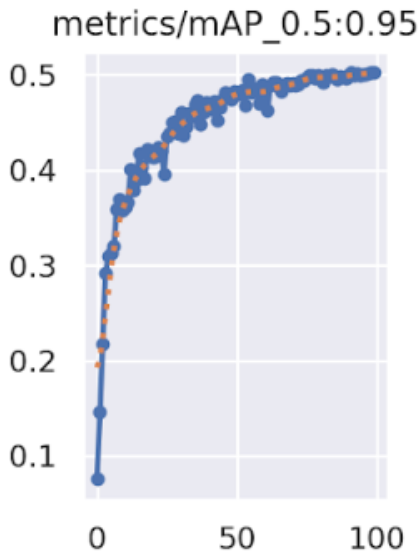

Figura 4.24: Evolución de mAP@0.5:0.95 en función de las épocas entrenamiento 3

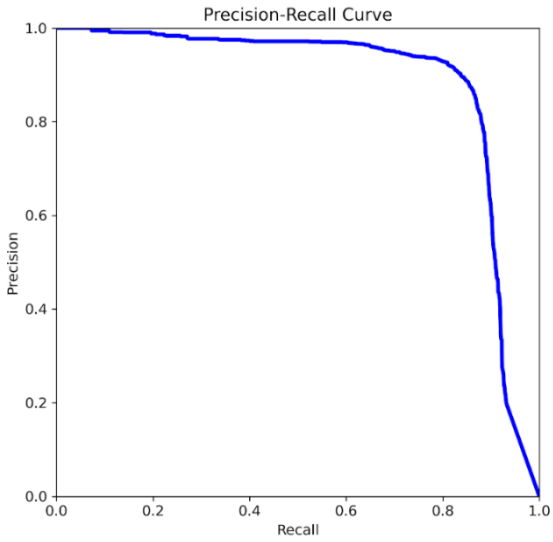

Figura 4.25: Curva Precisión contra Sensibilidad entrenamiento 3

Reentrenamiento	mAP@0.5	mAP@0.5:0.95	Precision	Recall
1	0,8470	0.4088	0,8392	0,7933
2	0,85115	0,48478	0,9017	0,8014
3	0.88615	0.50236	0.8931	0.8440

Tabla 4.6: Métrica reentrenamiento 3

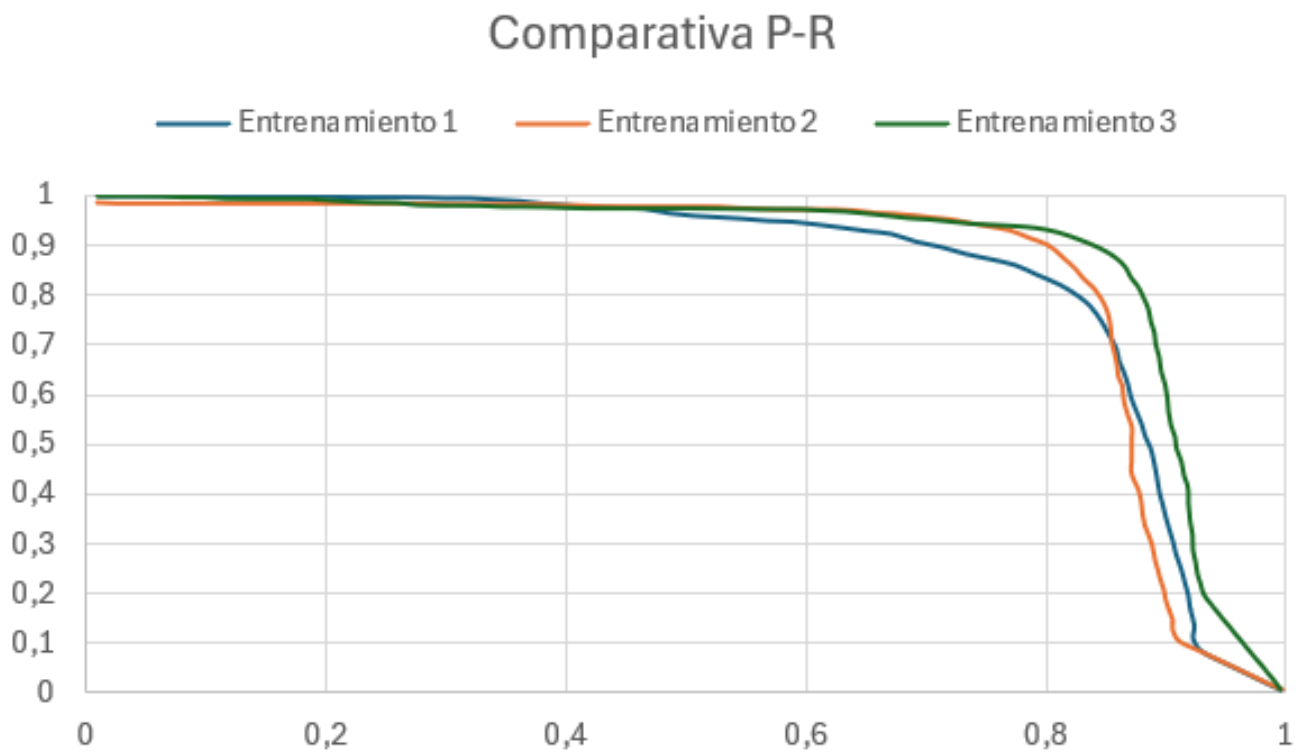

Figura 4.26: Comparativa P-R de los 3 entrenamientos

Analizando los resultados arrojados por el tercer entrenamiento, podemos ver que los resultados han mejorado notablemente en términos de sensibilidad. La Sensibilidad ha subido en un 5,3% mientras que la precisión ha mantenido el valor anterior. Esto resulta en una mejora en la detección, pues se detecta más basura que antes pasaba desapercibida y sin embargo la proporción entre detectadas y correctamente detectadas sigue constante.

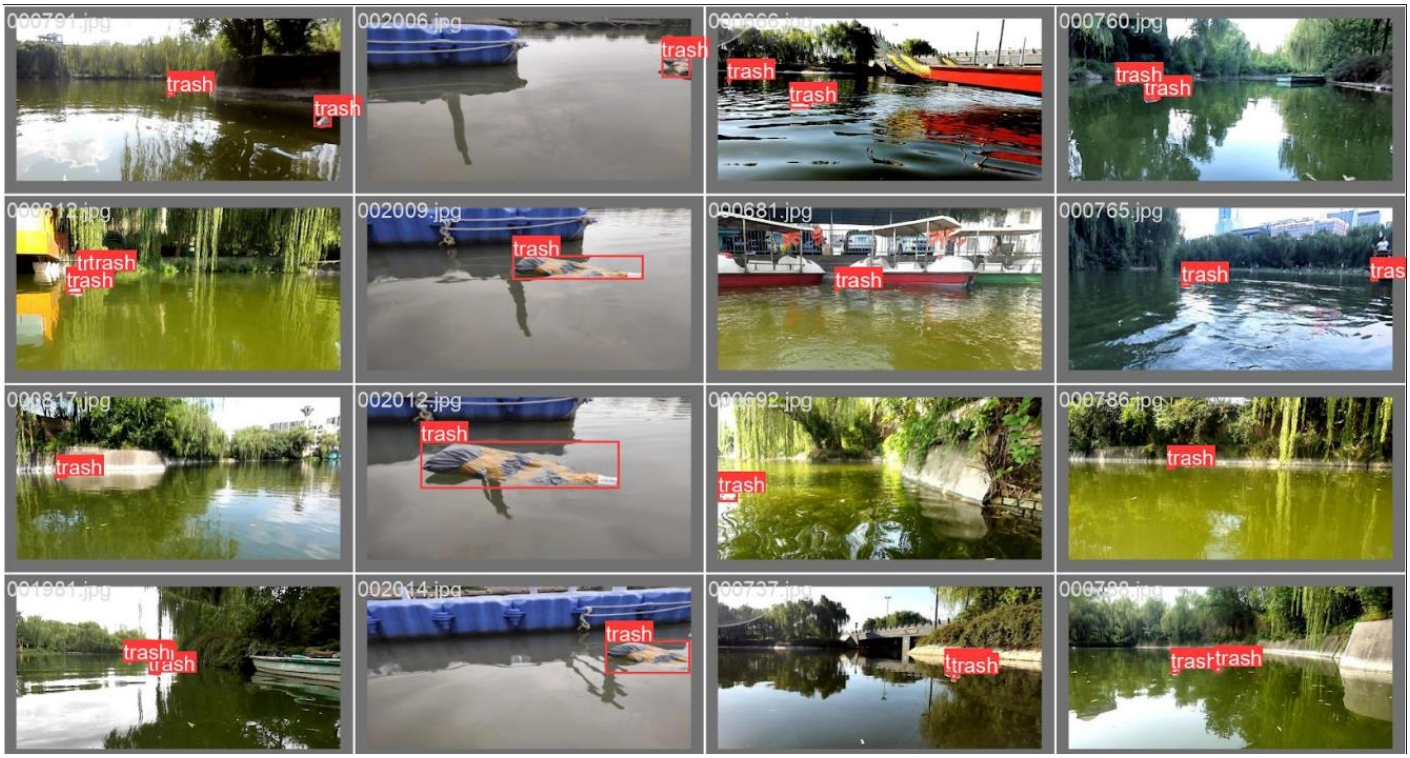

Figura 4.27: Subconjunto para validación entrenamiento 3

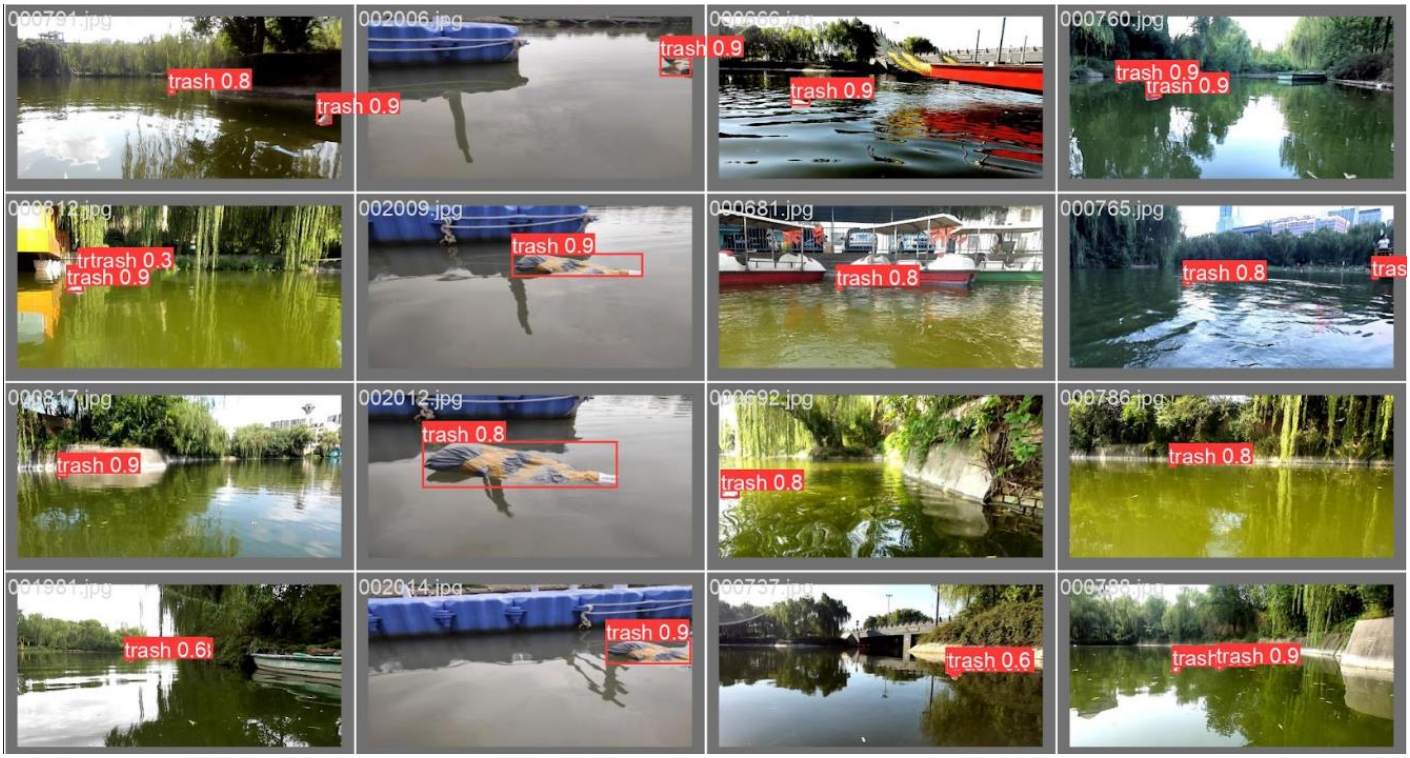

Figura 4.28: Resultado de la validación entrenamiento 3

Figura 4.29: Extracto de video de test con falso positivo entrenamiento 3

Como se puede observar en las pruebas, se continua con la dinámica de mejora vista anteriormente, se siguen detectando falsos positivos pero cada vez con menor confianza con respecto a las detecciones correctas. Destacar de en caso de ampliar la imagen o usar un ASV sin el disco que genera el falso positivo, el desempeño del algoritmo sería más que sobresaliente.

4.9 Exploración y mejora del dataset final

Una vez formado el dataset definitivo que se usará en el entrenamiento final, es pertinente realizar una exploración más profunda del dataset para perfeccionar lo más posible y conseguir aumentar su calidad lo más posible.

En esta exploración se han detectado 2 principales vectores de mejora, el primero de ellos es la eliminación de imágenes que presentan demasiadas piezas de basura ~~en pantalla~~ el segundo es el reetiquetado de imágenes que presentan fallos.

La eliminación de imágenes con valores atípicos o *outliers* es una práctica que pretende reducir el ruido del dataset y así mejorar el entrenamiento del modelo. Así, teniendo en cuanta la distribución de basura por imagen, se tomará como *outlier* las imágenes con más de 11 piezas de basura por imagen. Este criterio se establece para no limitar demasiado el rango de basura que puede detectar el algoritmo a la vez, entre 0 y 11, ~~ya que es poco probable que quepan más de 11 piezas de basura en una sola imagen.~~ Destacar también que las imágenes con cantidades de basura altas estaban en su mayoría mal etiquetadas o eran poco relevantes por lo que tendrían que ser eliminadas igualmente.

Figura 4.30: Ejemplo de outlier mal etiquetado

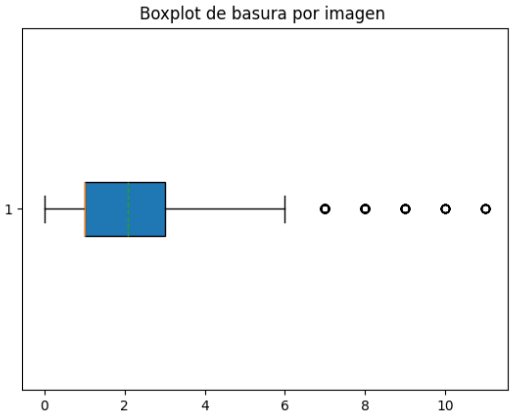

Figura 4.31: Boxplot sin outliers

N.º de piezas de basura	12	13	14	15	16	17	21	23	24	35	39	40	45	109	143	**Total**
N.º de imágenes	6	3	2	2	5	4	1	1	1	1	1	1	1	1	1	31

Tabla 4.7: Cantidad de outliers eliminados

Así, podemos concluir que se han eliminado un total de 31 imágenes pasando la cantidad total de imágenes de 3535 a 3504. Como se muestran estas imágenes eliminadas tenían errores en el etiquetado así que se ha mejorado la calidad del dataset por partida doble.

En cuanto al reetiquetado de imágenes, se han encontrado un total de 70 imágenes con errores de etiquetado incluso después de eliminar las que contenían errores y eran outliers. La mayoría de los errores presentes en estas imágenes se deben a que se ha realizado un proceso de data augmentation incluyendo rotación en las imágenes, pero la posición de la *bounding box* no se ha rotado correctamente.

Figura 4.32: Ejemplo de imagen mal etiquetada

Figura 4.33: Ejemplo de imagen reetiquetada correctamente

4.10 Búsqueda de hiper parámetros

Una vez construido el dataset final que se usará en el entrenamiento, es pertinente realizar una búsqueda los hiperparámetros óptimos. Según la documentación de YOLO mencionada anteriormente, no era necesario llevar a cabo esta búsqueda de primeras ya que el coste computacional es muy elevado y los parámetros que trae YOLO por defecto funcionan de manera muy satisfactoria de forma general. Debido a esto solo se ha ejecutado este paso, una vez probada la eficiencia del modelo.

Para realizar esta búsqueda se han realizado 300 evoluciones de los hiper parámetros ejecutando en cada una 10 época de entrenamiento. Así, el entrenamiento al completo ha constado de 3000 épocas de entrenamiento sobre el dataset ~~ya si~~ a 1280 x 1280 de resolución.

Después de concluir esta búsqueda, se han obtenido un conjunto de hiperparámetros que mejora el entrenamiento del modelo de manera notable. Así, por medio de cambiar los parámetros de serie por los hallados en este proceso se consigue aumentar la precisión casi en un 5% con el mismo dataset y las mismas épocas en de entrenamiento. Cabe destacar también el aumento en el esto de métricas de aproximadamente en 2%.

Métricas	Modelo por defecto	Modelo con hiper parámetros optimizados	Aumento (%)
Precision	0.823	0.870	4.7
Recall	0.819	0.833	1.4
mAP@0.5	0.841	0.867	2.6
mAP@0.5:.95	0.436	0.464	2,8

Tabla 4.8: Comparativa de la búsqueda de hiperparámetros

4.11 Reentrenamiento final

Llegados a este punto con el dataset construido y los hiper parámetros establecidos de la manera óptima encontrada, es momento de ejecutar el entrenamiento definitivo del modelo.

Este entrenamiento se ha llevado a cabo con una resolución de 1280 x 1280 ya que es la que se usará posteriormente en la cámara, y en total de 300 épocas. Mencionar que se le estableció un *early-stopping* de 30 épocas, de forma que, si no se había mejorado el mejor modelo en las siguientes 30 épocas de entrenamiento, se daba por concluido el mismo. Con esta práctica se pretendió ahorrar recursos en la medida de lo posible y evitar entrenar un modelo que se ha sobreajustado. Así, este entrenamiento ha durado 75 épocas en lugar de las 300 programadas.

Reentrenamiento	Resolución	Épocas	N.º de imágenes de entrenamiento	N.º de imágenes de validación
1	640x640	25	1800	200
2	720x720	30	2152	539
3	720x720	100	2828	707
Final	1280x1280	300	2805	699

Tabla 4.9: Características reentrenamiento final

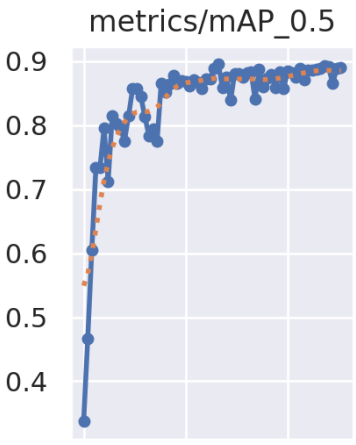

Figura 4.34: Evolución de mAP@0.5 en fnción de las épocas entrenamiento final

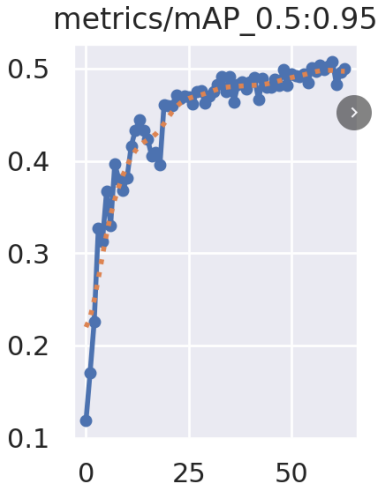

Figura 4.35: Evolución de mAP@0.5:0.95 en función de las épocas entrenamiento final

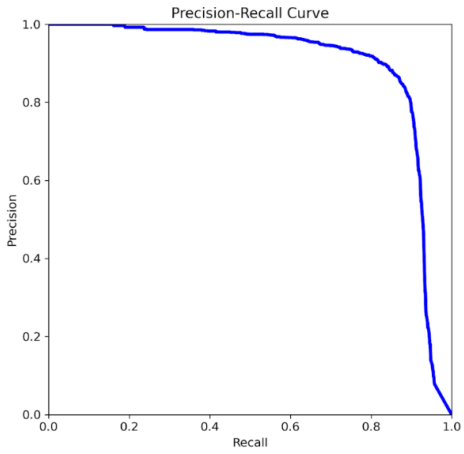

Figura 4.36: Curva precisión contra sensibilidad entrenamiento final

Reentrenamiento	mAP@0.5	mAP@0.5:0.95	Precision	Recall
1	0,8470	0.4088	0,8392	0,7933
2	0,85115	0,48478	0,9017	0,8014
3	0.88615	0.50236	0.8931	0.8440
Final	0.89612	0.5075	0.8847	0.8486

Tabla 4.10: Métricas entrenamiento final

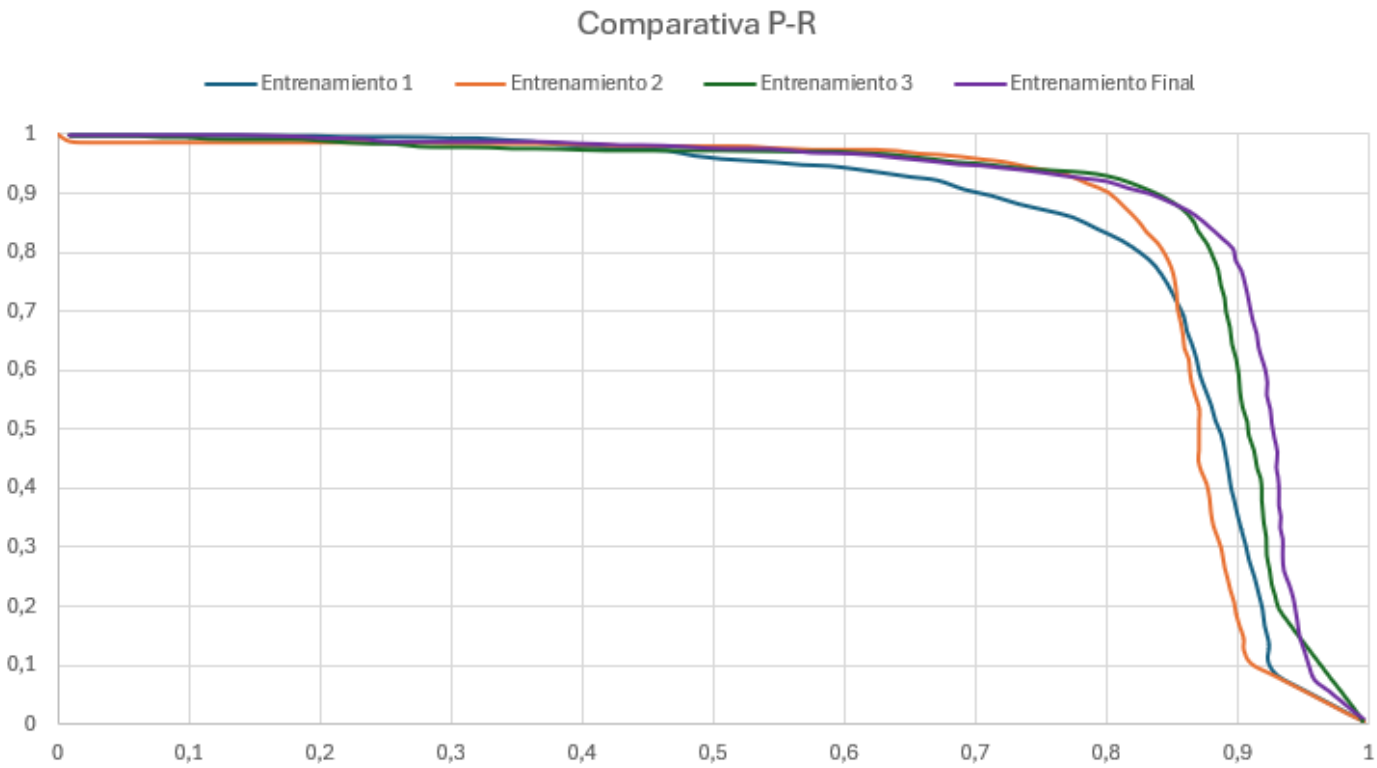

Figura 4.37: Comparativa P-R de todos los entrenamientos

Tras a analizar los resultados obtenidos en este último reentrenamiento, se puede apreciar que han aumentado ligeramente las métricas, pero no de manera significativa. Esto era esperable ya que en el anterior entrenamiento se entrenó ya con un número de épocas alta y el dataset al completo. Así, el desempeño que muestra el modelo en las pruebas de validación es similar al mostrado en el entrenamiento 3, sólo que permite procesar imágenes de resolución 1280 x 1280 mientras que el anterior las usaba 720 x 720.

4.12 Pruebas en el vehículo

Como prueba final y definitiva se han realizado unas pruebas en ASV donde se ejecutará definitivamente. Para realizar esta prueba se han lanzado diferentes piezas de basura a un lago y se ha fletado el ASV para grabar videos de los residuos flotando en el agua. Con estos videos de ha realizado un dataset de test para comprobar las métricas y desempeño del modelo en una simulación real. A partir de este resultado se pretende comprobar las limitaciones del modelo y proponer formas de mejorarlos.

En lo referente a la prueba, se han lanzado al Lago del Alamillo, Sevilla un total de 7 piezas de basura de diferentes tamaños y colores. Con este conjunto tan variado se pretende ver como el algoritmo generaliza con basura en general. Además, es conveniente ver cual el plano que utiliza la cámara del ASV para poder mejorar el dataset de entrenamiento en el futuro.

Figura 4.38: Basura usada en el experimento con escala fotográfica de 20cm

Figura 4.39: ASV usado en el experimento

4.13 Construcción y exploración del dataset de test

Gracias al video grabado por el ASV, es posible construir un dataset con imágenes del video que serán etiquetadas para formar un dataset de test. Este dataset de test lo componen 603 imágenes de las más de 35.000 que contenía el video completo. Esto se debe principalmente a que durante la mayor parte del video no hay basura en el plano de la cámara, por lo que se ha hecho un resumen de las tomas más relevantes.

Figura 4.40: Ejemplo de imagen del dataset de test etiquetada

En cuanto a la exploración del dataset, es conveniente realizar una comparativa con el dataset de entrenamiento, a fin de conocer en que se diferencian los 2 para poder contextualizar mejor los resultados del test. Analizando estas métricas se puede apreciar que las imágenes de test presentan entre 0 y 4 piezas de basura por imagen, de manera similar a la mayor parte de las imágenes de entrenamiento. Sin embargo, es importante destacar que las *bounding boxes* de test son sensiblemente más pequeñas que las de entrenamiento. Esto se debe a que en el video la basura aparece más lejos que en la mayoría de las imágenes de entrenamiento. Esto podría afectar de manera negativa a las métricas de test puesto que el algoritmo está entrenando para reconocer basura de mayor tamaño.

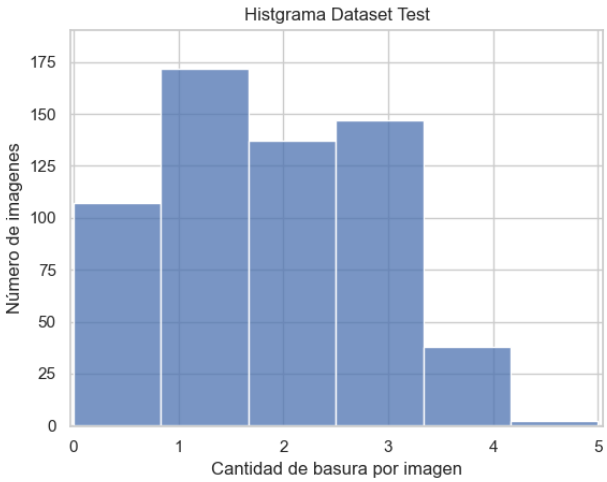

Figura 4.41: Histograma de cantidad de basura por imagen dataset test

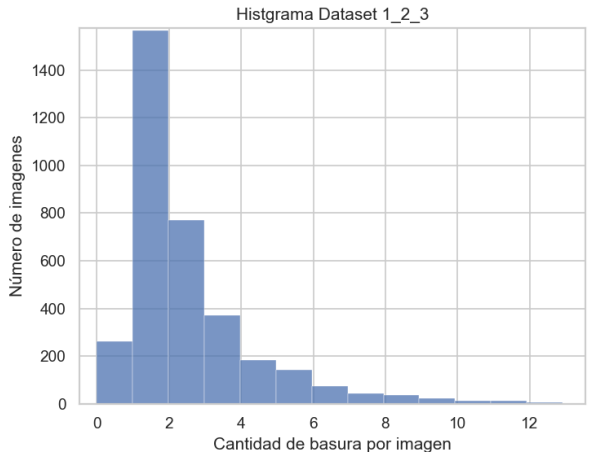

Figura 4.42: Histograma de cantidad de basura por imagen dataset entrenamiento

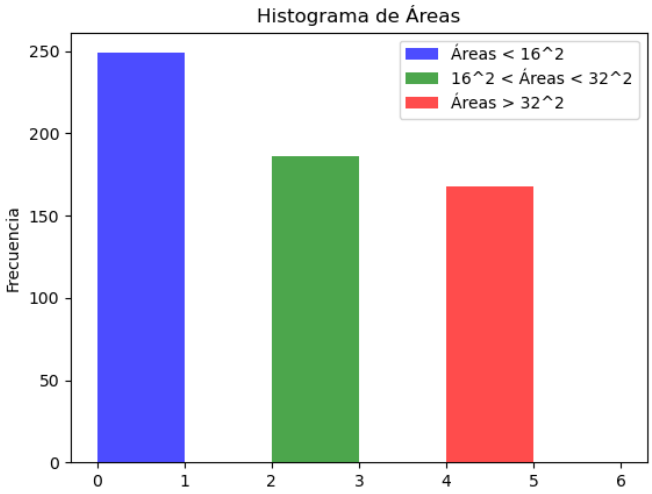

Figura 4.43: Histograma de áreas de las *Bounding boxes* dataset test

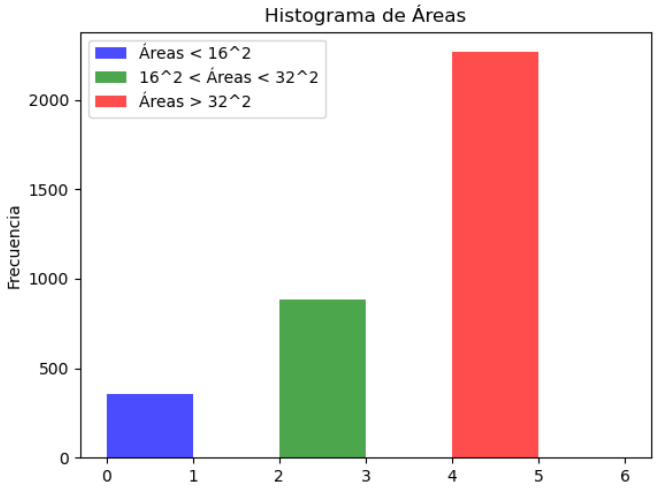

Figura 4.44: Histograma de áreas de las *Bounding boxes* dataset entrenamiento

Al analizar los resultados arrojados por las métricas resultantes del entrenamiento, se puede observar que los resultados son peores. Esto era esperable ya que el entrenamiento ha sido realizado con imágenes que no han sido tomadas por el ASV realmente.

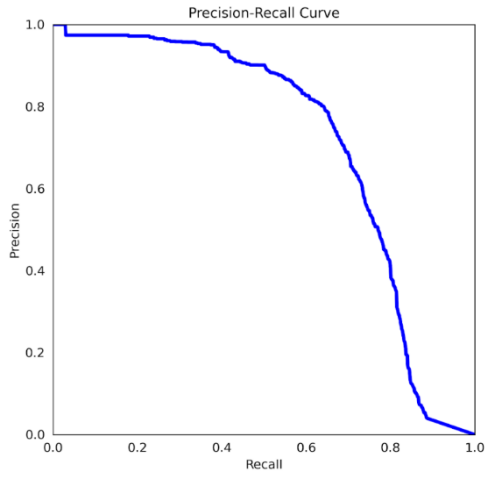

Figura 4.45: Curva Precisión contra Sensibilidad test

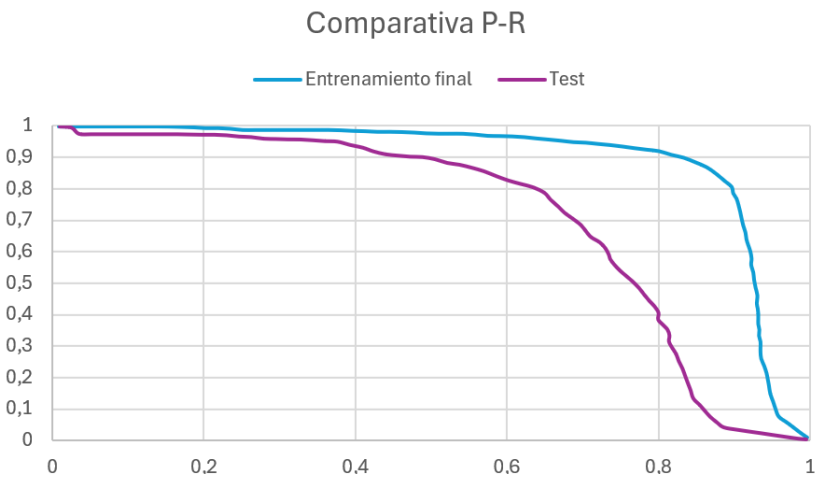

Figura 4.46: Comparativa P-R de resultados de entrenamiento y de test

	mAP@0.5	Precision	Recall
Entrenamiento	0,8470	0,8392	0,7933
Test	0,7150	0,7870	0,6495

Tabla 4.11: Métricas de test

Analizando los ejemplos que resultan del test, se puede observar que el modelo tiene problemas con la presencia del cielo. El modelo genera de manera sistemática falsos positivos debido a la presencia de nubes en el cielo. Esto deja abierta la posibilidad de mejora en el dataset de entrenamiento ya que simplemente con la inclusión de imágenes de basura donde aparezcan ciertos elementos como nubes en el cielo o patos, se mejorará el acierto de modelo de manera notable.

Figura 4.47: Subconjunto de test etiquetado

Figura 4.48: Resultados de la validación en el dataset de test

5 CONCLUSIONES Y FUTUROS TRABAJOS

Una vez explorados y analizados los resultados arrojados por la implementación del modelo desarrollado en este proyecto, este capítulo se dedicará a analizar las conclusiones que se pueden sacar de la realización de este proyecto, así como a proponer líneas de investigación y desarrollo para futuros trabajos relacionados con este trabajo.

6.1 Conclusiones

El objetivo principal de este trabajo era desarrollar un algoritmo de visión artificial para detectar contaminantes en recursos hídricos que pudiera ejecutarse desde un ASV operara en la superficie del agua. Este algoritmo debía de ser capaz de detectar basura en tiempo real utilizando las imágenes capturadas por la cámara presente en el vehículo con la finalidad de permitir una respuesta rápida que evitara la propagación de agentes contaminantes en los recursos hídricos siendo lo suficientemente robusto y eficiente para operar de manera precisa en entornos ante condiciones adversas.

A lo largo de este trabajo, se ha desarrollado y optimizado un algoritmo basado en YOLOv5, que debe ser lo suficientemente preciso para que el vehículo realice sus tareas sin supervisión humana y lo suficientemente eficiente para ejecutarse en hardware con recursos computacionales limitados. Este algoritmo ha sido entrenado y probado en condiciones similares a las reales, asegurando su funcionalidad y eficiencia. Para cumplir con estos objetivos se ha optado por un enfoque basado en la fusión iterativa de distintos conjuntos de imágenes que contuvieran basura en la superficie del agua.

Una vez finalizado el proyecto y analizado los resultados se puede llegar a las siguientes conclusiones. La primera es que el enfoque de fusionar distintos datasets relevantes que no han sido tomadas desde el ASV ha sido todo un acierto, ya que ha permitido crear un gran conjunto de imágenes para entrenar el algoritmo de detección evitando el proceso tener que tomar imágenes del ASV directamente, evitando también todos los costes económicos y de tiempo que se requerirían. Gracias a este dataset de imágenes, ha sido posible obtener un algoritmo lo suficientemente preciso como para poder ser ejecutado en el ASV y proporcionar resultados aceptables. Sin embargo, las pruebas finales sobre un entorno real de ejecución han mostrado que, si bien el modelo muestra en desempeño notable en general, es necesario seguir mejorando el dataset ya que existen ciertas situaciones, como la presencia de nubes, que no están ejemplificadas en el dataset resultando errores sistemáticos en la detección y clasificación de basura en esas situaciones.

6.2 Futuros trabajos

Ante los avances y resultados mostrados en este trabajo, pueden surgir varias líneas de investigación y desarrollo futuras para mejorar el desempeño global e integración del este algoritmo de detección. Estas líneas van desde la reproducción del enfoque presentado en este trabajo para otros ámbitos en la detección hasta la propia mejora de los datos de entrenamiento para mejorar los resultados.

~~La primera de ellas sería tratar de imitar el enfoque de entrenar un modelo de detección y clasificación en vehículos móviles a partir de dataset relevantes en lugar de tomar las imágenes desde la misma plataforma. Gracias al éxito de los resultados mostrados resulta muy prometedor adaptar este enfoque a la detección de otro tipo de objetos como pudieran ser señales de tráfico o basura presente en las calles en lugar de medios acuáticos.~~

La segunda línea de mejora va enfocada en mejorar el dataset ya existen. Como se ha mostrado, aunque el desempeño del modelo es bueno únicamente con datasets fusionados, sigue siendo necesario la inclusión de imágenes tomadas directamente desde de ASV para evitar los errores en ciertas situaciones. Así, se podría tratar de mejorar el dataset existente mediante la realización de distintas misiones del ASV en distintos entornos acuáticos con la correspondiente toma de imágenes y su fusión con el dataset ya existente. Esto permitiría que el algoritmo detectará mejor la basura con respecto a otro tipo de objetos presentes en el medio como pudieran ser nubes, seres vivos como humanos o aves o incluso otros objetos como pequeños barcos o rocas.

Otro tipo de modificación que se podría llevar a cabo para mejorar el desempeño del algoritmo sin necesidad de volver a entrenar el modelo con más datos sería restringir la distancia máxima de detección. Como se ha visto en los resultados, la mayoría de los fallos se concentran en objetos que por lo general están alejados del ASV como nubes o basura tan lejana que se percibe como de muy pequeño tamaño. Así, teniendo en cuenta que la función principal del algoritmo es permitir que la basura detectada sea recogida posteriormente y gracias a la capacidad de la cámara de medir la distancia que existe hasta el objeto, se podría tratar de sólo considerar como válidas las detecciones de objetos en un rango de distancia con respecto al ASV. De este modo, se eliminarían todos los fallos relacionados con la detección de nubes como basura, pues estarían muy alejadas. En el caso de la presencia de basura, pero muy lejana, en el momento en que el ASV se acercará a ellas, pasarían a ser consideradas detecciones válidas.

~~En resumen, el desarrollo y la implementación de un algoritmo de visión artificial para la detección de basura en recursos hídricos ha demostrado ser un avance significativo en la lucha contra la contaminación acuática. Este proyecto no solo ha validado la viabilidad del uso de algoritmos avanzados en vehículos autónomos, sino que también ha destacado la importancia de contar con datasets robustos y representativos. Las conclusiones obtenidas y las propuestas de futuras mejoras establecen una base sólida para la evolución y optimización de sistemas de detección y recolección de residuos, abriendo nuevas vías de investigación y aplicaciones en diferentes entornos. Con un enfoque continuo en la innovación y la precisión, este trabajo contribuye significativamente a los esfuerzos globales por proteger y preservar nuestros valiosos recursos hídricos.~~

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

BIBLIOGRAFÍA

[1] Europa Press, 2016. "¿Cuál fue el primer río contaminado de la historia?", iagua.es [Online]. Disponible: https://www.iagua.es/noticias/ep/16/12/05/cual-fue-primer-rio-contaminado-historia

[2] Mario Acevedo, 2017. "LA CONTAMINACIÓN MARINA Y LA EVOLUCIÓN DE SU NORMATIVA INTERNACIONAL", Universidad de Comillas. Disponible: https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/25031/TFM000848.pdf?sequence=1&isAllowed=y

[3] S. T. Marín and D. G. Reina, 2022. "Sistema de Detección de Residuos en Entornos Portuarios mediante Flota Vehículos Autónomos de Superficie", Vicerrectorado de Investigación Universidad de Sevilla. [Online]. Disponible: https://investigacion.us.es/sisius/sis_proyecto.php?idproy=35646

[4] C. Liu, N. Xie, X. Yang … X. Liu , 2022. "A Domestic Trash Detection Model Based on Improved YOLOX", mpi.com. Disponible: Sensors | Free Full-Text | A Domestic Trash Detection Model Based on Improved YOLOX (mdpi.com)

[5] S. Majchrowska, A. Mikolajczyk, M. Ferlin … K. Majeck ,2022. "Deep learning-based waste detection in natural and urban environments", Waste Management. Disponible: Deep learning-based waste detection in natural and urban environments - ScienceDirect

[6] Mind Yang and Gary Thung, 2016. "Classification of Trash for Recyclability Status", Universidad de Stanford. Disponible: https://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf

[7] Pedro F. Proença and Pedro Simões, 2020. "TACO: Trash Annotations in Context for Litter Detection", Cornell University. Disponible: https://arxiv.org/pdf/2003.06975

[8] Mingxing Tan, Ruoming Pang and Quoc V. Le, 2020. "EfficientDet: Scalable and Efficient Object Detection", Computer Vision Foundation. Disponible: https://openaccess.thecvf.com/content_CVPR_2020/papers/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.pdf

[9] M.S. Rad, A. Kaenel, … and J.Thiran, 2017. “A Computer Vision System to Localize and Classify Wastes on the Streets” ,11th International Conference, ICVS 2017 Shenzhen, China, July 10–13, 2017 pp. 208-217.

[10] P. Sermanet, D. Eigen, … and Y. LeCun, 2014. “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, ICRL 2014. Disponible: https://arxiv.org/abs/1312.6229

[11] C. Szegedy, W. Liu, … and A. Rabinovich, 2014 “Going deeper with convolutions”, Google Inc. Disponible: https://arxiv.org/pdf/1409.4842v1

[12] ChunMing Wu, YiQian Sun, TiaoJUn Wrang and YaLi Liu, 2022. “Underwater trash detection algorithm based on improved YOLOv5s”, Journal of Real-Time Image Processing volume 19, pp 910-920. Disponible: Underwater trash detection algorithm based on improved YOLOv5s | Journal of Real-Time Image Processing (springer.com)

[13] Fulton Michael S, Hong, Jungseok, Sattar, Junaed. ,2020. “Trash-ICRA19: A Bounding Box Labeled Dataset of Underwater Trash”,University Digital Conservancy. Disponible: https://doi.org/10.13020/x0qn-y082

[14] M. Tharani, A. W. Amin, … and A. Muhammad, 2021. “Trash Detection on Water Channels”, 28th International Conference, ICONIP 2021 Sanur, Bali, Indonesia, December 8–12, 2021 Proceedings, Part I, pp 403-413.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, 2015. “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany. Disponible: https://arxiv.org/abs/1505.04597v1

[16] R. Girshick, J. Donahue, T. Darrell y J. Malik, 2014. “Rich feature hierarchies for accurate object detection and semantic segmentation", Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. [Online]. Disponible: https://openaccess.thecvf.com/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_p aper.pdf

[17] R. Girshick, 2015. “Fast R-CNN”, Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1440-1448. [Online]. Disponible: https://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

[18] S. Ren, K. He, R. Girshick y J. Sun, 2015. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, Advances in neural information processing systems, vol. 28 [Online]. Disponible: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[19] W. Liu, D. Anguelov, D. Erhan y C. Szegedy, 2016. “SSD: Single Shot MultiBox Detector”, Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, pp. 21-37. [Online]. Disponible: https://arxiv.org/pdf/1512.02325.pdf

[20] MingXing Tan, Ruoming Pang and Quoc V.Le , 2020. “EfficientDet: Scalable and Efficient Object Detection” Google Research, Brain Team [Online]. Disponible: https://arxiv.org/abs/1911.09070v7

[21] Vineeth S Subramanyam, 2021 “Basics of Bounding Boxes”,medium.com [Online]. Disponible: https://medium.com/analytics-vidhya/basics-of-bounding-boxes-94e583b5e16c

[22] J.Redmon , A.Farhadi , 2018 “YOLOv3: An Incremental Improvement” University of Washington Disponible: https://arxiv.org/abs/1804.02767v1

[23] Z. Zhang, T. He, H. Zhang, Z.Zhang, J. Xie , M. Li 2019 “Bag of Freebies for Training Object Detection Neural Networks” Amazon Web Services Disponible: https://arxiv.org/abs/1902.04103

[24] Devin Schumacher “CSPDarknet53”, serp.ai [Online] Disponible: https://serp.ai/cspdarknet53/

[25] “Tips for Best Training Results” ultralytics.com [Online] Disponible: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/

[26] Glen Jocher ,2021 “v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations”, ultralytics.com [Online]. Disponible: https://github.com/ultralytics/yolov5/releases/tag/v5.0

[27] A. Benjumea, I. Teeti, F.Cuzzolin, A.Bradley , 2023 “YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles” brokees.ac.uk https://arxiv.org/pdf/2112.11798

[28] Ruman ,2023 “YOLO Data Augmentation Explained” medium.com [Online] Disponible: https://rumn.medium.com/yolo-data-augmentation-explained-turbocharge-your-object-detection-model-94c33278303a

[29] Verid, 2022 “¿Qué es el garbage in y garbage out y cómo evitarlo?”, verifid.net [Online]. Disponible:

https://www.verifid.net/blog/articulo/que-es-el-garbage-in-y-garbage-out-y-como-evitarlo152059

[30] shaip ,2022 "¿Cuál es el volumen óptimo de datos de entrenamiento que necesita para un proyecto de IA?", es.shaip.com [Online]. Disponible: https://es.shaip.com/blog/how-much-training-data-is-enough/

[31] "What is YOLO Darknet TXT?", roboflow.com https://roboflow.com/formats/yolo-darknet-txt

[32] "ZED 2", stereolabs.com https://www.stereolabs.com/products/zed-2

[33] "Serie Jetson Xavier NX", nvdia.com https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/jetson-xavier-nx/

[34] "CVAT". CVAT. https://www.cvat.ai

[35] "scikit-learn", scit-learn.org. https://scikit-learn.org/stable/

[36] Glenn Jocher, 2022. "yolov5", github.com. https://github.com/ultralytics/yolov5

[37] "PyTorch", pytorch.org. https://pytorch.org/

[38] "Google Colaboratory", Google. https://colab.google/

[39] "Matplotlib", matplotlib.org. https://matplotlib.org/

[40] "Seaborn", seaborn.pydata.org. https://seaborn.pydata.org/

[41] "OpenCv", opencv.org. https://opencv.org/

[42] "Tips for Best Training Results" ultralytics.com [Online] Disponible: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/

[43] Charles Gaillard, Rémy Brossard ,2022"The Danger of Batch Normalization in Deep Learning" mindee.com [Online]. Disponible: https://www.mindee.com/blog/batch-normalization

[44] Y. Cheng, J. Zhu, M. Jiang, J. Fu … and Y. Bengio ,2021 "FloW: A Dataset and Benchmark for Floating Waste Detection in Inland Waters", IEEE.org. Disponible: https://ieeexplore.ieee.org/document/9710581

[45] SJTU, 2024"floating_litter" roboflow.com floating_litter Dataset > Overview (roboflow.com)

[46] IPSA, 2023 "WCB5G", roboflow.com https://universe.roboflow.com/ipsa-4wlge/wcb5g