

PROYECTO FIN DE GRADO

TÍTULO: Desarrollo de una aplicación para la detección automatizada de lubinas en vídeos 360° utilizando YOLOv8

AUTOR: Raúl Calderón Moya

TITULACIÓN: Grado en Ingeniería Telemática

TUTORA: Juana María Gutiérrez Arriola

DEPARTAMENTO: Ingeniería Telemática y Electrónica (DTE)

VºBº TUTORA

Miembros del Tribunal Calificador:

PRESIDENTE: Daniel de la Prida Caballero

TUTORA: Juana María Gutiérrez Arriola

SECRETARIO: Nicolás Sáenz Lechón

Fecha de lectura: 12/02/2025

Calificación:

El Secretario,

Resumen

En la actualidad, el creciente problema de la sobrepesca amenaza la subsistencia de numerosas especies marinas y pone en riesgo los ecosistemas acuáticos. Ante esta problemática, la acuicultura ha surgido como una alternativa sostenible y que complementa a la pesca tradicional. Esta práctica permite la cría de peces y otras especies para consumo humano, ofreciendo una solución que favorece la conservación de los hábitats marinos.

A pesar de todo, para que la acuicultura sea eficiente, es necesario optimizar los recursos empleados en la cría de organismos acuáticos. Es fundamental analizar y entender los comportamientos de los animales para mejorar prácticas de alimentación y, en definitiva, mejorar su calidad de vida. En este contexto, las tecnologías de visión por ordenador y la inteligencia artificial desempeñan un papel clave. Entre los métodos más destacados se encuentran el análisis de flujo óptico y las redes convolucionales neuronales (CNN). Una de las implementaciones más avanzadas y utilizadas es YOLOv8, que se caracteriza por su alta precisión y sencillez de uso.

La primera tarea es el proceso de anotación de imágenes para crear un conjunto de datos lo suficientemente amplio y diverso como para que el modelo sea capaz de generalizar correctamente. En este caso, se han etiquetado 200 imágenes que se han repartido en: un 80% en el conjunto de entrenamiento (160 imágenes), un 15% en el conjunto de validación (30 imágenes) y un 5% en el conjunto de test (10 imágenes).

Posteriormente, se realizan múltiples iteraciones de entrenamiento y se deben ajustar los hiperparámetros para lograr el mejor resultado posible en términos de menor número de pérdidas y precisión del modelo medidos en métricas como mAP50, mAP50-95, F1-Score, entre otros. Una vez que se ha logrado el mejor resultado, se desarrolla una aplicación capaz de reproducir vídeos 360°, en la cual el modelo realiza las detecciones que se muestran por medio de cajas delimitadoras asociadas a cada lubina. Además, se ha integrado un módulo de código que lleva un registro por fotograma procesado de todas las cajas delimitadoras detectadas y lo almacena en un archivo de texto, lo cual es clave para poder analizar comportamientos de cara a futuro.

Para la elaboración de todo el proyecto se ha tenido como referencia toda la documentación del sitio web de Ultralytics, tanto para la parte de entrenamiento como para la parte de validación y posteriores ajustes de parámetros del modelo para mejorar su desempeño. En cuanto a resultados, el modelo YOLOv8 se ha entrenado hasta conseguir unos valores de precisión de IoU de 74,39%, lo cual es un resultado muy bueno. A pesar de los buenos resultados, se ha deducido que no siempre generaliza correctamente, lo cual se aprecia en las métricas de pérdidas en la parte del conjunto de validación, y esto significa que se hubiera requerido de un mayor conjunto de imágenes tanto en los conjuntos de validación como de entrenamiento ya que pasar demasiadas imágenes al conjunto de validación sin compensarlo agregando nuevas imágenes al de entrenamiento podría desbalancear la relación de imágenes recomendable entre ambos conjuntos.

Abstract

Currently, the growing issue of overfishing threatens the survival of numerous marine species and jeopardizes aquatic ecosystems. In response to this problem, aquaculture has emerged as a sustainable alternative that complements traditional fishing. This practice allows for the farming of fish and other species for human consumption, providing a solution that supports the conservation of marine habitats.

However, for aquaculture to be efficient, it is essential to optimize the resources used in the farming of aquatic organisms. Analyzing and understanding animal behavior is fundamental for improving feeding practices and ultimately enhancing their quality of life. In this context, computer vision technologies and artificial intelligence play a key role. Among the most notable methods are optical flow analysis and convolutional neural networks (CNN). One of the most advanced and widely used implementations is YOLOv8, known for its high accuracy and ease of use.

The research begins with an image annotation process using the CVAT tool, creating a dataset that is sufficiently large and diverse for the model to correctly generalize. Subsequently, multiple training iterations are carried out, and hyperparameters must be adjusted to achieve the best possible results in terms of minimal loss and model accuracy, measured by metrics such as mAP50, mAP50-95, F1-Score, among others. Once the optimal result is achieved, an application capable of playing 360° videos is developed, in which the model performs detections displayed through Bounding boxes associated with each sea bass. Additionally, a code module was integrated to log each processed frame's detected Bounding boxes and store them in a text file, which is crucial for future behavior analysis.

For the entire project, the documentation from the Ultralytics website was used as a reference, both for the training phase and for validation and subsequent parameter adjustments to improve model performance. Regarding the results, the YOLOv8 model was trained to achieve IoU precision values of 74,39%, which is considered very good. Despite these promising results, it was found that the model does not always generalize correctly, as seen in the loss metrics on the validation set. This indicates that a larger set of images would have been necessary for both the training and validation sets, as allocating too many images to the validation set could unbalance the recommended image ratio between the two sets. In this case, 80% of the images were used for training (160 images), 15% for validation (30 images), and 5% for testing (10 images).

Índice de figuras

Figura 1 Aparición regional de la acuicultura documentada en la literatura arqueológica [1] .	1
Figura 2 Diagrama del Aprendizaje supervisado [13]	9
Figura 3 Diagrama del aprendizaje no supervisado [14]	10
Figura 4 Esquema visual del Perceptrón [18]	11
Figura 5 Fórmula matemática del perceptrón [19].....	11
Figura 6 Ejemplo de Red MLP (MultiLayer Perceptron) [21]	12
Figura 7 Diagrama de funcionamiento de una neurona artificial [22]	12
Figura 8 Proceso de aprendizaje de una red neuronal [22].....	13
Figura 9 Estructura básica de una red neuronal convolucional (CNN) [29]	15
Figura 10 Proceso de convolución en una CNN [30].....	16
Figura 11 Ejemplo de funcionamiento de Pooling [30]	17
Figura 12 Ejemplo de CNN con proceso de convolución y posterior submuestreo [32]	18
Figura 13 Tareas típicas de YOLOv8 [34]	19
Figura 14 Ranking de lenguajes de programación más demandados en 2023 [40].....	22
Figura 15 Intefaz Insta360 Studio reproduciendo vídeo 360°	27
Figura 16 Imagen panorámica obtenida por el programa	28
Figura 17 Ejemplo de imagen etiquetada con CVAT	28
Figura 18 Archivo YAML de entrenamiento.....	29
Figura 19 Sistema de Archivos utilizado para el entrenamiento de la CNN.....	30
Figura 20 Matriz de confusión [42]	31
Figura 23 Ejemplo de resultados de entrenamiento de YOLOv8.....	32
Figura 24 Fórmula para el cálculo de IoU [45].....	33
Figura 25 Ejemplo de detecciones con IoU [45]	33
Figura 26 Gráfica de resultados de IoU para una imagen.....	35
Figura 27 Representación visual etiquetas de referencia frente a las predichas	36
Figura 28 Programa de detección de lubinas en entorno 360°.....	37
Figura 29 Programa de detección de lubinas en vista panorámica.....	38
Figura 30 Ejemplo fichero detecciones realizadas por el modelo entrenado.....	38
Figura 31 Valores Bounding box en fichero de texto [49]	39
Figura 32 Gráficas de métricas del primer entrenamiento.....	41
Figura 33 Gráficas recall y precisión primer entrenamiento	42
Figura 34 Gráficas de métricas del segundo entrenamiento	43
Figura 35 Gráficas de recall y precisión del segundo entrenamiento	43
Figura 36 Gráficas de métricas del tercer entrenamiento.....	44
Figura 37 Gráficas de recall y precisión tercer entrenamiento.....	45
Figura 38 Gráficas de métricas del cuarto entrenamiento	46
Figura 39 Gráficas de recall y precisión del cuarto entrenamiento.....	46
Figura 40 Gráficas de métricas del quinto entrenamiento.....	48
Figura 41 Gráficas de precisión y recall del quinto entrenamiento	48
Figura 42 Gráfica de métricas del sexto entrenamiento.....	49
Figura 43 Gráficas de precisión y recall del sexto entrenamiento.....	50

Figura 44 Gráfica Evolución mAP50 para los entrenamientos	51
Figura 45 Gráfica Evolución mAP50-95 para los entrenamientos	51
Figura 46 Gráfica Evolución Box(P) en los entrenamientos	52
Figura 47 Gráfica Evolución Box(R) en los entrenamientos	52
Figura 48 Histograma de valor de IoU acumulado	53
Figura 49 Gráficas de Resultados IoU para la imagen 1	54
Figura 50 Gráficas de resultados IoU para la imagen 2	54
Figura 51 Gráficas de resultados IoU para la imagen 3	54
Figura 52 Gráficas de resultados IoU para la imagen 4	54
Figura 53 Gráficas de resultados IoU para la imagen 5	55
Figura 54 Gráfica de resultados IoU para la imagen 6	55
Figura 55 Gráfica de resultados IoU para la imagen 7	55
Figura 56 Gráfica de resultados IoU para la imagen 8	55
Figura 57 Gráfica de resultados IoU para la imagen 9	56
Figura 58 Gráfica de resultados IoU para imagen 10	56
Figura 59 Perturbación por sombra humana en el medio acuático	57
Figura 60 Situación previa a la alimentación o posterior a una perturbación humana	57
Figura 61 Situación de alimentación de lubinas	58
Figura 62 Situación estándar de las lubinas	59
Figura 63 Gráfica de detecciones por fotograma de lubinas en un entorno real	59
Figura 64 Gráfica de porcentaje de área cajas delimitadoras por cada fotograma	60
Figura 65 Gráfica tamaño medio cajas delimitadoras en cada fotograma	61
Figura 66 Gráfica centroide X por fotograma	62
Figura 67 Gráfica centroide Y por fotograma	62
Figura 68 Objetivos de desarrollo sostenible de la ONU [53]	68
Figura 69 Ejemplo de Data Augmentation [58]	78
Figura 70 Gráficas de relación entre parámetros y latencia frente a precisión [59]	79
Figura 71 Tareas compatibles con modelos YOLOv8 [59]	79
Figura 72 Desplegable de opciones de exportado de vídeos 360°	80
Figura 73 Archivo Excel con los paths de los vídeos	81
Figura 74 Interfaz principal de CVAT	82
Figura 75 Campos para creación de un proyecto en CVAT	82
Figura 76 Menú de un proyecto en CVAT	82
Figura 77 Menú de una tarea (task) en CVAT	83
Figura 78 Menú de las tareas en CVAT	83
Figura 79 Menú de la tarea creada en CVAT	84
Figura 80 Menú de etiquetado de imágenes en CVAT	84
Figura 81 Desplegable opción "Draw a new rectangle" en CVAT	84
Figura 82 Etiquetado de una lubina con la opción de dibujar un rectángulo	85
Figura 83 Cuadro de opciones para navegar entre imágenes en CVAT	85
Figura 84 Exportado de imagenes de una tarea en CVAT	85
Figura 85 Selección del formato de salida de las etiquetas con CVAT	85
Figura 86 Archivos en el fichero descargado de etiquetas de CVAT	86

Figura 87 Contenido de la carpeta obj_train_data	86
Figura 88 Sistemas de ficheros para el entrenamiento.....	86
Figura 89 Fichero dataset.yaml	87
Figura 90 Resultados de entrenamiento generados por YOLOv8.....	87
Figura 91 Resultados obtenidos por el programa de IoU	88

Índice de tablas

Tabla 1 Tabla Valores Bounding boxes en fichero de texto [50].....	38
Tabla 2 Tabla resumen métricas principales de entrenamiento	52
Tabla 3 Presupuesto total del proyecto	65

Lista de acrónimos

Acrónimo	Significado
Adam	Adaptative Moment Estimation
AdamW	Adaptative Moment Estimation with Weight decay
BPNN	Back Propagation Neural Network
CITSEM	Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CVAT	Computer Vision Anotation Tool
DPM	Deformable Part Models
ECTS	Sistema europeo de transferencia y acumulación de créditos
ETSIST	Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación
FFNN	Feed Forward Neural Network
FN	False Negative
FP	False Positive
FPS	Frame Per Second
GAMMA	Grupo de Aplicaciones Multimedia y Acústica
GMM	Gaussian Mixture Model
GPU	Graphics processing unit
IA	Inteligencia Artificial
IoU	Intersection over Union
IVA	Impuesto sobre el Valor Añadido
LR	Learning Rate
MAP	Mean Average Precision
ML	Machine Learning
MLP	MultiLayer Perceptron
MNN	Modular Neural Network
MSE	Mean Square Error
NAS	Network Attached Storage
NumPy	Numerical Python
ODS	Objetivos de Desarrollo Sostenible
OpenCV	Open Source Computer Vision
PCA	Principal Component Analysis
RAM	Random Access Memory
RBF	Radial Basis Function
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neral Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UPM	Universidad Politécnica de Madrid
YAML	Yet Another Markup Language
YOLO	You Only Look Once

Índice de contenidos

Resumen	i
Abstract	iii
Índice de figuras	v
Índice de tablas	vii
Lista de acrónimos.....	ix
1. Introducción.....	1
1.1 Marco y motivación del proyecto.....	1
1.2 Objetivos técnicos y académicos	2
1.3 Estructura del resto de la memoria	3
2. Marco tecnológico.....	5
2.1 Tecnologías empleadas para el conteo en el contexto acuícola	5
2.1.1 Conteo manual.....	5
2.1.2 Conteo basado en tecnología de sensores	5
2.1.3 Conteo basado en tecnología de visión artificial	6
2.1.4 Conteo mediante sistemas acústicos	7
2.2 Aprendizaje automático	7
2.2.1 ¿Cómo funciona el aprendizaje automático?	8
2.2.2 Tipos de aprendizaje automático.....	9
2.3 Redes Neuronales: origen y evolución	10
2.4 Detección de Objetos usando redes neuronales.....	13
2.4.1 Tipos de redes neuronales	14
2.5 Redes neuronales convolucionales (CNN)	15
2.6 YOLOv8.....	18
2.6.1 Visión por computadora con YOLOv8.....	18
2.6.2 Ventajas de YOLO frente a otros sistemas de detección de objetos	20
2.6.3 Aplicaciones de YOLOv8 en acuicultura	21
2.7 Herramientas de despliegue de YOLOv8	21
3. Especificaciones y restricciones de diseño	23
3.1 Especificaciones del diseño.....	23
3.2 Restricciones del diseño	23
4. Descripción de la solución propuesta	27
4.1 Creación del dataset.....	27
4.2 Entrenamientos del modelo de YOLOv8	29
4.3 Validación de resultados de entrenamiento	30
4.4 Optimizadores de entrenamiento.....	33
4.5 Evaluación de resultados de IoU	35
4.6 Programa final de detecciones	36
5. Resultados	41

5.1	Evolución de entrenamientos	41
5.1.1	Primer entrenamiento.....	41
5.1.2	Segundo entrenamiento	42
5.1.3	Tercer entrenamiento	44
5.1.4	Cuarto entrenamiento.....	45
5.1.5	Quinto entrenamiento	47
5.1.6	Sexto entrenamiento.....	49
5.1.7	Conclusiones etapas de entrenamiento	50
5.2	Resultados de IoU (Intersection over Union).....	53
5.3	Resultados en entornos reales.....	56
5.3.1	Detecciones por Frame	59
5.3.2	Porcentaje de área ocupada por cajas delimitadoras	60
5.3.3	Tamaño medio de las cajas delimitadoras	60
5.3.4	Centroides X e Y por fotograma	61
6.	Presupuesto	64
6.1	Recursos Hardware	64
6.2	Recursos Software.....	64
6.3	Costos finales	64
7.	Impacto del proyecto	67
7.1	Impacto social	67
7.2	Impacto de salud.....	67
7.3	Impacto ambiental	67
7.4	Impacto económico.....	67
7.5	Impacto tecnológico.....	67
7.6	Contribución a los Objetivos de Desarrollo Sostenible (ODS)	68
8.	Conclusiones	69
8.1	Conclusiones.....	69
8.2	Trabajos futuros	70
9.	Referencias	71
Anexo	77	
A.1	Hiperparámetros de YOLOv8	77
A.2	Versiones de YOLOv8	79
Manual de usuario	80	
A.3	Exportado del vídeo 360°	80
A.4	Etiquetado de imágenes.....	81
A.5	Conjuntos de datos para el entrenamiento	86
A.6	Entrenamiento y ficheros de métricas.....	87
A.7	Funcionamiento de la aplicación	88

1. Introducción

1.1 Marco y motivación del proyecto

La acuicultura es una actividad basada en la cría de especies acuáticas en entornos controlados como en piscifactorías o jaulas en zonas costeras, entre otros. Dicha actividad desempeña un papel crucial en la seguridad alimentaria y en la gestión sostenible de los recursos necesarios, pudiendo convertirse en una práctica clave para garantizar el suministro de comida en el planeta. Esta práctica se ha desarrollado durante milenios, con evidencias arqueológicas que muestran su existencia en civilizaciones antiguas, como en Egipto y China. Como se aprecia, desde tiempos remotos se han criado todo tipo de especies acuáticas para el consumo humano en mayor o menor medida. Esta técnica no se ha empleado de la misma manera y ni siquiera con las mismas especies, ya que según la ubicación geográfica esto cambia. Un ejemplo de ello es el cultivo de algas en Japón en el siglo XVII, mientras que en Europa la acuicultura se centró fundamentalmente en la cría de carpas en monasterios para los periodos de ayuno. En la siguiente figura, se muestra por región la fecha en la que hay registros de acuicultura, junto con la especie que usaban. [1]

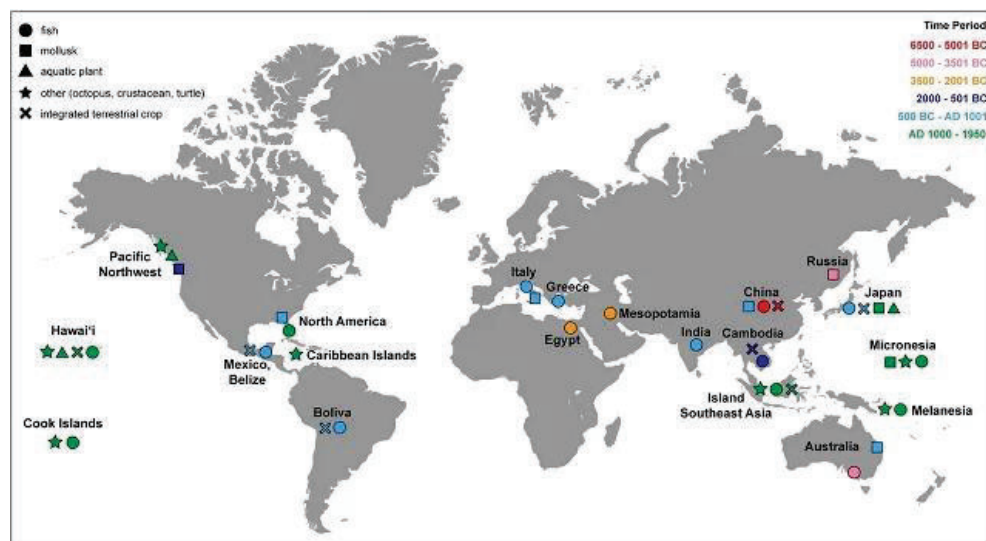


Figura 1 Aparición regional de la acuicultura documentada en la literatura arqueológica [1]

A lo largo del tiempo, las culturas han ido perfeccionando la técnica y haciendo que evolucione hacia versiones mucho más sofisticadas, lo cual se demuestra en su uso incremental con el paso de los siglos. De hecho, se ha pasado de utilizar estanques artificiales como se hacía en la antigüedad a usar sistemas de recirculación del agua, jaulas flotantes, junto con técnicas que optimizan y mejoran las capacidades de cría de seres acuáticos. [1] [2]

Como se ve, la acuicultura ha evolucionado hasta convertirse en una solución clave para evitar la sobreexplotación de recursos marinos y satisfacer las necesidades exponenciales de alimentación de la población. En función de la ubicación, se clasifica en acuicultura marina para mares u océanos centrada en zonas costeras y la variante continental, más destinada para zonas de interior. Además, la importancia de la acuicultura es evidente no solo por los

beneficios de producción de alimento sino también económicos, al generar empleo en zonas costeras y rurales impulsando la economía. También, su aportación a la sostenibilidad es crucial al ofrecer una alternativa a la pesca intensiva, reduciendo el impacto en los ecosistemas marinos, haciendo que éstos puedan recuperarse mucho más fácilmente de la pesca convencional y mantener su biodiversidad. [1] [3]

A pesar de las grandes ventajas de esta técnica, se deben tener en cuenta un gran número de desafíos como los desechos orgánicos que se generan, el espacio que se requiere para la cría de los peces y el posible desequilibrio que pueda producirse al introducir especies exóticas en un entorno local. En la actualidad, se están investigando alternativas para poder reducir al mínimo los inconvenientes que pueda causar; por ejemplo, usando dos especies en lugar de una para que los residuos generados por una sean aprovechados por la otra, lo cual podría mejorar aún más las condiciones de cara a un futuro próximo y hacer de la acuicultura una alternativa aún más interesante y sostenible. [1] [4]

Por otro lado, los avances tecnológicos ofrecen soluciones prometedoras. Se pretende utilizar la tecnología para conseguir optimizar todo el proceso; por ejemplo, la cantidad de alimento necesaria en cada momento para un correcto crecimiento de los animales, sin excesos ni carencias para garantizar su bienestar y evitar el despilfarro de recursos. También, se contemplan otros planes como el análisis de los comportamientos de los peces para aprender mucho más y así poder atender mejor sus necesidades. Es aquí donde las técnicas de aprendizaje automático pueden ser de utilidad al aplicar redes neuronales para optimizar los recursos mediante el análisis de imágenes utilizando Deep learning. [1]

Por tanto, en este proyecto para poder optimizar el proceso acuícola, se pretende extraer información de vídeos 360° de una jaula en la que hay lubinas. Para poder detectar las lubinas se ha utilizado una red neuronal convolucional (CNN), YOLOv8, y se ha desarrollado una aplicación que usando el modelo realice las detecciones y a la salida ofrezca un fichero de texto en el que muestre las etiquetas de las detecciones en cada uno de los vídeos, asociando cada una a un determinado fotograma. De este modo, quedará todo preparado para que este trabajo se pueda continuar con el análisis del comportamiento de las lubinas ante diferentes estímulos, ya sea el movimiento de todo el grupo de lubinas a unas zonas u otras, analizar el momento en el que están comiendo, en los que están interactuando entre ellas, y otras situaciones que se puedan producir. El objetivo es que en un futuro se pueda realizar un análisis mucho más detallado de lo que ocurre para garantizar un mejor aprovechamiento de recursos y un mejor cuidado de los animales, pero para ello primero se debe conseguir la detección de las lubinas y este es el aspecto principal en el que se centra este proyecto.

1.2 Objetivos técnicos y académicos

Los objetivos de este proyecto fin de carrera son, desde el punto de vista técnico:

- Desarrollar una aplicación capaz de reproducir los vídeos 360°, mientras que el modelo entrenado de YOLOv8 es capaz de realizar detecciones.

- Optimizar la detección de los peces hasta conseguir un valor de IoU que supere el 70% en las imágenes de test.
- Generar información que sirva para analizar el comportamiento de las lubinas.

Desde el punto de vista académico, el proyectista adquiere las siguientes competencias y habilidades:

- Adquirir los conocimientos para el uso de técnicas basadas en Deep Learning, tales como las redes neuronales.
- Profundizar en el entendimiento de los diferentes hiperparámetros de entrenamiento para poder mejorar los resultados. Esto incluye la capacidad de optimizar y validar los modelos obtenidos.
- Conocer y profundizar las diferentes técnicas posibles para desarrollo de aplicaciones de reproducción de vídeo que puedan ofrecer una visión 360°, integrando la reproducción del vídeo con la detección de las lubinas.
- Realizar el etiquetado de imágenes con CVAT correctamente para que los entrenamientos obtengan resultados mucho mejores y así reducir falsos positivos y falsos negativos.

1.3 Estructura del resto de la memoria

En el resto de la memoria, se tratará sobre el marco tecnológico en el que se va a plantear y desarrollar la solución, para después detallar todo lo relacionado con los requisitos e impedimentos que se han encontrado para aplicar lo principalmente propuesto.

Más adelante, se va a exponer al detalle la forma en la que se ha abordado la solución y todas las técnicas que se han empleado para el desarrollo del proyecto teniendo en cuenta los objetivos definidos. Una vez que se explican todas las tecnologías, la siguiente fase es indicar la parte de validación de resultados comentando las razones de cualquier cambio que se haya producido a la hora de trabajar y exponiendo la evolución del proyecto por fases.

Para terminar, se redactarán las partes de presupuesto necesario para la realización de esta investigación, junto con los trabajos futuros y los ODS (Objetivos de Desarrollo Sostenible) que cumple para garantizar la viabilidad medioambiental.

2. Marco tecnológico

En este apartado se presentarán las principales tecnologías utilizadas para el conteo de organismos acuáticos en acuicultura, desde métodos básicos como el conteo manual hasta soluciones avanzadas como las redes neuronales convolucionales (CNN). Se analizarán las ventajas y desventajas de cada enfoque, considerando aspectos como la precisión, los costos y el impacto en el bienestar de las especies. Además, se explicará el papel del aprendizaje automático en este contexto, incluyendo etapas como la recopilación y preparación de datos, el diseño de características y los diferentes tipos de aprendizaje (supervisado, no supervisado, semisupervisado y de refuerzo). Se pretender mostrar cómo estas técnicas realizan las detecciones y el conteo, para la gestión de recursos y el cuidado de los organismos.

Por último, se profundizará en el uso de redes neuronales convolucionales, con especial atención a la tecnología YOLOv8. Se explicará su funcionamiento, sus ventajas frente a otros sistemas y su aplicación en acuicultura, con ejemplos como el conteo de peces y el análisis de comportamientos anómalos. Estas herramientas están transformando la industria, haciendo la gestión más sostenible y precisa.

2.1 Tecnologías empleadas para el conteo en el contexto acuícola

El conteo de especies acuáticas es fundamental para lograr una mayor eficiencia en los recursos necesarios. Se debe tener en cuenta que antes de realizar cualquier análisis sobre los peces, lo primero es ser capaz de detectarlos. Para ello, en este apartado se van a exponer algunas de las alternativas más importantes empleadas en el conteo de especies, mostrando los puntos fuertes y débiles de cada una de ellas.

2.1.1 Conteo manual

Esta técnica de conteo es la más rudimentaria de todas las opciones que se van a exponer y también la alternativa más laboriosa, esto es debido a que requiere del conteo individual de cada uno de los peces, lo cual puede ser razonable para entornos con un número reducido de ejemplares; sin embargo, en caso de ser un número elevado hace que sea una opción completamente inviable. Además, el conteo manual implica que se debe tener un contacto destructivo, que va a afectar al bienestar y al estado de salud de los peces. [5]

2.1.2 Conteo basado en tecnología de sensores

En este caso, se realiza el proceso de conteo mediante el uso de sistemas infrarrojos, que utilizan ondas electromagnéticas con longitudes de onda de entre 760 nm y 1 mm. Se caracteriza por requerir de una estructura similar a un túnel para limitar el movimiento de los peces y poder contarlos cuando pasan a través del conducto. En este caso, a diferencia del conteo manual no se debe manipular a los peces, lo cual no interferirá en su estado de salud. Sin embargo, cuenta con una gran desventaja y es que su rendimiento se ve gravemente afectado por la profundidad y el estado del agua, ya que depende del grado de turbidez para

tener un funcionamiento correcto o de lo contrario esta tecnología no funcionará como se espera. [6]

Además de los factores ambientales, la precisión de los sistemas infrarrojos es susceptible a errores, debidos principalmente a confusiones que se pueden producir al acceder dos o más peces al mismo tiempo o por causa de la velocidad del organismo en cuestión. A pesar de todo, cuenta con una gran ventaja y es la capacidad de poder funcionar en la oscuridad y la posibilidad de resolver mediante software el conteo falso de múltiples elementos, utilizando algoritmos de seguimiento de múltiples objetos. [6]

2.1.3 Conteo basado en tecnología de visión artificial

El conteo de peces basado en visión por computadora se refiere al uso de cámaras y algoritmos de inteligencia artificial para contar todo tipo de objetos o entidades. Es uno de los métodos preferidos en la actualidad no solo por ser eficiente y de bajo costo, sino también por no ser invasivo, lo cual hace que sea una opción factible para este caso de uso. A pesar de todo, la mayor complejidad es el propio entorno submarino por las cambiantes condiciones de luz, el número de peces y la proximidad entre ellos. Todo ello, puede suponer un desafío para su recuento preciso. Dentro de la visión artificial debemos distinguir dos tipos que son el **conteo mediante imágenes** o el **conteo mediante vídeos**. [6] [7]

- **Conteo mediante imágenes.** Se pretende detectar todos los peces de un área geográfica. En un primer momento, se apostó por la utilización de las redes neuronales artificiales, las cuales mostraron un gran potencial para el conteo de peces, pero solo era aplicable a entornos reducidos y carecía de escalabilidad. Los problemas no solo se reducían a estos, sino también los relacionados con la luz y el ruido, lo cual supuso un gran problema ya que los resultados en un entorno acuícola real eran muy inferiores a los obtenidos en un entorno de laboratorio. Todo esto, llevó a desarrollar métodos de segmentación, cuyo objetivo era corregir la iluminación no uniforme y el ruido, pero no se encontró solución a las sombras producidas por los peces o la propia refracción del agua, lo cual disminuía la eficacia de estos métodos. [6]

El cambio se produjo con la introducción de técnicas de Deep Learning, ya que, gracias a la gran disponibilidad de diversos conjuntos de datos, los modelos de Deep Learning, como las redes convolucionales neuronales (CNN) son capaces de obtener una alta precisión en el recuento de peces de diferentes tamaños. [6]

- **Conteo basado en vídeo.** Se ofrece un sistema confiable para el conteo de organismos acuáticos; sin embargo, los grandes desafíos de este sistema se relacionan con la contaminación del agua que afectan a la calidad del vídeo y en consecuencia a la capacidad de detección y segmentación de objetos. Para mitigar estos problemas, se han desarrollado algoritmos de mejora de imágenes que aumentan el contraste y que restauran los colores. [6]

En ambos casos, a pesar de las grandes limitaciones que ofrecen ambos sistemas, éstos son muy utilizados en acuicultura y a medida que vayan evolucionando se lograrán

mejores detecciones en todas las situaciones, para lo cual se espera un aumento en la capacidad de generalización del modelo e incremento de la precisión del conteo. Para el caso de uso que se plantea, se considera que el conteo mediante imágenes es la mejor alternativa por los buenos resultados obtenidos, y esto está unido a un precio más que aceptable teniendo en cuenta el costo de los dispositivos y de personal cualificado que se requiere en las otras variantes. [6]

2.1.4 Conteo mediante sistemas acústicos

La tecnología acústica se emplea para el conteo de peces mediante imágenes acústicas o por métodos hidroacústicos. Se debe tener en cuenta que las imágenes submarinas sufren limitaciones debido a la atenuación de la luz, que da como resultado imágenes borrosas. Como punto positivo, las ondas sonoras pueden viajar lejos sin tener una gran atenuación. [6]

- **Métodos de conteo a partir de imágenes acústicas.** Se emplean sonares de imágenes multihaz con el objetivo de monitorear los peces migratorios en ríos. Estos dispositivos ofrecen salidas de alta resolución, incluso en condiciones nocturnas o en aguas turbias. Sin embargo, en situaciones complejas o muy oscuras, el ruido submarino puede interferir con la precisión. Además, el alto costo del sonar y la necesidad de contar con personal especializado para su operación hacen que esta opción no sea la más viable. [6]
- **Métodos de conteo hidroacústicos.** Son populares por su carácter no invasivo y su capacidad de proporcionar datos de manera relativamente sencilla. Su funcionamiento se basa en la emisión de una onda acústica, la cual se propaga a través del agua y se encuentra con diferentes objetos a su paso. La profundidad se mide según la diferencia entre el tiempo que transcurre entre la emisión de la onda y su regreso al punto de partida. Además, se debe analizar la intensidad y la forma de la señal recibida para determinar el número y la distribución de los elementos. [6]

Sin embargo, las desventajas de esta técnica incluyen la reverberación que puede producirse, especialmente en entornos pequeños, y la existencia de un área ciega en la ecosonda que impide analizar ciertas zonas. Por tanto, aunque la técnica no sea excesivamente complicada en principio, su implementación efectiva requiere equipos costosos y personal con experiencia técnica para su uso óptimo. [6]

2.2 Aprendizaje automático

El concepto de aprendizaje automático hace referencia a uno de los campos de la inteligencia artificial, que ofrece la capacidad de identificación de patrones para posteriormente realizar detecciones, gracias al aprovechamiento de algoritmos como las redes neuronales. Estas redes, inspiradas en el cerebro humano, están formadas por neuronas artificiales conectadas en capas que procesan datos para identificar patrones y aprender a partir de ellos. En el aprendizaje profundo, que es un tipo avanzado de aprendizaje automático, las redes neuronales poseen varias capas, lo que les permite identificar todo tipo de características de los datos de manera autónoma. Esto las hace ideales para tareas como la identificación,

clasificación por tamaño, análisis de comportamientos y la detección de tendencias en función de estímulos, como el estrés, enfermedades o hambre. La etapa de aprendizaje de estas redes consiste en entrenarlas con datos etiquetados, ajustando los parámetros para mejorar su precisión. Una vez entrenadas, la etapa de inferencia permite a las redes aplicar lo aprendido para hacer predicciones sobre nuevos datos. Esta capacidad es fundamental para optimizar procesos y mejorar la calidad de vida de los organismos en acuicultura, permitiendo un crecimiento más sano y una gestión más eficiente de los recursos. [8] [9] [10] [11]

2.2.1 ¿Cómo funciona el aprendizaje automático?

Una vez conocemos de qué se trata el aprendizaje automático, pasamos a ver cómo funciona. Existen cinco pasos clave:

- 1. Recopilación y preparación de datos.** Se basa en la obtención de toda la información que sea necesaria para abordar el problema a resolver. Una vez que se ha hecho la recopilación, se preprocesan los datos recopilados con la finalidad de conseguir el formato adecuado para el posterior análisis. En el caso de estudio de detección de lubinas, se deben conseguir capturas de las lubinas en diferentes escenarios y realizar el etiquetado para preparar los conjuntos de datos. [9]
- 2. Diseño de características.** Se deben extraer todos los valores significativos del dataset para el caso de uso a aplicar, teniendo en cuenta la captura de patrones y de relaciones. En el caso de uso acuícola, se basa en realizar aumento de datos usando técnicas como rotación, ajuste de brillo y mejorar la capacidad de generalización del modelo en cualquier condición. [9]
- 3. Selección y entrenamiento del modelo.** Se debe elegir un modelo de aprendizaje automático adecuado, en función de los datos disponibles y requerimientos del problema a resolver y una vez elegido, se entrena con los datos preprocesados ya sea aprendizaje supervisado o no supervisado, en función de la naturaleza del problema. En cuanto a su aplicación en el caso de uso del proyecto, se utilizan técnicas de aprendizaje supervisado donde los datos se han etiquetado previamente, dividiéndose en los conjuntos de entrenamiento, validación y test. Además, se deben elegir los hiperparámetros como el tamaño de lote (`batch_size`), tasa de aprendizaje y número de épocas. [9]
- 4. Evaluación y validación del modelo.** Se analizan las métricas de rendimiento del modelo para evaluar su efectividad y fiabilidad, de ahí se puede saber qué precisión tiene el modelo al realizar detecciones. En el caso de uso propio, se evalúan métricas como mAP o puntuación F1 para medir la efectividad de las detecciones. [9]
- 5. Implementación y supervisión.** Una vez que se obtiene el modelo más satisfactorio, se debe probar su uso en un entorno real con el objetivo de poner a prueba su rendimiento, para ir mejorándolo en caso de ser necesario. En el proyecto, se aplicó este paso mediante una aplicación que cargaba el vídeo 360° y se hace uso del modelo para realizar predicciones. [9]

2.2.2 Tipos de aprendizaje automático

El aprendizaje automático, también conocido como aprendizaje máquina, se caracteriza por la capacidad de los sistemas a aprender a partir de la información que se les ofrece, reconocer patrones y realizar tomas de decisiones con una reducida intervención humana. [9] Los tipos que sobresalen son:

- **Aprendizaje supervisado.** En este caso, el modelo se entrena usando datos etiquetados, lo que supone que cada entrada de datos se asocia con una etiqueta conocida. Los algoritmos aprenden mediante el conjunto de entrenamiento y aplican los conocimientos en un conjunto de datos que no han sido etiquetados. Es muy usado en tareas de regresión, redes neuronales y algoritmos de clasificación como máquinas de vectores de soporte (SVM). Su función es el reconocimiento de imágenes y el análisis predictivo, entre otros. [8] [12]

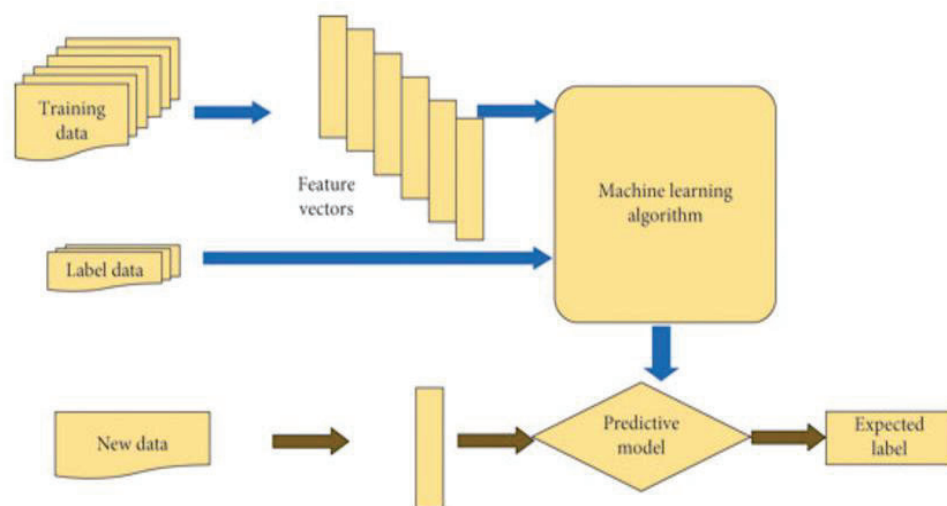


Figura 2 Diagrama del Aprendizaje supervisado [13]

- **Aprendizaje no supervisado.** En este caso, los algoritmos carecen de conocimiento previo; por tanto, poseen un conjunto de información que es desconocida. Se emplean algoritmos de aprendizaje automático para que el modelo sea capaz de extraer la información. Destacan los modelos de mezcla de gaussiana (GMM) y el análisis de componentes principales (PCA). Es muy usado por la comunidad científica a la hora de identificar asociaciones entre objetos en grandes bases de datos, para facilitar su visualización y reducir la dimensionalidad. [8] [12]

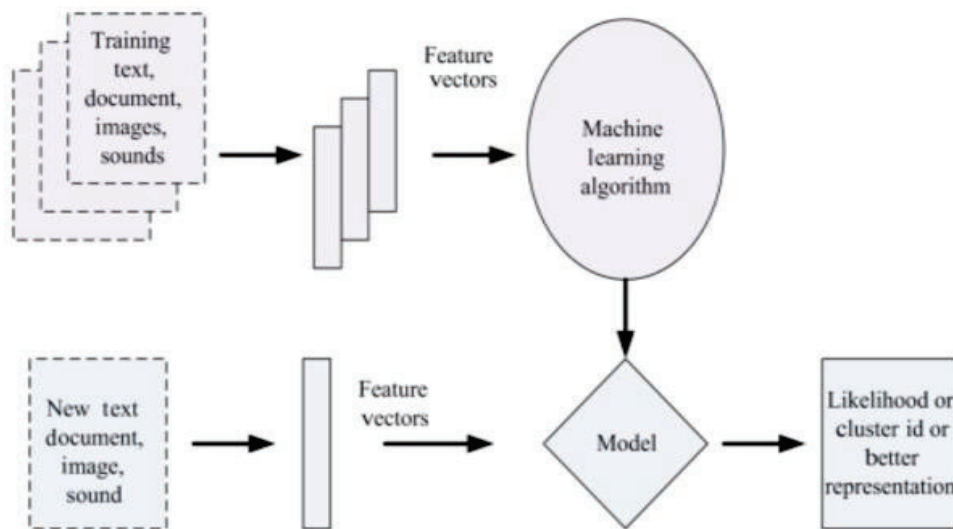


Figura 3 Diagrama del aprendizaje no supervisado [14]

- **Aprendizaje semisupervisado.** Se trata de un algoritmo que combina los dos tipos anteriores, con lo que utiliza datos etiquetados y no etiquetados. Es útil especialmente en situaciones en las que la obtención de los datos etiquetados o bien requiere demasiado tiempo o es excesivamente costoso. Utilizamos estas técnicas cuando una pequeña parte de los datos está etiquetada y el resto no lo está. En la actualidad, se emplea principalmente para el reconocimiento de voz, el procesamiento de lenguaje natural, clasificación de imágenes y reconocimiento de objetos. [15]
- **Aprendizaje automático de refuerzo.** Se trata de un método que permite a un algoritmo aprender interactuando con su entorno ejecutando acciones e identificando errores. Se caracteriza por mejorar su rendimiento al recibir recompensas como retroalimentación, para aprender un patrón específico. Actualmente, se usa para el desarrollo de vehículos autónomos. Es importante saber que el modelo cada vez que recibe nuevos datos, los integra como nuevos conocimientos y así de manera progresiva mejora su desempeño con el tiempo. [15]

2.3 Redes Neuronales: origen y evolución

Una red neuronal es un modelo de aprendizaje automático. Estas redes se basan en el funcionamiento del cerebro humano al emplear neuronas para procesar datos. Dentro del aprendizaje automático se encuentra el aprendizaje profundo, que se refiere al uso de capas intermedias (capas ocultas), que permiten extraer patrones avanzados del conjunto de datos de entrada. Este enfoque permite la creación de sistemas adaptables que ayudan a los ordenadores a aprender de sus fallos y a mejorar de manera progresiva. [16]

Este campo nació gracias a la publicación de Warren S. McCulloch y Walter Pitts en el año 1943, en el cual pretendieron comprender el funcionamiento del cerebro humano e hicieron una analogía de la toma de decisiones de las neuronas cerebrales con los estados lógicos 0 y 1, que es la función escalón. De hecho, todo esto serviría más adelante de base para la creación por parte de Frank Rosenblatt del perceptrón, el cual sirvió para simular el comportamiento de las neuronas humanas, al procesar datos de entrada mediante la

multiplicación por coeficientes de peso, y la suma de estos productos se evalúa frente a un umbral y si el resultado supera el umbral la neurona se activa, y de lo contrario, permanecerá inactiva. Se utiliza fundamentalmente para la clasificación de datos de forma binaria y en este proceso los pesos se ajustan, lo cual permite que el modelo pueda aprender de los errores y mejorar su precisión. Esta técnica es fundamental para afinar el rendimiento y asegurar que la salida predicha se acerque al valor real. En la siguiente captura, se aprecia el diagrama del perceptrón por bloques. [17] [18]

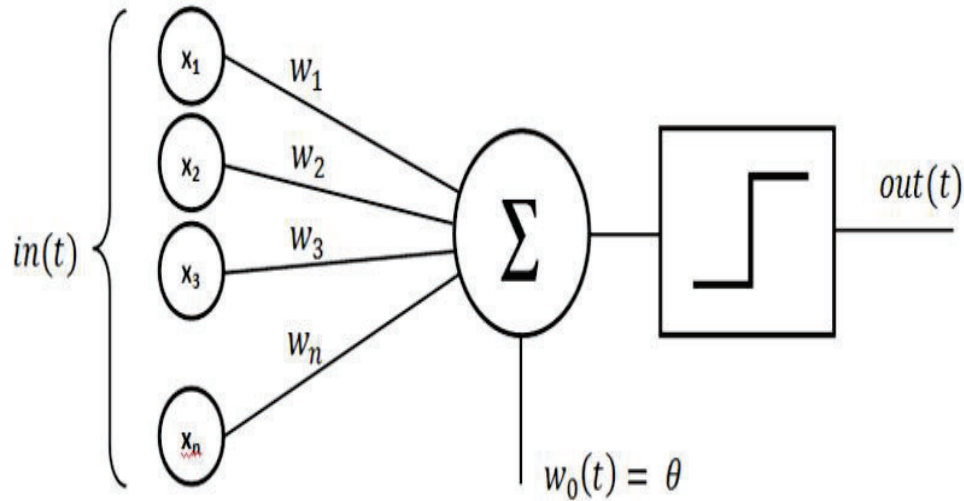


Figura 4 Esquema visual del Perceptrón [18]

En cuanto al diagrama de la figura anterior, se tiene un nodo o neurona con una serie de **N** entradas, cada una de ellas con un peso determinado **W** y un valor **X**. En último lugar, **θ** se refiere al bias (b), que ajusta la salida de la neurona, incluso si las entradas son cero. Con toda esta información, el nodo calcula el resultado para después determinar si debe o no activarse la neurona. Una vez hecho el cálculo, se procede a la comparación entre el resultado predicho y el conocido y en caso de haber diferencia, el error se retropropaga (Back-propagate) para ajustar los pesos. [18] En la siguiente figura, se muestra la fórmula matemática que emplea durante este proceso:

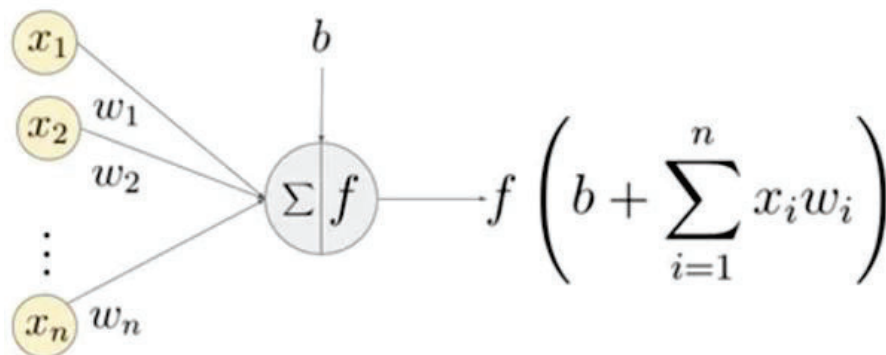


Figura 5 Fórmula matemática del perceptrón [19]

En esta fórmula $\sum_{i=1}^n x_i w_i$, hace referencia a la suma ponderada de las entradas multiplicadas por sus pesos. f hace referencia a la función de activación (escalón de Heaviside), que indica si la neurona se activa o no. Devuelve 0 si el argumento es negativo y 1 en caso contrario. [20]

Una vez que se logró desarrollar el funcionamiento correcto para una neurona, se comenzó a elaborar un proceso de interconexión de neuronas, lo cual supondría el origen de las redes neuronales de tipo MLP (MultiLayer Perceptron), las cuales se caracterizan por tener varias capas adicionales, además de las de entrada y salida. En la siguiente figura, se muestra un ejemplo de una red de este tipo: [21]

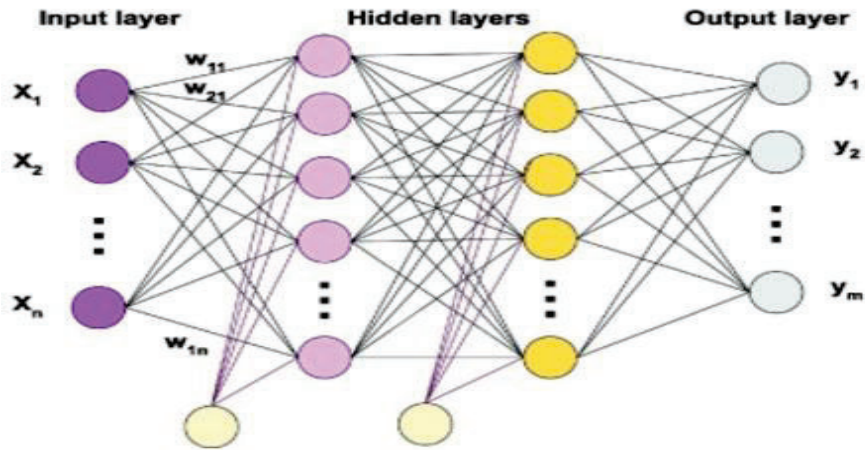


Figura 6 Ejemplo de Red MLP (MultiLayer Perceptron) [21]

Tal y como se puede ver en la Figura 6, una red neuronal MLP se organiza en una jerarquía de capas: una capa de entrada, una o más capas ocultas intermedias y una capa de salida. Las conexiones entre neuronas tienen pesos que representan la memoria del sistema a largo plazo, y el aprendizaje ocurre al ajustar repetidamente estos pesos. Este proceso se repite y se ajusta hasta que la diferencia entre la salida real y la esperada se minimiza, permitiendo que la red aprenda y optimice su rendimiento. [22] En la Figura 7, se va a mostrar el procedimiento que realiza cada neurona de manera esquemática:

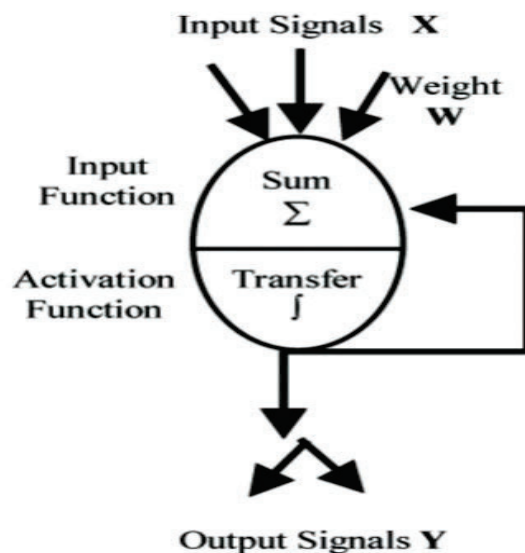


Figura 7 Diagrama de funcionamiento de una neurona artificial [22]

Ahora, se pasa al proceso de entrenamiento. Durante esta etapa, se utilizan algoritmos de optimización, como el descenso por gradiente, con el fin de ajustar los pesos de las conexiones en la red neuronal. Este ajuste se realiza minimizando una función de pérdida denominado

MSE (Mean Squared Error), que mide la discrepancia entre los valores previstos por la red ($\hat{y}_i$) y los valores reales (y_i) en un conjunto de datos. [23] [24] [25] Esta función de pérdida es ampliamente utilizada en problemas de regresión, se conoce como el error cuadrático medio, expresado como:

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

Esta función cuantifica el error promedio al elevar al cuadrado las diferencias entre las predicciones y los valores reales, facilitando la retroalimentación necesaria para ajustar los pesos y mejorar la precisión del modelo. Además, los pesos (w) se actualizan a lo largo del entrenamiento siguiendo la regla del descenso por gradiente [23] [24]. La fórmula del descenso por gradiente es la siguiente, donde η es la tasa de aprendizaje:

$$w_{i+1} = w_i - \eta \frac{\partial L}{\partial w_i}$$

Ahora, se va a explicar una imagen que muestra el proceso de entrenamiento de una red neuronal. La fase de propagación hacia adelante (Feedforward) lleva los datos a través de la red, mientras que la retropropagación o Back-propagate ajusta los pesos de las conexiones basándose en el error calculado, utilizando algoritmos como el descenso por gradiente, mencionado previamente. Las optimizaciones en las capas de la red permiten que el modelo aprenda y mejore continuamente su precisión. [22]

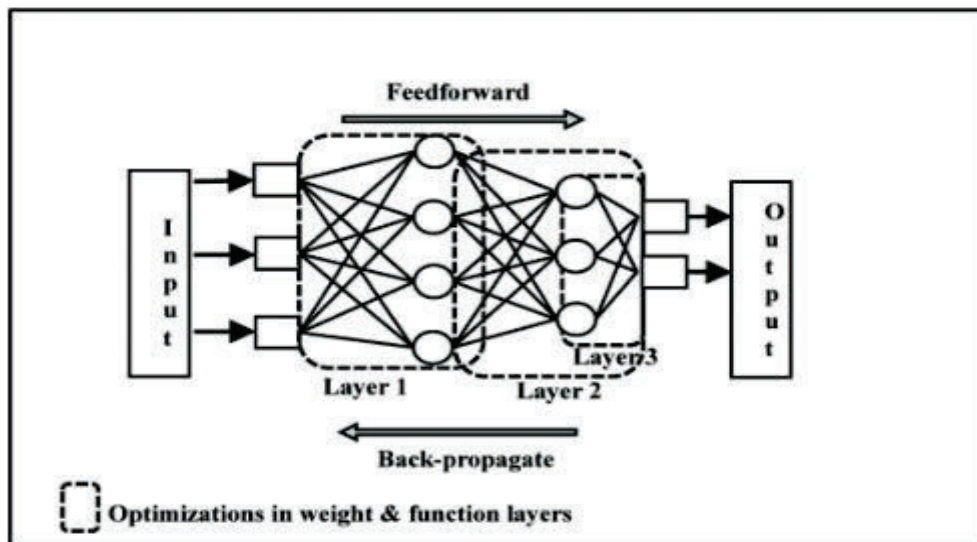


Figura 8 Proceso de aprendizaje de una red neuronal [22]

2.4 Detección de Objetos usando redes neuronales

La utilización de redes neuronales para identificar la ubicación correcta de los objetos y clasificarlos mediante cajas delimitadoras es una de las aplicaciones más destacadas, lo cual puede ser utilizado para múltiples campos, ya sea en el ámbito médico o en el ámbito de la seguridad, entre otros.

2.4.1 Tipos de redes neuronales

Existen varios tipos de redes neuronales adecuadas para distintos objetivos y son las siguientes: [26]

- **Redes neuronales FeedForward (FFNN).**

Se caracterizan por ser redes neuronales con grandes facilidades de implementación, tienen una capa de entrada y una de salida y algunas capas opcionales entre medias. Los datos de entrada viajan a través de la red hasta la salida, con lo cual no hay bucles de retroalimentación. Son eficientes para el manejo de datos con mucho ruido y por su relativa facilidad de implementación.

- **Redes neuronales recurrentes (RNN).**

Una red neuronal recurrente (RNN) está diseñada para almacenar la salida de una capa y reutilizarla como entrada para otra neurona, lo que le permite influir en la predicción de la siguiente capa. La estructura generalmente comienza con una red de retroalimentación, seguida por una capa de RNN, la cual incluye una función de memoria que recuerda información del paso de tiempo anterior.

- **Redes neuronales con función de base radial (RBF).**

Una red de función de base radial (RBF) tiene una capa de entrada, una de neuronas RBF y una capa de salida con nodos para cada categoría. Durante la clasificación, las neuronas comparan la entrada con prototipos de datos de entrenamiento. Cada una calcula la distancia euclídea la entrada, determinando la categoría según la cercanía. Las neuronas emiten un valor de similitud entre 0 y 1, siendo la respuesta máxima cuando la entrada coincide con el prototipo y decrece exponencialmente a medida que la distancia aumenta, siguiendo un típico patrón de campana.

- **Redes neuronales convolucionales (CNN).**

Una red neuronal convolucional (CNN) es eficaz para tareas como procesamiento de imágenes, visión por computadora y reconocimiento de voz, gracias a su estructura tridimensional de neuronas. Se compone de capas convolucionales que usan filtros para extraer características como bordes y texturas, seguidas de capas de agrupación que reducen la dimensión espacial para simplificar la información y disminuir la carga computacional. Finalmente, las capas completamente conectadas clasifican las imágenes usando funciones de activación como softmax. Las CNN destacan por su eficiencia en aprendizaje profundo con menos parámetros, aunque pueden ser complejas de diseñar y su velocidad depende del número de capas.

Este tipo es idóneo para el caso de uso de este proyecto al centrarnos en tareas de clasificación y detección de objetos. A lo largo de los últimos años se han optimizado de manera considerable, pero es cierto que requieren de alto coste computacional. Aun así, debido a los buenos resultados obtenidos y rapidez de adaptación a su uso por personal no cualificado, se considera la mejor opción. [27]

- **Red neuronal modular (MNN).**

Una red neuronal modular se compone de múltiples redes independientes que realizan subtarefas sin interactuar ni intercambiar señales durante el cálculo. Esta estructura

permite dividir un proceso computacional grande en componentes más manejables, lo que acelera significativamente el procesamiento. Las MNN se aplican en áreas como la predicción del mercado de valores, reconocimiento adaptativo de caracteres y compresión de datos complejos. Entre sus ventajas se encuentran la eficiencia, el entrenamiento independiente y la robustez. Sin embargo, enfrentan dificultades con objetos en movimiento que pueden afectar su rendimiento, lo cual indica que no es el más recomendable para el caso de uso de este proyecto.

2.5 Redes neuronales convolucionales (CNN)

Una red neuronal convolucional, es una variante de Deep Learning que se diseña para tareas como clasificación, reconocimiento de objetos, segmentación de imágenes y detección. Las capas convolucionales permiten identificar y extraer patrones sobre los datos a analizar. Además, otra característica relevante es la conexión entre las neuronas que, en este caso, no se conectan todas entre sí, sino que solo están conectadas a una región local del volumen de entrada gracias a la operación de convolución, lo cual permite mayor eficiencia. [28]

Además, según se aprecia en la siguiente figura, las funciones se dividen en dos fases, que son extracción de la información (Feature Learning de la Figura 9) y posterior clasificación de las imágenes (Classification de la Figura 9).

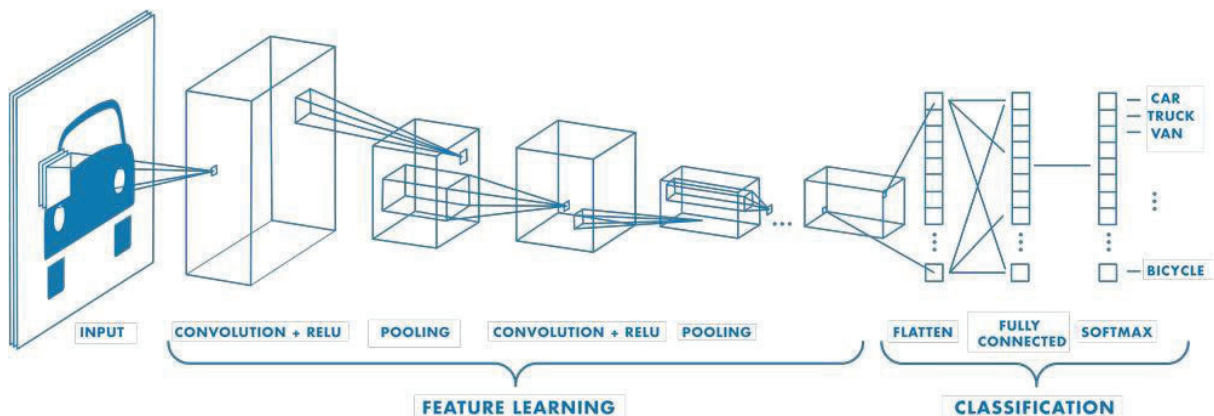


Figura 9 Estructura básica de una red neuronal convolucional (CNN) [29]

El proceso de convolución es el primer paso esencial en una red neuronal convolucional (CNN). Como su nombre sugiere, la convolución es una operación matemática que implica la aplicación de una función de ventana deslizante a una matriz de píxeles que representa la imagen de entrada. Esta función de ventana deslizante se conoce como núcleo, filtro o kernel. Los filtros son pequeñas cuadrículas que se mueven sobre la imagen e identifican patrones específicos, como líneas, bordes o curvas, en distintas regiones de la imagen. [28] [30]

En una capa convolucional, se aplican múltiples filtros de igual tamaño, cada uno diseñado para reconocer un tipo específico de patrón o característica en la imagen. Por ejemplo, un filtro podría estar ajustado para identificar líneas rectas, mientras que otro podría detectar curvas o texturas más complejas. El resultado de aplicar estos filtros es la generación de un mapa de características, que refleja cómo y dónde se encuentran los patrones identificados

en la imagen de entrada. La operación de convolución en sí implica multiplicar los valores del núcleo por los valores de la región de la imagen sobre la que se desplaza y sumar los resultados. Esto se repite en todas las posiciones posibles de la imagen, generando un mapa de características que destaca las áreas donde el patrón buscado se encuentra presente. Una ventaja importante de usar convoluciones es que el mismo filtro se puede aplicar en toda la imagen, lo que supone una reducción considerable del número de conexiones y parámetros a entrenar, con respecto a una red totalmente conectada. [28] [30]

En la siguiente figura, se aprecia un ejemplo del proceso de convolución:

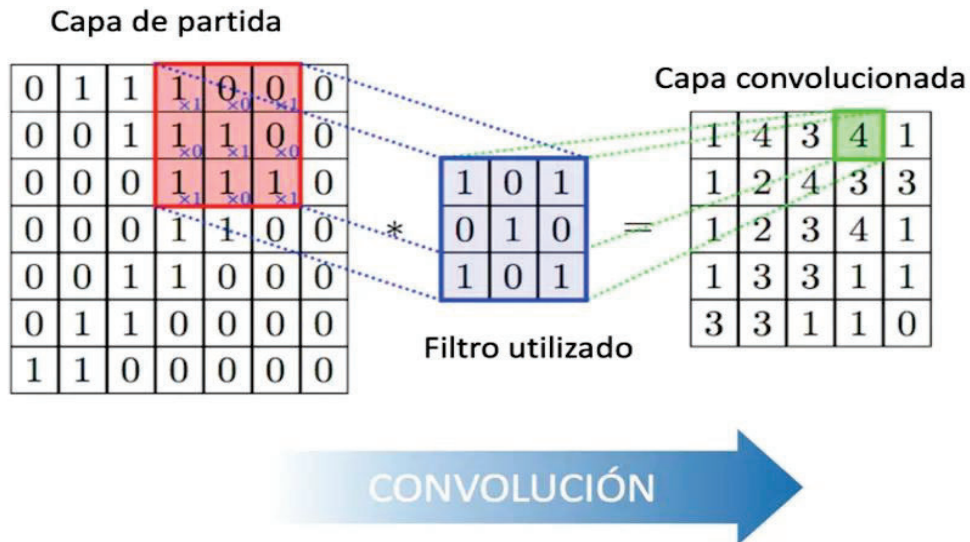


Figura 10 Proceso de convolución en una CNN [30]

Después de aplicar la convolución, los mapas de características pasan por una función de activación para introducir no linealidad. La función de activación más recomendada es Rectified Linear Unit (ReLU). Es eficiente de calcular y evita el problema del desvanecimiento del gradiente, permitiendo actualizaciones de peso más efectivas. Además, funciona bien en redes profundas al no sufrir saturación, a diferencia de otras funciones de activación. Se define con la siguiente expresión: [31]

$$f(x) = \max(0, x)$$

Otro aspecto fundamental a tener en cuenta es el Pooling. Tal y como se aprecia en la Figura 11, las capas de Pooling en las redes neuronales convolucionales reducen la dimensión de la entrada en un área específica, resumiéndola en un solo valor. Este proceso de submuestreo permite concentrar la información de manera eficiente, lo que reduce el uso de memoria sin perder características clave. Gracias a la agrupación, las redes obtienen una cierta invariancia a rotaciones y traslaciones, mejorando su capacidad de detectar objetos que se encuentran en posiciones inusuales dentro de una imagen. Además, este proceso permite crear redes más compactas que pueden ser alojadas más fácilmente en la memoria de la tarjeta gráfica (GPU). No obstante, si el área de agrupación es demasiado grande, existe el riesgo de perder demasiada información, lo que puede afectar la precisión del modelo en la predicción.

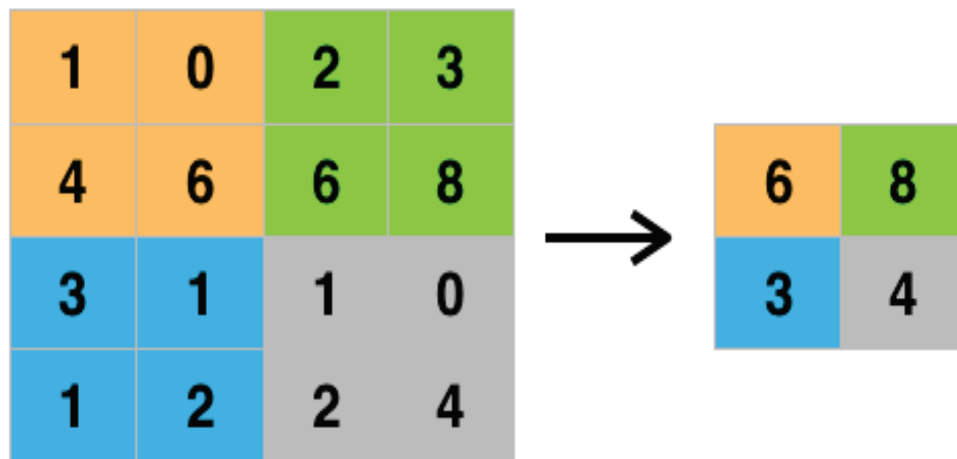


Figura 11 Ejemplo de funcionamiento de Pooling [30]

Las capas de Pooling ayudan a disminuir el uso de memoria durante el proceso de entrenamiento y esto contribuye a mitigar el riesgo de overfitting o sobreajuste. Entre las operaciones más comunes se encuentran el **max-pooling**, que selecciona el valor máximo dentro del mapa de características; el **sum-pooling**, que suma todos los valores; y el **average-pooling**, que calcula la media de dichos valores. Estas operaciones permiten que la red comprima la información de manera efectiva, reduciendo la dimensión del mapa de características tras cada aplicación de la función de agrupamiento. Al finalizar esta etapa, la última capa de agrupamiento aplanar el mapa de características, de forma que pueda ser procesado por las capas totalmente conectadas. [27] [28]

Tras el paso por las convoluciones y el pooling, las capas totalmente conectadas reciben la matriz aplanada generada. Estas capas aplican funciones de activación como ReLU, que introducen no linealidades en el modelo para una representación más compleja de los datos. Finalmente, una capa softmax transforma las salidas en probabilidades que suman 1, facilitando la selección de la categoría final. [27] [28]

En la figura 12, se muestra un ejemplo sobre cómo opera una red neuronal convolucional (CNN). La imagen de entrada tiene un tamaño inicial de 32x32 píxeles y pasa primero por un filtro de convolución de 5x5, encargado de extraer patrones básicos como bordes o texturas. Este proceso reduce la resolución a 28x28 píxeles. Posteriormente, se realiza un submuestreo, que comprime aún más la información clave, dejando la imagen en un tamaño de 14x14. En la etapa final, las capas completamente conectadas procesan las características obtenidas para clasificar la imagen en una categoría específica (fase de "Classification").

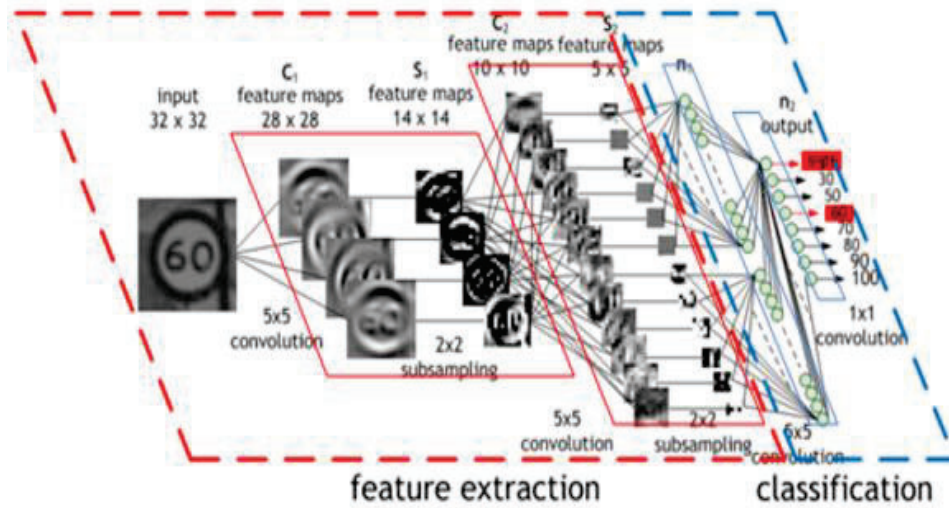


Figura 12 Ejemplo de CNN con proceso de convolución y posterior submuestreo [32]

2.6 YOLOv8

La detección de objetos es una habilidad natural en los humanos que permite identificar y comprender objetos de manera rápida y precisa. Crear algoritmos que imiten esta capacidad podría facilitar tareas complejas, como la conducción autónoma sin sensores avanzados, mejorar los dispositivos de asistencia y potenciar los sistemas robóticos. Tradicionalmente, los métodos de detección, como los modelos de partes deformables (DPM), usan clasificadores aplicados a múltiples áreas de la imagen, mientras que enfoques más recientes como R-CNN emplean propuestas de región y clasificadores, lo cual es complejo y difícil de optimizar. [33]

YOLO (You Only Look Once) redefine la detección de objetos como un problema de regresión, procesando la imagen completa con una red convolucional en un solo paso para predecir cuadros delimitadores y clases. Esto le permite razonar sobre la imagen de forma global, reduciendo errores de detección de fondo y superando a métodos como Fast R-CNN. Además, YOLO es extremadamente rápido, procesando hasta 45 fotogramas por segundo (fps) en su versión estándar y más de 150 fps en su versión rápida, lo que lo hace ideal para análisis en tiempo real. Aunque tiene limitaciones en la precisión de la localización de objetos pequeños, su equilibrio entre velocidad y rendimiento lo posiciona como una de las mejores alternativas en la detección de objetos. La disponibilidad de su código y modelos preentrenados fomenta la colaboración y mejora constante por parte de la comunidad. [33]

2.6.1 Visión por computadora con YOLOv8

YOLOv8 permite realizar una serie de tareas y las principales son las siguientes [34] :

- **Detección.** Identifica objetos en imágenes o vídeos, mostrando las cajas delimitadoras alrededor de ellos. Se utiliza principalmente para reconocimiento de elementos con precisión en imágenes y/o vídeos. Además, no solo se centra en la predicción de elementos rectos, sino también de la detección de elementos que están rotados, lo cual es útil en escenas complejas.

- **Segmentación.** Divide una imagen en diferentes regiones, asignando una etiqueta a cada zona. Su uso radica fundamentalmente en el ámbito médico, donde se requiere de precisión para la separación de zonas específicas.
- **Clasificación.** Asigna una imagen completa a una categoría. Sirve para diferenciar entre cada uno de los elementos que se detectan.

En la siguiente figura, se puede apreciar un ejemplo de cada una de las funciones extraída de la propia página web de Ultralytics, que es la empresa que comercializa y facilita el uso de YOLOv8.



Figura 13 Tareas típicas de YOLOv8 [34]

A la hora de realizar los entrenamientos de un modelo es esencial dividir los datos en diferentes conjuntos para garantizar un desempeño óptimo. Generalmente, se sigue una proporción de 70% para el conjunto de entrenamiento, 15% para el conjunto de validación y 15% para el conjunto de pruebas. [35] Las funciones y características de cada uno son las siguientes:

- **Conjunto de entrenamiento:** Es la parte más significativa del dataset y se emplea para el entrenamiento del modelo, permitiéndole aprender características y patrones de las imágenes. Es crucial que este conjunto sea representativo y lo suficientemente diverso para que el modelo pueda generalizar adecuadamente. Una práctica común es aplicar técnicas de aumento de datos, como rotaciones, recortes o cambios de iluminación, para incrementar el conjunto de entrenamiento y mejorar la capacidad de generalización del modelo sin necesidad de más datos reales. La distribución de los datos se realiza de manera cuidadosa, con alrededor del 70-80% del total de datos destinado al entrenamiento, mientras que el resto se reserva para la validación y pruebas. [35]
- **Conjunto de validación:** Este conjunto tiene la función de monitorear el rendimiento del modelo durante el proceso de entrenamiento. Sirve para evaluar el comportamiento del modelo en datos que no ha manejado durante el entrenamiento, evitando el problema del sobreajuste, donde el modelo memoriza ejemplos

específicos en lugar de identificar patrones útiles. Los resultados obtenidos del conjunto de validación permiten ajustar hiperparámetros y tomar ciertas decisiones como cuándo detener el entrenamiento o si se necesita aplicar técnicas de regularización para mejorar la capacidad de generalización. [35]

- **Conjunto de pruebas:** Una vez que el modelo ha sido entrenado y validado, se utiliza el conjunto de pruebas para evaluar su rendimiento final. Este conjunto debe estar completamente separado de los datos de entrenamiento y validación para ofrecer una medición objetiva de la capacidad del modelo de generalizar en datos completamente nuevos. Es el conjunto con el que se mide la eficacia y precisión real del modelo en un entorno de producción y ayuda a prever su comportamiento en escenarios del mundo real. [35]

Además, es importante asegurarse de que la selección de los datos para cada conjunto se realice de manera aleatoria y balanceada, para evitar sesgos que puedan distorsionar los resultados. Esto, junto con un uso adecuado de técnicas de aumento de datos, contribuye a un entrenamiento robusto y a un modelo final más fiable. [35]

2.6.2 Ventajas de YOLO frente a otros sistemas de detección de objetos

YOLO (You Only Look Once) ha cambiado radicalmente la manera en que se aborda la detección de objetos, gracias a un enfoque integral que contrasta con los métodos tradicionales. Mientras que los sistemas convencionales dividen el proceso en varias etapas separadas como la extracción de características, la clasificación y la predicción de cajas delimitadoras, YOLO combina todos estos pasos en una única red neuronal. Por ejemplo, los modelos de partes deformables (DPM) utilizan un enfoque de ventanas deslizantes y procesos independientes, lo que ralentiza su desempeño. En cambio, YOLO es capaz de unificar la extracción de características, la predicción de cajas y la supresión de resultados redundantes en un solo proceso, logrando así una detección más rápida. [33]

Las R-CNN y sus variantes dependen de propuestas de regiones generadas mediante algoritmos como Selective Search y que requieren un pipeline complejo con múltiples etapas. YOLO por su parte, simplifica el proceso al proponer un número reducido de cajas por imagen y aplicar restricciones espaciales para evitar duplicados. Esto le permite realizar predicciones de manera más eficiente, en contraste con R-CNN, que puede tomar más de 40 segundos por imagen. En cambio, YOLO mantiene un rendimiento en tiempo real y con una precisión competitiva. Si bien modelos como Fast R-CNN y Faster R-CNN mejoran la velocidad al compartir cálculos y reemplazar Selective Search con redes neuronales para la propuesta de regiones, aún no logran igualar la velocidad de YOLO. Por ejemplo, Fast R-CNN sigue dependiendo de Selective Search y se limita a 0.5 fps, mientras que Faster R-CNN, aunque es más rápido, solo alcanza 18 fps en su versión más ágil, en comparación con los más de 30 fps de YOLO. Otros enfoques, como Deep Multibox, utilizan redes para predecir regiones de interés, pero requieren pasos adicionales de clasificación y validación, lo cual reduce su eficiencia. Por otro lado, YOLO es un sistema completo que predice cajas delimitadoras y probabilidades de clase de manera simultánea, optimizando el proceso de principio a fin. Un

caso similar es OverFeat, que emplea un método de ventanas deslizantes y solo accede a información local, lo que obliga a realizar un postprocesamiento intensivo para obtener detecciones coherentes. En cambio, YOLO puede considerar el contexto global de la imagen sin necesidad de etapas adicionales. [33]

Una de las características más destacadas de YOLO es su capacidad para actuar como un detector versátil, capaz de aprender y reconocer múltiples tipos de objetos al mismo tiempo, a diferencia de otros modelos que están optimizados para clases específicas como rostros o personas. Esta habilidad de generalización, junto con su enfoque simplificado, le da a YOLO una gran ventaja en términos de velocidad, precisión y adaptabilidad frente a otros sistemas que requieren pipelines segmentados y más complejos. [33]

2.6.3 Aplicaciones de YOLOv8 en acuicultura

Como se había mencionado, la acuicultura se dedica a la cría de especies acuáticas en entornos controlados. Ha ganado popularidad recientemente debido a su menor impacto en los ecosistemas marinos respecto a otras opciones tradicionales como la pesca intensiva. Para optimizar la eficiencia de este proceso, se han empezado a aplicar redes neuronales, siendo YOLOv8 una de las mejor posicionadas. A continuación, se presentarán ejemplos que ilustran sus aplicaciones en este campo:

- **Sistema de conteo de peces.** El artículo presenta un método mejorado de YOLO, que es YOLOv8n-MEMAGD, para conteo de peces en acuicultura, que supera limitaciones previas al optimizar precisión y velocidad, haciéndolo adecuado para dispositivos móviles y entornos desafiantes. [36]
- **Sistema de detección y seguimiento.** En el artículo figura el empleo de un modelo de YOLOv8 adaptado para su uso por drones, que tiene como objetivo el seguimiento de atunes para llevar a cabo prácticas de pesca sostenibles, pudiendo gestionar mucho mejor los recursos y contribuyendo a la sostenibilidad. [37]
- **Sistema de análisis de comportamientos anómalos.** En el proyecto encontrado, se hace uso de YOLOv8 para identificar comportamientos poco habituales en camarones blancos, tales como canibalismo, encorvamiento, muerte, etc. Este análisis se realiza para ver si existen problemas en el entorno acuícola. [38]

Además de estos ejemplos, también se cuentan con proyectos realizados por antiguos alumnos de la ETSIST UPM, correspondientes al grupo GAMMA.

2.7 Herramientas de despliegue de YOLOv8

En este apartado, se van a mostrar algunas de las técnicas y elementos utilizados para el entrenamiento de redes neuronales con YOLOv8. Las más destacadas son:

- **Lenguaje de programación Python.** Python es conocido por su versatilidad, facilidad de uso y eficiencia, haciéndolo ideal para el aprendizaje automático y la ciencia de datos. Su sintaxis clara y sencilla permite un desarrollo rápido y comprensible,

optimizando la productividad. Librerías como NumPy y Pandas simplifican la manipulación y análisis de datos, mientras que OpenCV es esencial para el procesamiento de imágenes. PyTorch, construido sobre Python, facilita la creación y entrenamiento de redes neuronales, aprovechando CPU y GPU para un rendimiento escalable. Su sólida comunidad ofrece soporte constante, reforzando su liderazgo en machine learning y deep learning [39]. La siguiente captura, muestra el número de puestos de trabajo demandados recientemente, lo cual es una muestra que Python es un lenguaje muy importante en nuestros días.

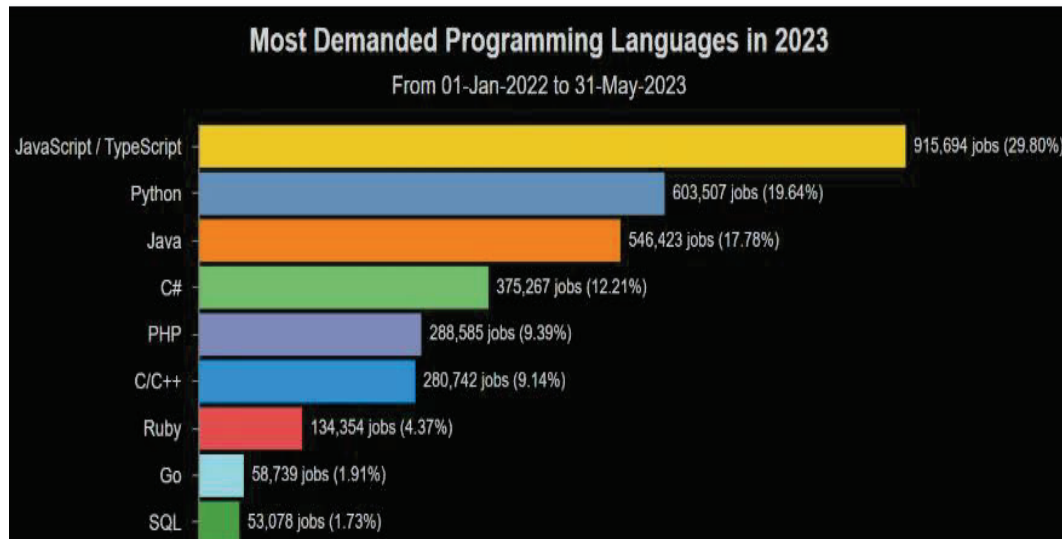


Figura 14 Ranking de lenguajes de programación más demandados en 2023 [40]

- **Librerías necesarias para el entrenamiento.** PyTorch es un marco de deep learning que se distingue por su estrecha integración con Python, un lenguaje esencial en ciencia de datos gracias a su simplicidad y versatilidad. Python permite desarrollar modelos de aprendizaje profundo de manera rápida y eficiente, complementado por bibliotecas como NumPy, que facilita operaciones complejas con matrices, y matplotlib, que ayuda a visualizar resultados y procesos de entrenamiento. Esta combinación hace que la creación, depuración y prototipado de modelos sea más sencilla. La amplia comunidad de Python enriquece el ecosistema de PyTorch con recursos y soporte continuos, consolidándolo como una herramienta clave para desarrolladores. Además de las propias librerías de Python, se usarán las librerías propias de Ultralytics para la llamada a las funciones para lanzar los entrenamientos. [41]

3. Especificaciones y restricciones de diseño

Este proyecto se centra en la creación de una aplicación que sea capaz de reproducir un vídeo 360° al mismo tiempo que realiza detecciones a partir de un modelo de YOLOv8 entrenado previamente, habiendo sido probado con un conjunto de imágenes amplio y diverso.

3.1 Especificaciones del diseño

- La aplicación debe ser capaz de realizar la detección de lubinas tanto en las imágenes del conjunto de pruebas como en el propio vídeo 360°, utilizando para ello el mejor modelo obtenido. A la hora de reproducir el vídeo, por cada lubina detectada se debe mostrar una caja delimitadora.
- El sistema debe ser robusto y ser capaz de realizar detecciones en diferentes situaciones independientemente de la posición de las lubinas.
- Durante el desarrollo del proyecto se utilizará control de versiones.
- El sistema final debe contar con buenos resultados en cuanto a precisión y pérdidas se refiere y debe tener un valor de IoU superior al 70%.

3.2 Restricciones del diseño

- El etiquetado de imágenes se realiza por personal no cualificado, lo cual puede suponer una mayor cantidad de errores de cara a los entrenamientos y en los resultados finales del modelo.
- La aplicación que reproduce los vídeos 360° se debe realizar preferiblemente con las librerías de Python de PyOpenGL y PyOpenGL accelerate, en caso de no ser compatible por alguna razón se debe buscar una alternativa.
- Se deberá usar el programa Insta360 Studio, para que previo a la obtención de imágenes para elaborar el dataset, se puedan reproducir los vídeos originales que son propiedad de la UPM y seleccionar cada vídeo considerando que debe ser distinto de los anteriores para facilitar la capacidad del modelo para generalizar.
- El sistema hará uso de YOLOv8 para la detección de objetos. Se caracteriza por ser una herramienta muy fiable y con la capacidad de ofrecer muy buenos resultados. Además, es muy sencilla de usar, gracias en parte a la gran documentación que se puede localizar en el propio sitio web de Ultralytics.
- Durante el proceso de etiquetado, no se pueden utilizar etiquetas que tengan algún tipo de rotación. Las etiquetas deben mantener un ángulo de 0°, garantizando que se ajusten de manera uniforme a los objetos sin inclinaciones, lo cual facilita el proceso de entrenamiento y mejora la precisión del modelo al evitar posibles errores en la detección y localización de los objetos.
- El entorno del proyecto se elaborará en Visual Studio Code, que no solo ofrece la opción de edición de código Python, sino que se puede utilizar el terminal para descargar las librerías que el proyecto en su conjunto requiere.

- A la hora de seleccionar fotos para la elaboración del dataset de imágenes, éstas se deben conseguir únicamente a partir de los vídeos 360°, ya que ofrecen distintos ángulos y así se puede hacer un análisis más profundo.
- A la hora de etiquetar imágenes, se deberá utilizar la herramienta CVAT, pero no la herramienta online, sino un despliegue local de la herramienta realizado en el NAS del CITSEM usando contenedores Docker. Esto se hace así por privacidad de la información.
- La aplicación debe ser desarrollada utilizando el lenguaje de programación Python, ya que es el más común para tareas de Machine Learning, y en el que más información se puede encontrar tanto para la etapa de entrenamiento como para la posterior validación de resultados.

4. Descripción de la solución propuesta

Se describe en este capítulo la solución propuesta para la detección de lubinas en vídeos 360 desde la creación del conjunto de datos hasta el despliegue del modelo en una aplicación que muestra las cajas delimitadoras sobre el vídeo.

4.1 Creación del dataset

La creación del dataset es un paso fundamental para garantizar que el modelo de red neuronal alcanzara los resultados esperados. Los vídeos en formato 360° que sirvieron como fuente de datos fueron proporcionados por la tutora del proyecto. Estos vídeos incluían diversas condiciones de iluminación, ángulos de visión y niveles de contraste, lo cual resultó ideal para construir un conjunto de datos variado y representativo. Aunque los vídeos variaban en tamaño y duración, la mayoría tenían una duración aproximada de 30 minutos y un tamaño cercano a 20 GB. En cuanto a su calidad, la resolución era de 3840x1920 píxeles, con una tasa de 30 fotogramas por segundo (fps).

El primer paso del proceso consistió en cambiar el formato del video del propio de la cámara Insta360 a un formato estándar, en este caso MP4. Para ello, se utilizó el programa Insta360 Studio que proporciona el fabricante de la cámara. Este programa permite exportar los vídeos en formato 360° compatible con reproductores como VLC Media Player. En la siguiente figura, se puede apreciar la interfaz de Insta360 Studio reproduciendo un vídeo 360°:

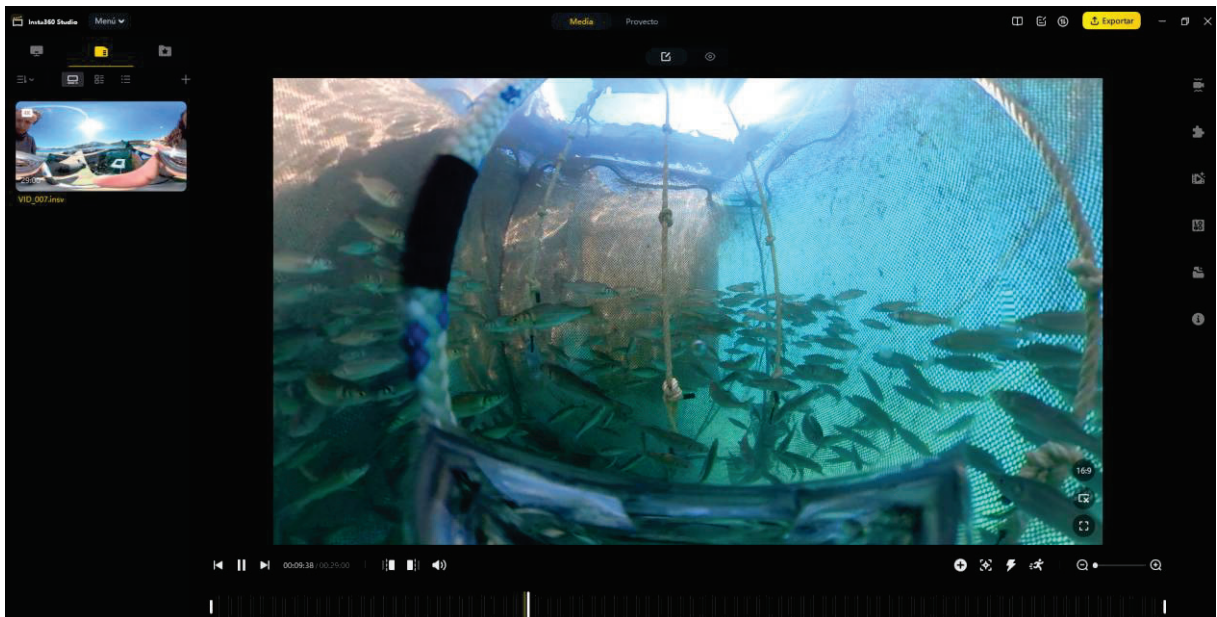


Figura 15 Intefaz Insta360 Studio reproduciendo vídeo 360°

Una vez convertidos los vídeos, se empleó un programa desarrollado en el Proyecto Fin de Grado de Héctor García Palencia que está elaborando a la vez que este trabajo para la extracción de imágenes de manera aleatoria a partir de los vídeos. Este software permitía al usuario especificar la cantidad de imágenes deseadas y seleccionaba de forma aleatoria los momentos exactos de captura. En este caso, se decidió extraer 6 imágenes por vídeo, con el

objetivo de maximizar la diversidad del dataset. Han sido extraídos un total de 23 vídeos, de los cuales se han obtenido 138 imágenes. Las imágenes restantes, hasta completar las 200 en total, provienen del material previamente etiquetado por Rocío Herruzo Barroso. Aunque esta herramienta no generaba imágenes 360° al momento de la extracción, sí producía vistas panorámicas que integraban todas las perspectivas en una única imagen, lo que facilitaba la identificación y posterior etiquetado de las lubinas. En la figura 16, se muestra un ejemplo de imagen que se obtenía y que serían utilizadas para los entrenamientos de YOLOv8. Finalmente, se dispone de 200 imágenes.



Figura 16 Imagen panorámica obtenida por el programa

Posteriormente, las imágenes obtenidas fueron etiquetadas utilizando una herramienta local de CVAT. Durante este proceso, se etiquetaron todas las lubinas visibles, independientemente de si aparecían deformadas debido a la proyección panorámica. En la siguiente figura, se aprecia un ejemplo de etiquetado de imágenes:

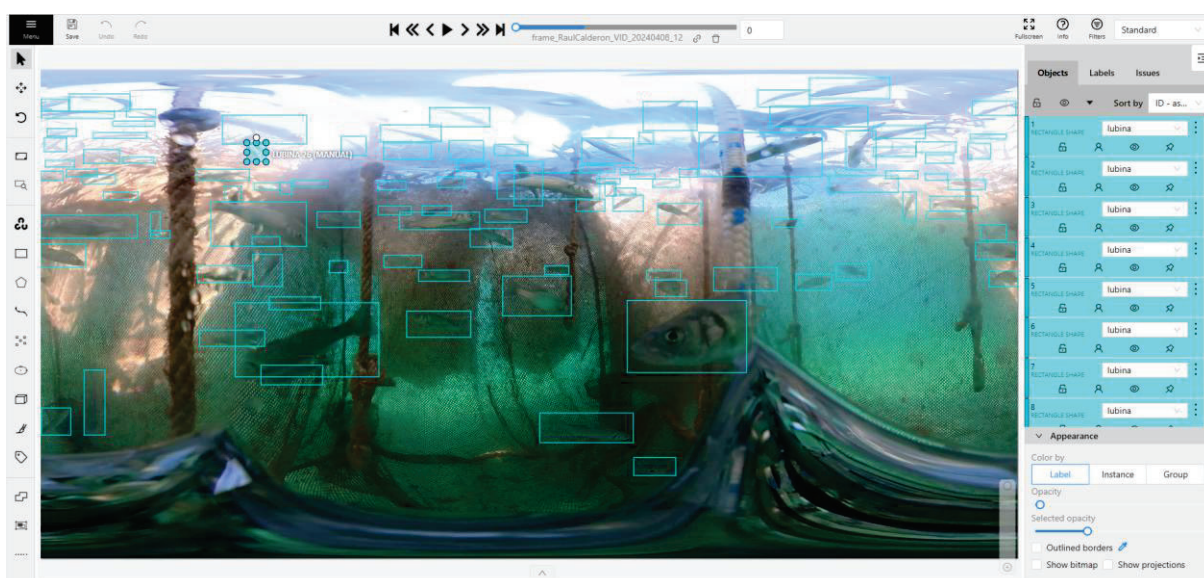


Figura 17 Ejemplo de imagen etiquetada con CVAT

Una vez etiquetadas las imágenes se separaron en distintos conjuntos de datos, comprobando que las imágenes provenientes de un vídeo están todas en el mismo conjunto. Esta separación evitó mezclar datos de entrenamiento, validación y prueba, asegurando resultados fiables y

no sesgados. Se dispone de un total de 200 imágenes, distribuidas de la siguiente forma: el 80% (160 imágenes) se destinó al conjunto de entrenamiento, el 15% (30 imágenes) al conjunto de validación y el 5% restante (10 imágenes) al conjunto de prueba.

4.2 Entrenamientos del modelo de YOLOv8

Al finalizar el proceso de extracción y etiquetado de imágenes, se procedió al entrenamiento del modelo YOLOv8. Este entrenamiento se llevó a cabo en el entorno local del ordenador del CITSEM, que estaba equipado con un hardware capaz de realizar este tipo de tareas en tiempos razonables. Como entorno de desarrollo se utilizó Visual Studio Code, una plataforma que facilitó tanto la edición del código en Python como la instalación de las librerías necesarias para configurar y entrenar la red neuronal.

Para iniciar el entrenamiento, se utilizó un archivo YAML que definía la configuración del modelo. Este archivo especificaba las rutas de los conjuntos de entrenamiento (train) y validación (val), además de incluir parámetros como el número de clases (nc) y los nombres de dichas clases. En este caso, no se incluyó la ubicación del conjunto de prueba (test) en el archivo YAML, ya que este conjunto no se emplea durante el entrenamiento propiamente dicho. En lugar de eso, el conjunto de test se reserva exclusivamente para evaluar y medir la precisión del modelo tras finalizar el proceso de entrenamiento. En la siguiente figura, se muestra el archivo YAML utilizado para este trabajo:

```
train: D:\TRABAJO FINAL DE GRADO (YOLOv8-lubinas)\tfg\Entrenamiento210ctubre2024Mixto\images\train
val: D:\TRABAJO FINAL DE GRADO (YOLOv8-lubinas)\tfg\Entrenamiento210ctubre2024Mixto\images\val
nc: 2
names: ["lubina", "Lubina"]
```

Figura 18 Archivo YAML de entrenamiento

Cuando se define la configuración en un archivo YAML, es importante señalar que al especificar la ubicación para el conjunto de entrenamiento (train), únicamente se debe proporcionar la ruta al directorio que contiene las imágenes correspondientes. Lo mismo aplica para el conjunto de validación (validation). Esto se debe a que, siempre que el sistema de archivos esté correctamente organizado, YOLO será capaz de identificar de forma automática las etiquetas asociadas a cada conjunto, sin necesidad de especificarlas de manera adicional. En cuanto a sistema de archivos utilizado, se tiene la siguiente estructura:

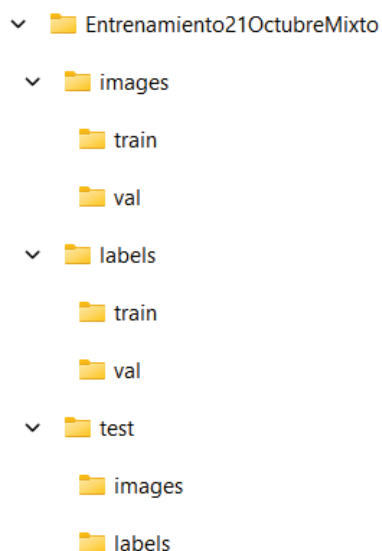


Figura 19 Sistema de Archivos utilizado para el entrenamiento de la CNN

Lo más llamativo de este apartado es que la carpeta de test también contiene etiquetas y esto se debe a que para poder determinar la precisión del modelo, se va a hacer una comparación entre la zona común del área que abarcan las etiquetas de las imágenes de test con el área obtenida por el modelo al hacer las detecciones. A esto se le conoce como IoU (Intersection over Union) y se utiliza para conocer la precisión de las detecciones. Más adelante, se explicará cómo funciona con detenimiento.

4.3 Validación de resultados de entrenamiento

Al finalizar los entrenamientos con YOLOv8, el sistema genera una carpeta que contiene todos los resultados obtenidos. En esta carpeta se incluyen una serie de imágenes que ofrecen información visual sobre el rendimiento del entrenamiento, así como los modelos generados, ya sea el mejor modelo (best.pt) y el último modelo (last.pt). El mejor modelo es aquel que alcanza el valor óptimo de la función de pérdidas en el conjunto de validación. Se van a definir métricas útiles para evaluar modelos entrenados. Las imágenes resultan especialmente útiles para analizar el desempeño del modelo y extraer conclusiones que puedan servir para mejorar los resultados en futuros entrenamientos. Lo más destacado a tener en cuenta es:

- **Matriz de confusión.** La matriz de confusión es una herramienta que organiza la información en forma de tabla y muestra los resultados obtenidos al evaluar un modelo de clasificación con un conjunto de datos de prueba. En ella se reflejan cuatro métricas clave: los verdaderos positivos (TP) representan los casos en los que el modelo clasifica correctamente un objeto o entidad en su clase correspondiente. Por otro lado, los falsos positivos (FP) que tienen lugar cuando el modelo clasifica incorrectamente una instancia como perteneciente a una clase cuando no lo es. En cuanto a los verdaderos negativos (TN), se refieren a los objetos correctamente clasificados como no pertenecientes a una clase. Por último, los falsos negativos (FN) son las situaciones en las que el modelo no identifica correctamente a un elemento que sí pertenece a una clase. Gracias a esta matriz, es posible identificar cuántas veces

el modelo realiza predicciones correctas y cuántas veces se equivoca, proporcionando una visión clara de su rendimiento. [42] En la siguiente figura, se muestra la distribución de estos valores en una matriz de confusión:

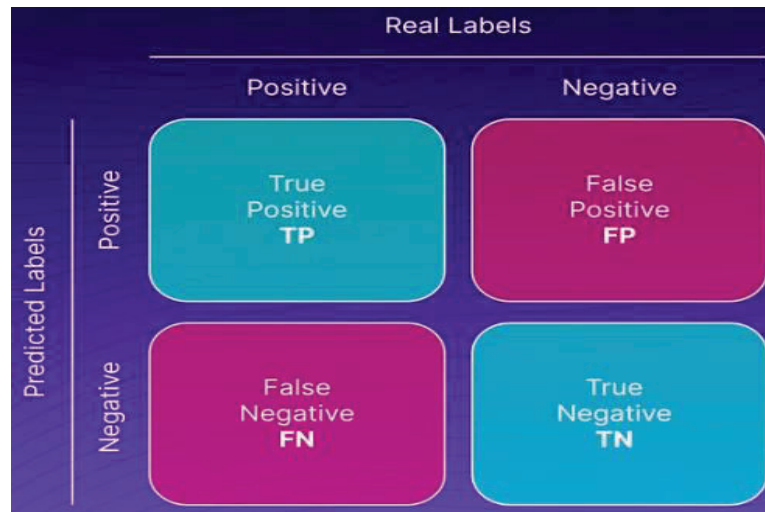


Figura 20 Matriz de confusión [42]

- **Curva de puntuación F1.** Es de gran utilidad para analizar el equilibrio que logra el modelo entre falsos positivos y falsos negativos según el umbral seleccionado, proporcionando información clave sobre su desempeño. [43]

Para calcular este valor se hace de la siguiente manera:

$$f\text{-measure} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Curva de precisión frente a recall.** Refleja el equilibrio entre la precisión y el recall en problemas de clasificación. Tiene gran importancia al trabajar con clases desequilibradas. [43]
- **Curva de precisión.** Muestra cómo varía la precisión del modelo al modificar los diferentes umbrales de confianza. Esta visualización permite analizar cómo se comporta la precisión a medida que se ajusta el umbral. Un valor muy bueno estaría cerca del 1 y en un caso ideal sería similar a una función escalón. [43]
- **Curva de recall.** Refleja cómo cambia el valor del recall con los distintos valores de confianza. En el mejor de los casos, sería una línea horizontal cercana al 1. [43]
- **Box_loss.** Mide el error en la predicción de las coordenadas de las cajas delimitadoras, con el objetivo que el modelo trate de realizar predicciones más exactas y que se alineen mucho más las predicciones con las cajas delimitadoras reales. [44]

- **Cls_loss.** Evalúa el error al predecir la clase del objeto dentro de cada caja delimitadora. Esta pérdida se asegura de que el modelo identifique correctamente a qué categoría pertenece el objeto detectado. [44]
- **Dfl_loss.** Mide la detección de objetos en imágenes que son borrosas, en los que se plantean escenarios difíciles. Se pretende que sirva al modelo para lograr la mayor precisión ante situaciones adversas. [44]
- **mAP50.** Representa la media de la precisión calculada utilizando un umbral de intersección sobre unión (IoU) de 0.50. Este valor se centra en medir la precisión del modelo considerando las detecciones más sencillas, es decir, aquellas con un solapamiento razonable entre las predicciones y las cajas reales. [43]
- **mAP50-95.** Se calcula como el promedio de la precisión en un rango de umbrales de IoU que va desde 0.50 hasta 0.95. Esta métrica proporciona una visión más completa del rendimiento del modelo, ya que incluye tanto detecciones fáciles como más complejas y exigentes. [43]

En la figura 23, se puede apreciar como normalmente según van pasando las iteraciones los resultados normalmente van mejorando:

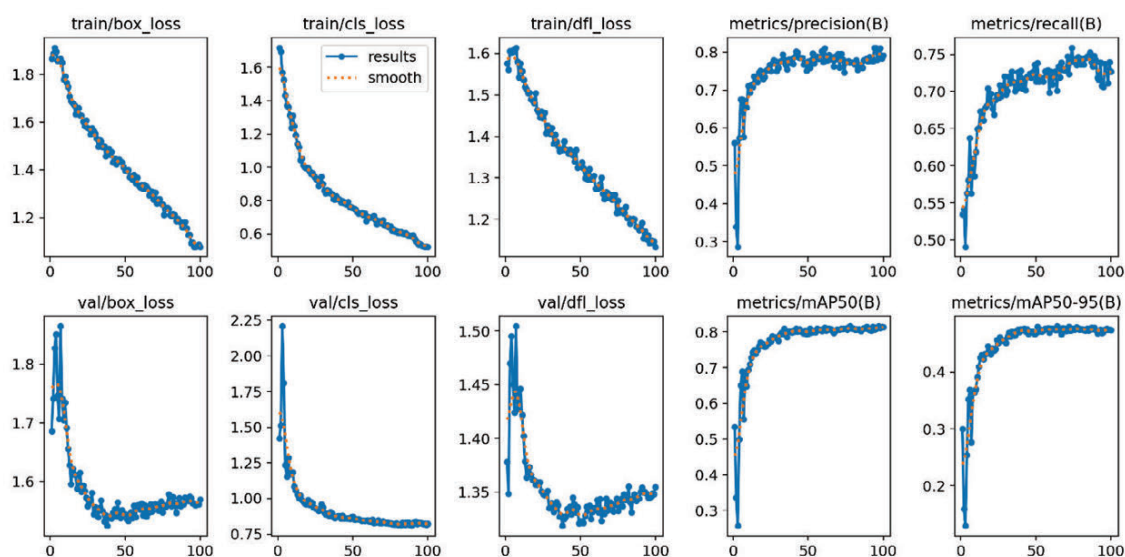


Figura 21 Ejemplo de resultados de entrenamiento de YOLOv8

- **Intersection over Union (IoU).** Es una métrica importante para medir la precisión de modelos de detección de objetos, tal y como se plantea en este caso. Básicamente, mide qué tan bien coincide la predicción del modelo con la realidad, representada por dos cajas: una que el modelo propone (color morado) y otra que marca la posición real del objeto (color azul), conocida como “ground truth”. El cálculo del IoU se hace de la siguiente manera: se divide el área en la que ambas cajas se superponen por el área total que abarcan juntas. Un valor de $\text{IoU}=1$ indica una coincidencia exacta, mientras que un valor de $\text{IoU}=0$ significa que no hay superposición alguna. [45]



Figura 22 Fórmula para el cálculo de IoU [45]

Una vez que se sabe cómo realizar el cálculo, se deben conocer los valores que se consideran buenos, aceptables o malos resultados. Esto se va a mostrar mediante un ejemplo de varios IoU en la Figura 25:



Figura 23 Ejemplo de detecciones con IoU [45]

4.4 Optimizadores de entrenamiento

Los optimizadores tienen una labor clave a la hora de realizar los entrenamientos de la red neuronal, ya que se encargan de ajustar la tasa de entrenamiento para minimizar las funciones de pérdida con el objetivo de alcanzar mayor precisión y rendimiento. Algunos de los optimizadores más destacados son:

- **Adam (Adaptive Moment Estimation).** Adam es un optimizador diseñado para ajustar los parámetros de los modelos de aprendizaje automático de manera adaptativa, siendo especialmente útil en escenarios de alta complejidad, como en redes neuronales profundas o tareas de visión por computadora. Su capacidad para manejar gradientes dispersos, ajustar automáticamente la tasa de aprendizaje y corregir el sesgo inicial lo convierte en una gran alternativa para entrenar modelos con grandes volúmenes de datos. Sin embargo, en casos como conjuntos de datos

pequeños o problemas que requieren un control extremadamente preciso del proceso de aprendizaje, alternativas como SGD, con ajustes manuales en la tasa de aprendizaje, pueden ser más efectivas para obtener una mayor capacidad de generalización. [46]

- **AdamW (Adam with weight decay).** Es un optimizador ampliamente utilizado en el aprendizaje profundo por su capacidad para manejar eficazmente la regularización de modelos complejos. A diferencia de Adam, AdamW introduce una mejora clave al desacoplar el decaimiento de peso de las actualizaciones del gradiente, lo que garantiza que la regularización no interfiera con el proceso de optimización. Esta característica permite que los modelos generalicen mejor, especialmente en arquitecturas de gran escala como redes convolucionales profundas o tareas de procesamiento de lenguaje natural. Además, AdamW destaca por su estabilidad durante el entrenamiento y su eficacia para prevenir el sobreajuste, lo que lo convierte en una opción ideal para tareas con grandes conjuntos de datos y modelos exigentes. Además, mantiene las ventajas de Adam, pero con un enfoque más preciso para aplicar regularización, consolidándolo como una herramienta esencial en la construcción de modelos de última generación. [47]
- **SGD (Stochastic Gradient Descent).** Se caracteriza por ser un método clave en el aprendizaje automático, especialmente para entrenar modelos de aprendizaje profundo debido a su simplicidad y efectividad. A diferencia del descenso de gradiente tradicional, que calcula el gradiente sobre todo el conjunto de datos, SGD realiza actualizaciones utilizando muestras individuales o pequeños lotes, lo que reduce el costo computacional y acelera el entrenamiento en grandes volúmenes de datos. Su enfoque estocástico introduce ruido en las actualizaciones y puede generar una trayectoria de aprendizaje más volátil, que ayuda a escapar de puntos de silla en el espacio de la función de pérdida. SGD es particularmente eficaz para evitar el sobreajuste en escenarios con datos ruidosos, ya que sus actualizaciones simples y progresivas favorecen una mejor generalización en comparación con optimizadores más complejos como Adam. Sin embargo, requiere un ajuste cuidadoso de la tasa de aprendizaje, ya que un valor inadecuado puede provocar divergencias o ralentizar la convergencia. [48]

Cada optimizador tiene aplicaciones específicas para cada caso de uso. En el contexto del proyecto, aunque SGD puede ofrecer buenos resultados, presenta una limitación importante al no incluir un mecanismo de decaimiento de pesos, lo que implica realizar múltiples ajustes para encontrar la tasa de aprendizaje óptima para el conjunto de datos. Por esta razón, se ha decidido optar principalmente por Adam, debido a su eficiencia en el ajuste de parámetros y su capacidad para adaptarse automáticamente a los gradientes. Sin embargo, si durante el entrenamiento se detecta sobreajuste (overfitting), se utilizará AdamW, ya que su capacidad para desacoplar el decaimiento de pesos de las actualizaciones del gradiente permite mejorar la regularización sin afectar negativamente el proceso de optimización. En el apartado de resultados se mostrará en detalle qué optimizador se utilizó en cada entrenamiento y las razones que llevaron a usarlo.

4.5 Evaluación de resultados de IoU

Para este apartado, debido a que YOLOv8 no genera al final del entrenamiento un resultado de IoU, se ha codificado un programa independiente al de entrenamiento, que utiliza el modelo entrenado por YOLOv8, junto con el conjunto de imágenes de test y también se deben pasar las etiquetas del conjunto de test. Este programa se elaboró teniendo como referencia el pseudocódigo de la referencia [45] de este documento. Además, se le agregaron dos módulos de código que realizan las siguientes funcionalidades:

- **Generación de Histogramas.** Esta opción permitía generar al terminar la ejecución del programa un histograma por cada foto de tal manera que se pudiese ver con qué probabilidad se habían hecho las detecciones. Un ejemplo se muestra en la Figura 26:

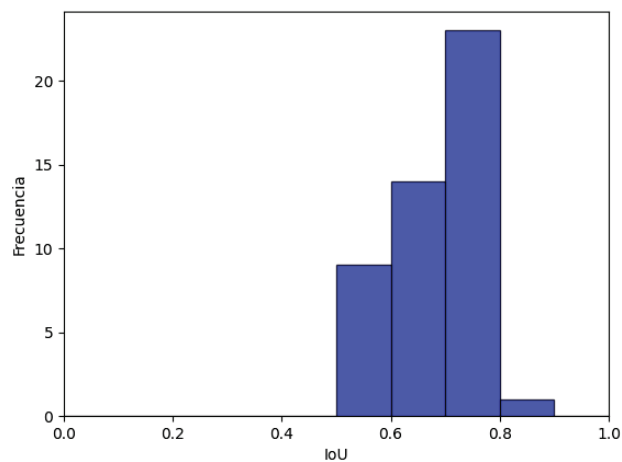


Figura 24 Gráfica de resultados de IoU para una imagen

La función del histograma es mostrar de manera visual la distribución de los valores de IoU obtenidos por el modelo YOLOv8 para las detecciones realizadas en cada imagen del conjunto de prueba. Cada barra del histograma representa un rango de IoU y muestra la cantidad de detecciones que caen dentro de ese rango. Esto permite evaluar si el modelo está realizando detecciones con alta certeza (valores cercanos a 1) o si hay muchas detecciones con baja precisión, lo que podría indicar falsos positivos o un bajo rendimiento en general. También, se genera un histograma que recoge todas las detecciones de cada una de las imágenes en una sola representación.

- **Generación de imágenes con detecciones.** Se diseñó un módulo para procesar cada imagen del conjunto de prueba y añadir una representación visual de las detecciones. En este módulo, las cajas delimitadoras correspondientes a las etiquetas de referencia se dibujan en color rojo, mientras que las predicciones del modelo se representan en color verde. De esta forma, se puede observar de manera clara y visual cómo se comporta el modelo, comparando directamente sus predicciones con las etiquetas reales. En la figura 27 se muestra un ejemplo de ello:

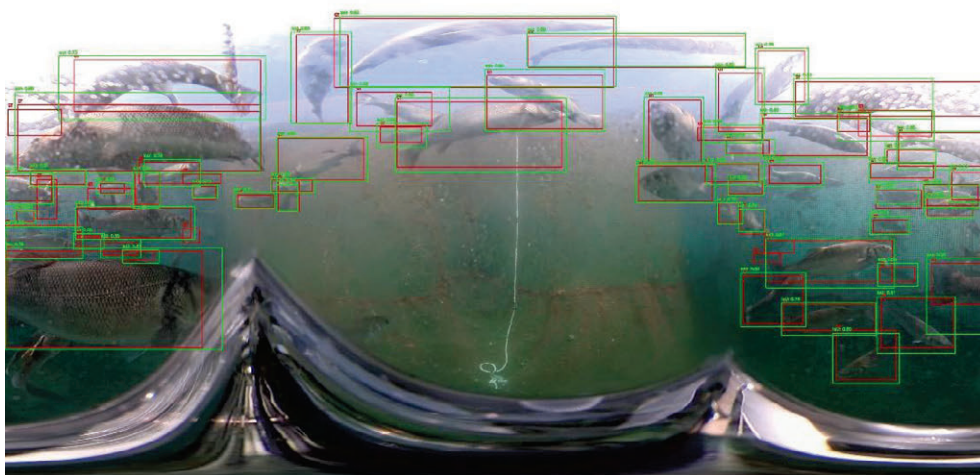


Figura 25 Representación visual etiquetas de referencia frente a las predichas

4.6 Programa final de detecciones

Una vez anotadas las imágenes y completados los entrenamientos, se seleccionó el mejor modelo para la siguiente fase: desarrollar un programa capaz de realizar detecciones de lubinas en cualquier vídeo proporcionado, simulando además un entorno de reproducción 360°.

Inicialmente, se exploraron diferentes opciones para implementar un reproductor de vídeos 360°. Entre las herramientas evaluadas, PyOpenGL y PyOpenGL Accelerate destacaron como las más completas debido a su soporte avanzado para gráficos tridimensionales. Sin embargo, durante la implementación surgieron incompatibilidades con otras librerías esenciales del proyecto, lo que llevó a descartar esta opción. En su lugar, se optó por una solución alternativa basada en las librerías OpenCV, NumPy y Math, que permitieron simular el entorno 360° generando las proyecciones necesarias para lograr una experiencia inmersiva adecuada.

El programa no solo reproduce los videos 360° con detecciones en tiempo real, sino que también realiza tareas adicionales al finalizar la ejecución. Por un lado, genera un archivo de texto en el que almacena las cajas correspondientes a las detecciones realizadas por el modelo. Estas cajas incluyen coordenadas normalizadas y se añade una columna extra para registrar el número exacto del fotograma en el que se detectaron. Por otro lado, el programa guarda un vídeo procesado en el que se visualizan todas las detecciones realizadas durante la reproducción, lo que facilita un análisis posterior. La ejecución del programa puede ser detenida manualmente pulsando la tecla "q", momento en el cual todos los resultados se almacenan de manera automática. En la Figura 28, se muestra un ejemplo de ejecución:

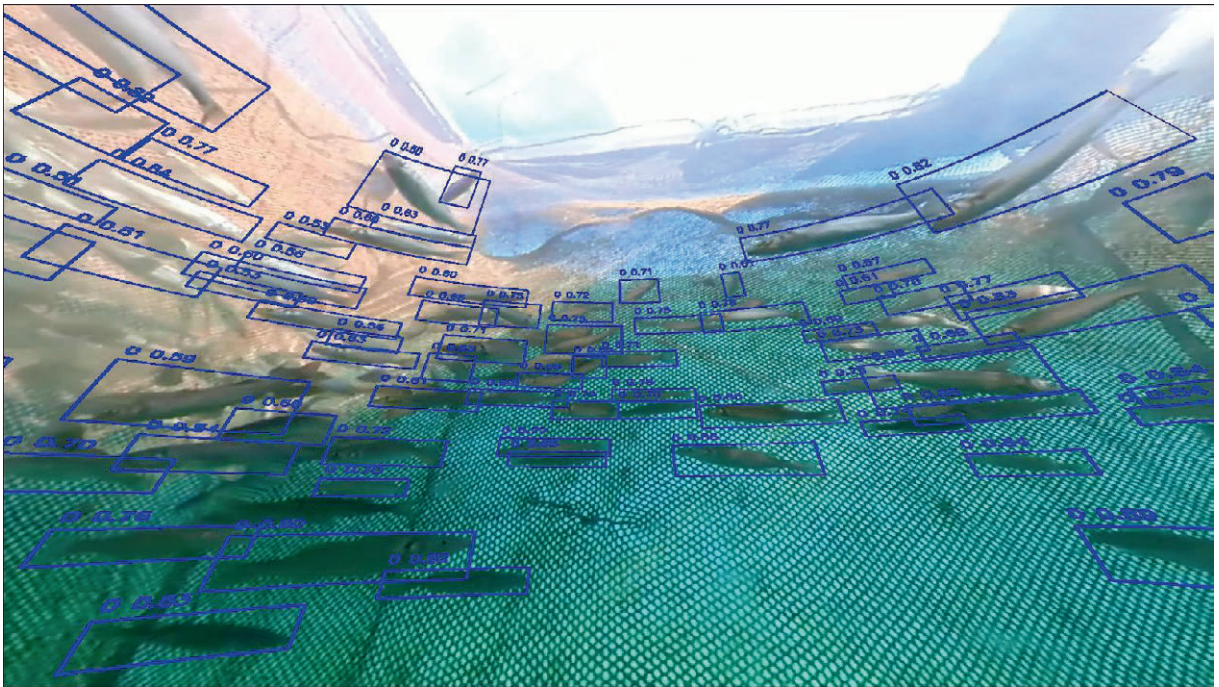


Figura 26 Programa de detección de lubinas en entorno 360°

Sin embargo, uno de los principales inconvenientes del reproductor 360° fue que no ofrecía una vista completa del entorno. El enfoque se centraba únicamente en la región visible en cada momento, lo que implicaba que algunas detecciones pudieran quedar fuera del campo de visión. Esto planteó un problema significativo: si en el archivo de texto solo se almacenaban las detecciones de una parte del entorno, se perdía información valiosa para analizar el comportamiento completo de las lubinas. Esto limitaba la utilidad del sistema, especialmente pensando en facilitar el análisis para futuras personas que trabajen en este proyecto.

Por esta razón, se decidió cambiar la estrategia y desarrollar un reproductor panorámico, capaz de mostrar todo el campo visual en una única vista. Este enfoque permite monitorear a todas las lubinas en todos los instantes de tiempo, asegurando que ninguna detección quede fuera de registro. Con esta mejora, el archivo de texto generado almacena información completa, lo que garantiza que futuras investigaciones puedan llevar a cabo un análisis más exhaustivo del comportamiento de las lubinas, optimizando así los procesos acuícolas. Esta nueva solución asegura que los datos recopilados sean más consistentes y útiles para los objetivos finales del proyecto, aportando un marco sólido para su continuidad y aplicación práctica. En la Figura 29, se muestra la aplicación desarrollada para detección de todas las vistas de forma panorámica:



Figura 27 Programa de detección de lubinas en vista panorámica

Finalmente, se ofrece el archivo con las detecciones por cada fotograma, que se almacena de la siguiente manera:

- **Indicador de fotograma en el que se produce la detección.** Se recuadra con color rojo en la Figura 30.
- **Identificador de la clase del objeto detectado.** En este caso, al haber solo un elemento a detectar será siempre de valor 0. Se representa con el color verde en la Figura 30.
- **Valores de la Bounding box asociada a la lubina detectada.** Representado con color naranja en la Figura 30.

899	0	0.367448	0.425000	0.029687	0.018750
899	0	0.727344	0.443490	0.020833	0.015104
899	0	0.020313	0.423698	0.025521	0.032813
899	0	0.725521	0.373958	0.027083	0.019792
900	0	0.932552	0.332813	0.030729	0.030208
900	0	0.895312	0.556510	0.028125	0.050521
900	0	0.665885	0.219010	0.021875	0.027604
900	0	0.899740	0.483854	0.026042	0.015625

Figura 28 Ejemplo fichero detecciones realizadas por el modelo entrenado

Finalmente, vamos a entender qué significa cada uno de los valores asociados a cada Bounding box, que en la Figura 30 se muestran recuadradas en color naranja. Se distribuyen según este orden: [49]

Tabla 1 Tabla Valores Bounding boxes en fichero de texto [49]

Centro_Eje_X	Centro_Eje_Y	Ancho	Alto
--------------	--------------	-------	------

Los valores asociados a Centro_Eje_X y Centro_Eje_Y, hacen referencia al punto central de los ejes X e Y de la Bounding box detectada. Con estos valores, se pueden hacer muchos análisis como veremos en el siguiente apartado para la obtención de métricas acerca de la posición, área o centroides de las cajas delimitadoras que representan las detecciones. A continuación, se pasa a mostrar cómo funcionan los ejes, ya que en este caso no se siguen los ejes cartesianos tradicionales, sino que en este caso el eje de coordenadas se encuentra en la esquina superior izquierda. [49] Se muestra un ejemplo en la Figura 31:



Figura 29 Valores Bounding box en fichero de texto [49]

5. Resultados

En este apartado, se van a mostrar los resultados obtenidos, tanto los relacionados con los entrenamientos como los del IoU y otras métricas. Además, se va a mostrar de manera detallada las razones que llevaron a hacer ciertos cambios en función de tendencias que se encontraron y cómo se trató de dar solución, indicando en todo momento si los resultados obtenidos después de cada entrenamiento eran los esperados según lo planeado previamente.

5.1 Evolución de entrenamientos

En este apartado, se van a incluir en primer lugar los detalles sobre las métricas obtenidas en el fichero generado al finalizar cada entrenamiento y al final se comentarán las métricas de salida por consola para cada uno de ellos.

5.1.1 Primer entrenamiento

- **Fecha:** 7 de octubre de 2024 (Prueba de concepto).
- **Número de imágenes:** 109 imágenes en entrenamiento y 15 en validación.
- **Conclusiones:** El modelo, siendo una primera prueba de concepto, ofrece unos resultados prometedores. Obtiene un mAP50 de 0.739 y un F1-score promedio de 0.71. Además, aunque la precisión mejora con umbrales altos, esto reduce el recall, mostrando que el modelo prefiere ser cauteloso al hacer predicciones. En este caso, las gráficas de recall y precisión dan a entender que el modelo en muchos casos duda de si etiquetar un elemento como objeto a detectar y no lo anota finalmente. Las pérdidas a medida que avanza el entrenamiento se reducen. En cuanto al modelo de YOLOv8 utilizado al ser una prueba de concepto, se utilizó YOLOv8n, que es el modelo más básico y en caso de ser necesario se evaluará la opción de hacer uso de una versión más sofisticada de YOLOv8 si los recursos hardware así lo permiten para mejorar los valores finales de las métricas y reducir las pérdidas.
- **Gráficas:**

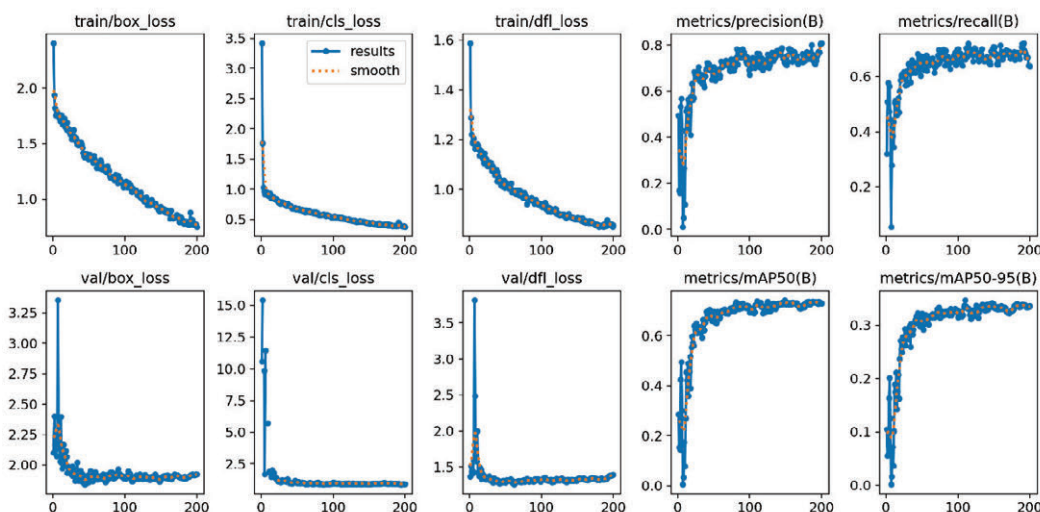


Figura 30 Gráficas de métricas del primer entrenamiento

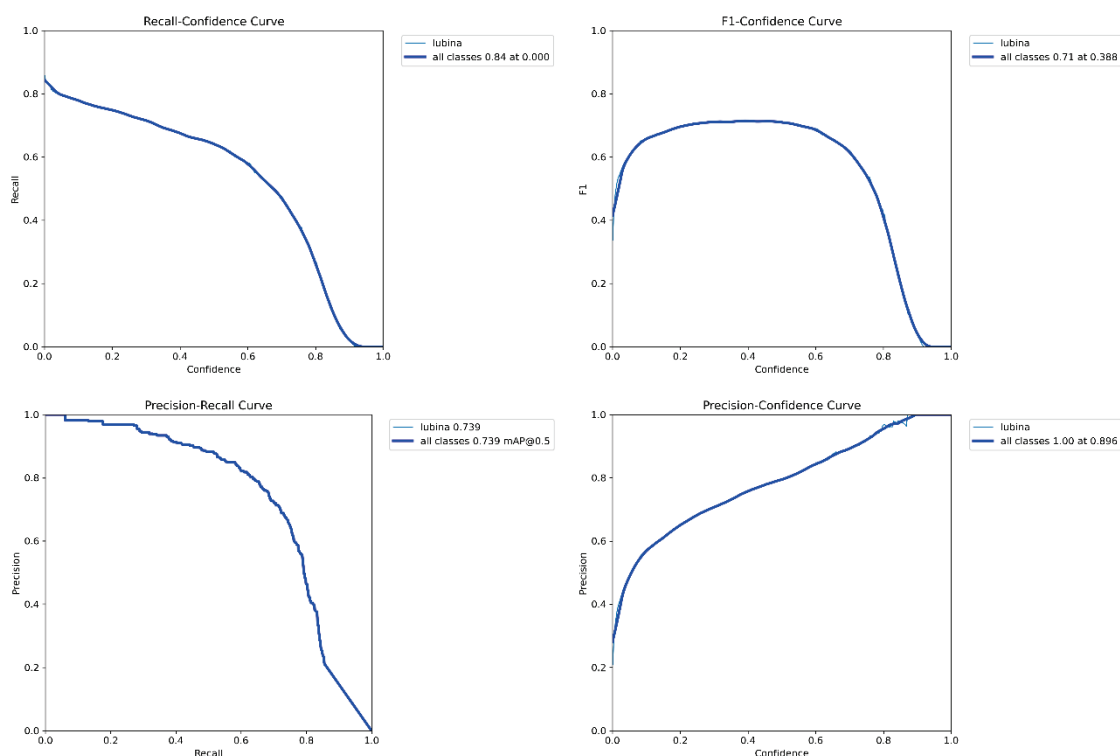


Figura 31 Gráficas recall y precisión primer entrenamiento

5.1.2 Segundo entrenamiento

- **Fecha:** 17 de octubre de 2024
- **Número de imágenes:** 168 imágenes en entrenamiento y 27 imágenes en validación.
- **Conclusiones:** En este entrenamiento, al tener capacidad de hardware suficiente se apostó por pasar de un modelo YOLOv8n a YOLOv8l, lo cual hizo que el modelo fuese capaz de detectar un mayor número de elementos de interés; sin embargo, la precisión no era la esperada. Con respecto al anterior entrenamiento, se producen disminuciones en los valores de F1-score y en mAP50 y mAP50-95, lo cual es normal teniendo en cuenta que se han agregado nuevas imágenes de alta complejidad visual, debido a zonas de muy baja luminosidad en las que se concentra un gran número de lubinas, de ahí que se apostara por un modelo de mayor capacidad de detección como YOLOv8l. Se analizó que muchos de los falsos positivos y falsos negativos se producían en las regiones en las que las condiciones de luminosidad eran de mayor complejidad, con lo cual el trabajo de cara al siguiente entrenamiento se centró en obtener imágenes para el conjunto de entrenamiento que tuviesen condiciones mucho más complejas que las que se habían usado hasta el momento y en valorar la posibilidad de utilizar un modelo aún mayor de YOLO probando el rendimiento de las versiones superiores y de mayor exigencia.

■ Gráficas:

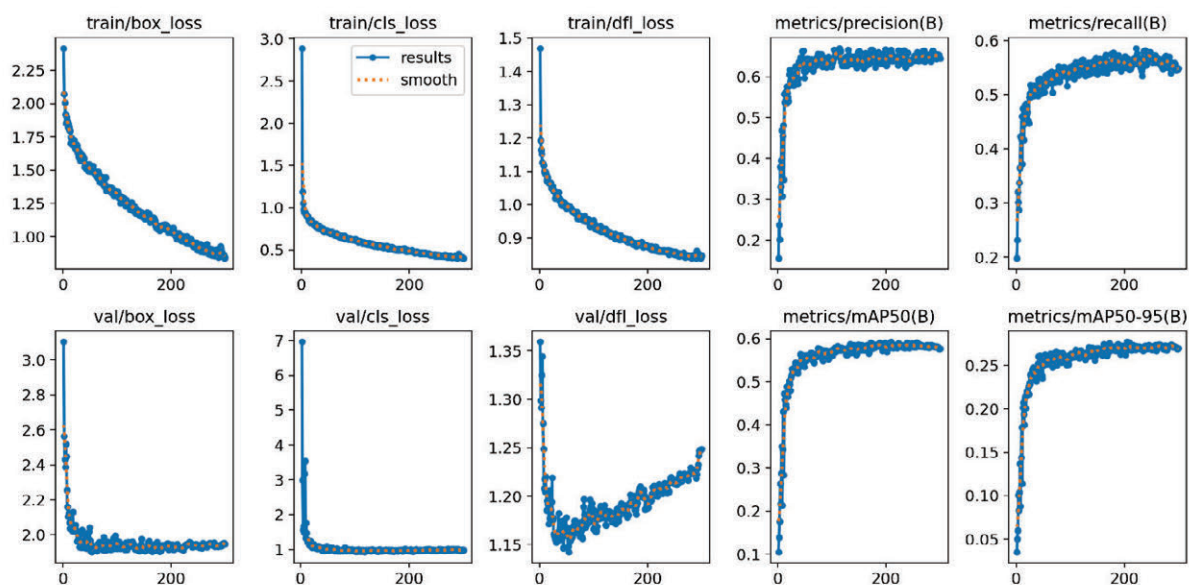


Figura 32 Gráficas de métricas del segundo entrenamiento

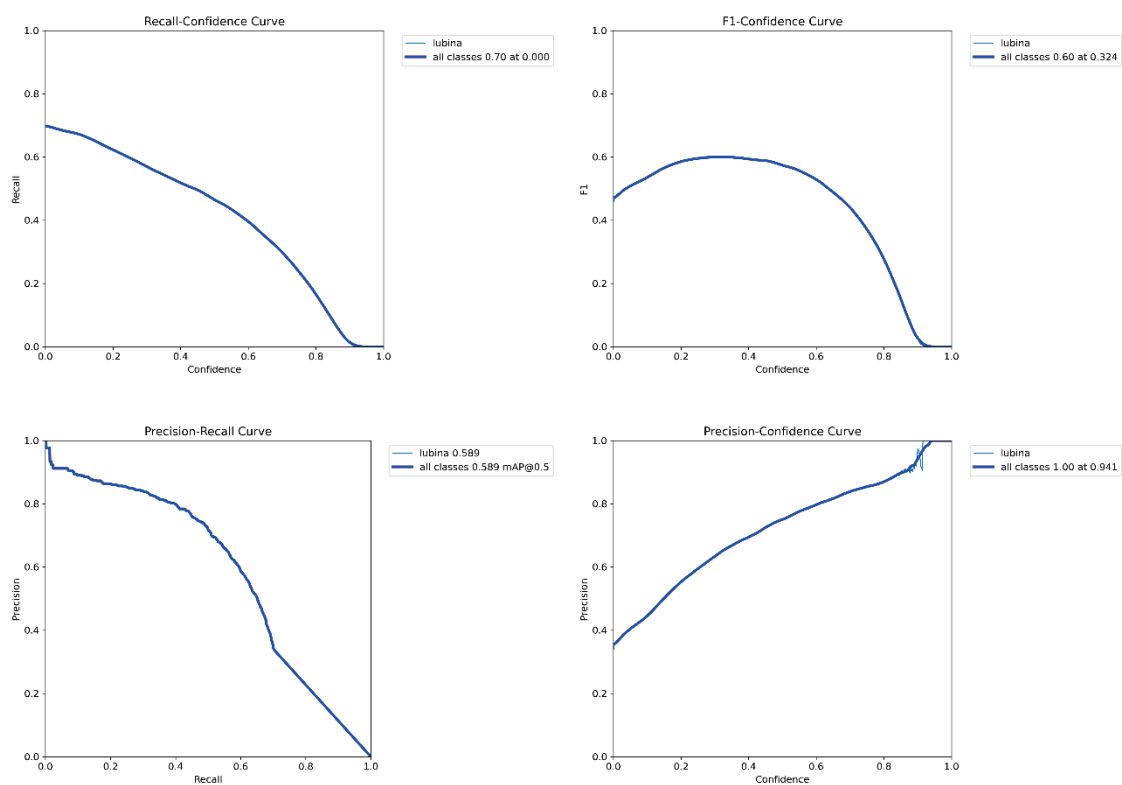


Figura 33 Gráficas de recall y precisión del segundo entrenamiento

5.1.3 Tercer entrenamiento

- **Fecha:** 25 de octubre de 2024
- **Número de imágenes:** 180 imágenes en entrenamiento y 20 en validación.
- **Conclusiones:** En este entrenamiento, se reorganizó el dataset para mejorar los resultados en situaciones de baja luminosidad y condiciones visualmente complejas. Como parte de la preparación, se realizó una investigación sobre hiperparámetros para optimizar aún más el rendimiento del modelo. Uno de los principales hallazgos fue que al incrementar el valor de `imgsz` (tamaño de imagen aplicado al entrenamiento) de 640 a 1920, los resultados mejoraron considerablemente, ya que el modelo podía captar detalles más precisos en las imágenes. Además, se realizaron ajustes en el tamaño del lote (batch size), reduciéndolo a 1 o 2, lo que permitió utilizar el modelo más potente de la familia YOLOv8, que es YOLOv8x. Se optó por esta versión después de comprobar en pruebas previas que las otras versiones no lograban los mismos niveles de rendimiento. Gracias a estos cambios, se observan mejoras significativas. El mAP50 alcanzó un valor de 0.809, superando a los entrenamientos previos, mientras que el F1-score mejoró a 0.75, mostrando un balance mucho más sólido entre precisión y recall. Todo esto se ve reflejado en la drástica reducción de los falsos positivos y negativos. Las curvas de precisión y recall también evidencian un comportamiento más consistente y confiable, destacándose una precisión de 1 a partir de un umbral de confianza de 0.815. Este entrenamiento marca un salto importante en la calidad de las predicciones.
- **Gráficas:**

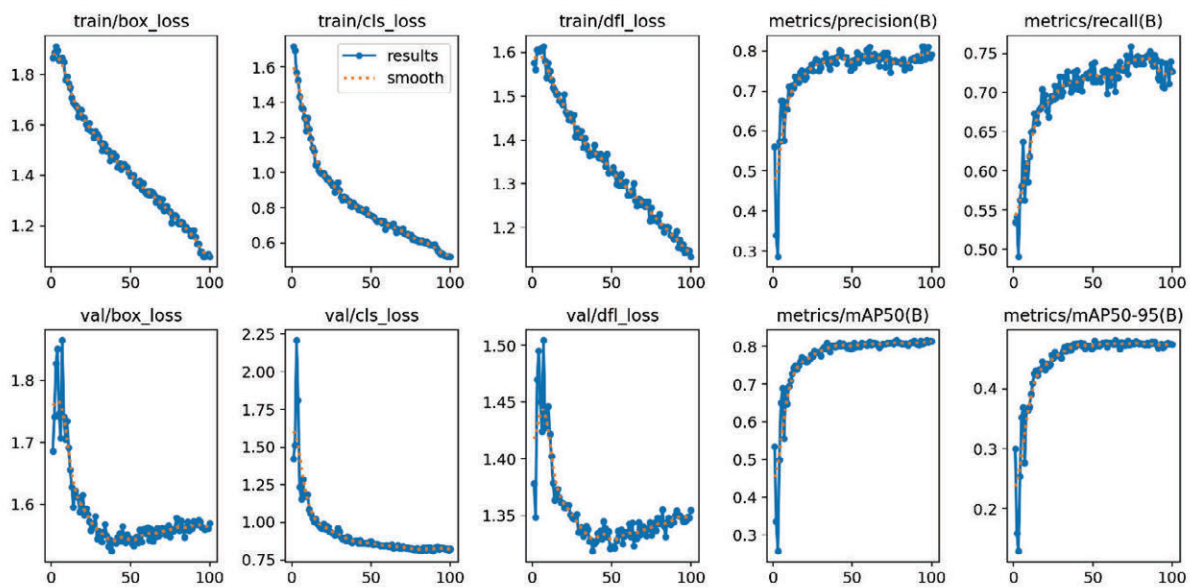


Figura 34 Gráficas de métricas del tercer entrenamiento

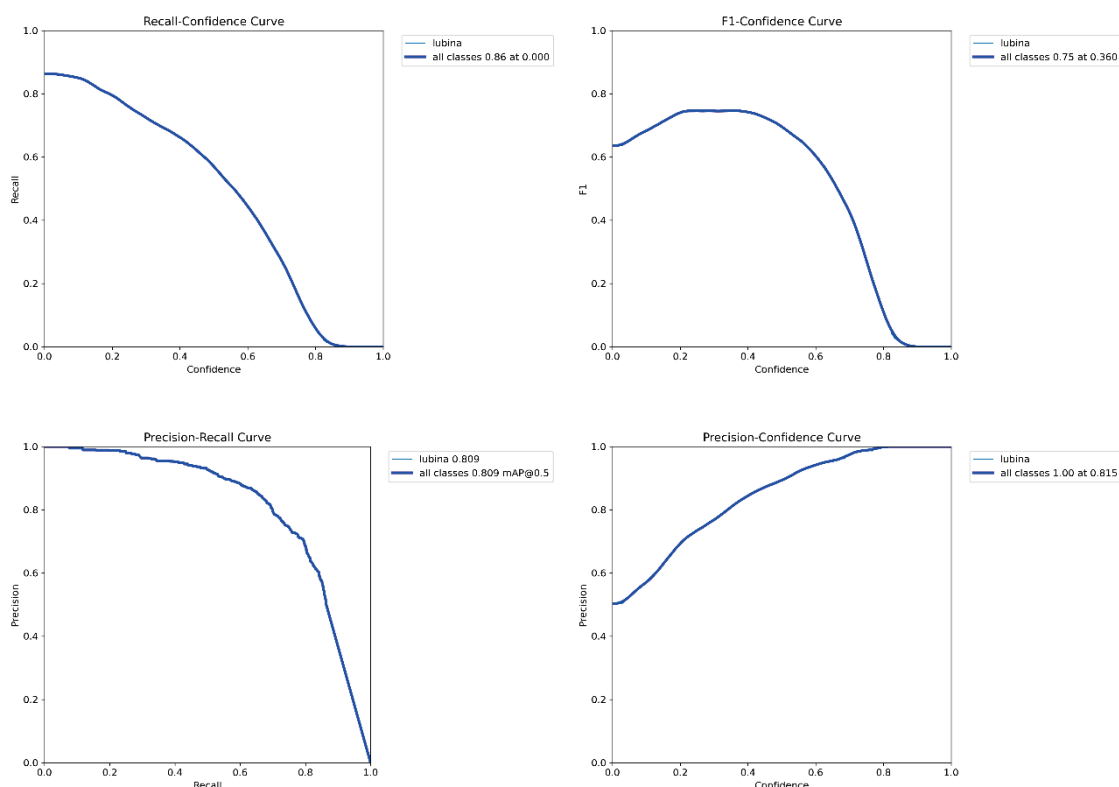


Figura 35 Gráficas de recall y precisión tercer entrenamiento

5.1.4 Cuarto entrenamiento

- **Fecha:** 29 de octubre de 2024
- **Número de imágenes:** 180 imágenes en entrenamiento y 20 en validación.
- **Conclusiones:** En este entrenamiento, manteniendo el modelo YOLOv8x y el tamaño de las imágenes en 1920, se decidió probar con una tasa de aprendizaje inicial más baja, de 0.0001, para ver cómo afectaba al rendimiento. Este ajuste trajo mejoras en métricas como el F1-score y el mAP50, llegando al valor de 0.833. Comparado con el entrenamiento anterior, se redujeron los falsos positivos y negativos, lo que demuestra que el modelo diferenció mejor todos los objetos. Además, las gráficas de pérdida muestran una bajada más consistente, especialmente al inicio del entrenamiento, lo que sugiere que el modelo se optimizó de forma más eficiente. Esto deja claro que ajustar la tasa de aprendizaje puede marcar una gran diferencia en cómo el modelo generaliza y enfrenta condiciones complejas. A pesar del incremento de las métricas de precisión del modelo, a partir de este momento se comenzó a experimentar de manera clara overfitting en el conjunto de validación, al producirse una caída inicial de las pérdidas, hasta que llega un punto en el que vuelven a incrementar. De cara a las siguientes iteraciones de entrenamiento se va a intentar mejorar aún más las métricas para reducir este problema. Para ello, se van a volver a analizar todos los hiperparámetros y se va a consultar documentación para ver cómo evitarlo. Además, se valorará en reducir un poco el conjunto de entrenamiento y pasar imágenes al conjunto de validación para observar si el modelo es capaz de generalizar mejor con este cambio.

■ Gráficas:

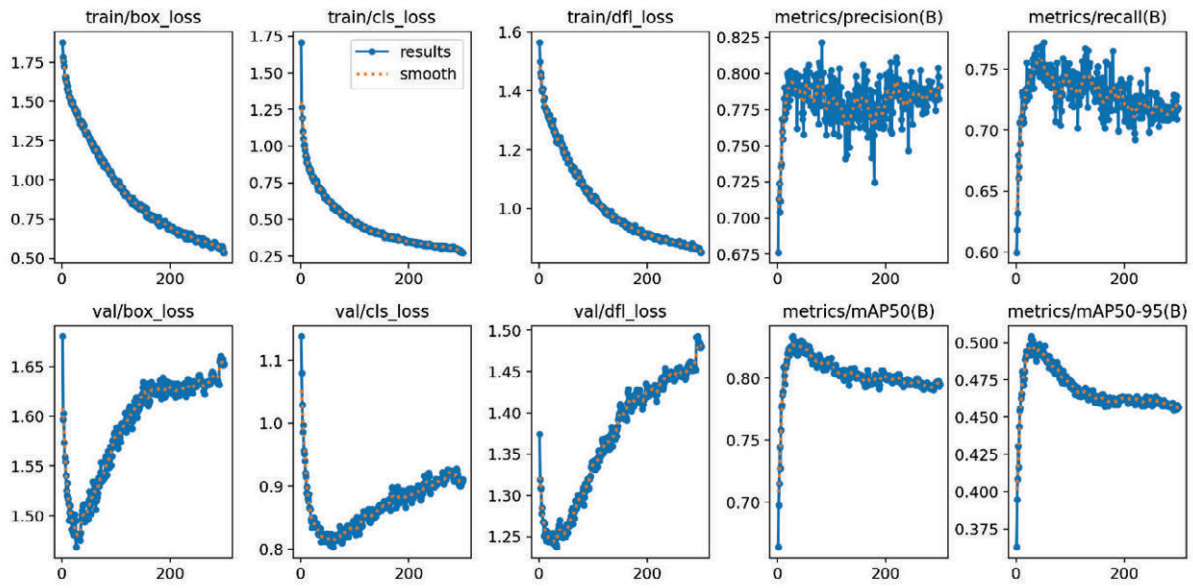


Figura 36 Gráficas de métricas del cuarto entrenamiento

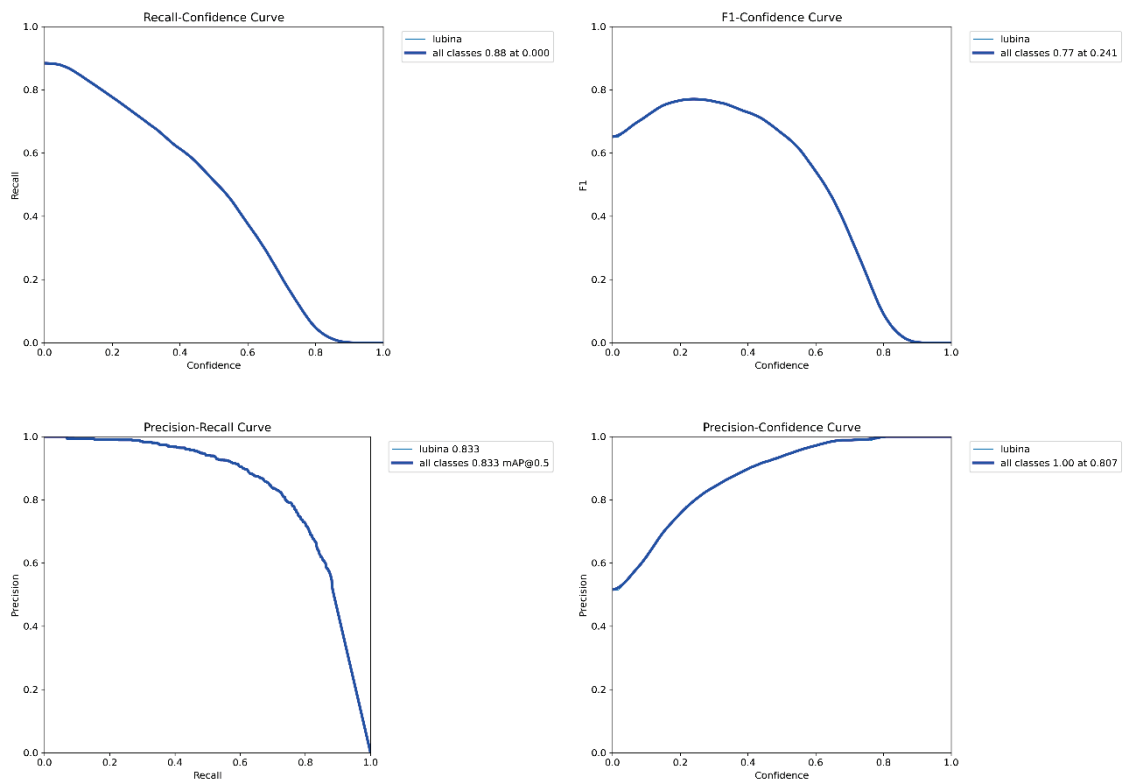


Figura 37 Gráficas de recall y precisión del cuarto entrenamiento

5.1.5 Quinto entrenamiento

- **Fecha:** 4 de noviembre de 2024
- **Número de imágenes:** 160 imágenes en entrenamiento, 30 en validación y 10 de test.
- **Conclusiones:** Tras detectar problemas de sobreajuste en el entrenamiento anterior, se decidió reorganizar el dataset para equilibrar los conjuntos de entrenamiento y validación, ya que esto podía estar intensificando el overfitting. La reorganización se realizó cuidadosamente, asegurando que las imágenes no estuvieran repetidas ni fueran demasiado similares entre ambos conjuntos, con el fin de conseguir una relación equilibrada de los datos y que no se produjese un falseado de los resultados finales del modelo. Además, se ajustaron las tasas de aprendizaje para mitigar el problema: la tasa inicial (lr_0) se estableció en 0.0001 y la tasa final (lr_f) en 0.00001, buscando ralentizar el aprendizaje del modelo y mejorar su capacidad para generalizar. Aunque los resultados mostraron una ligera mejora, el problema de sobreajuste siguió, lo que llevó a considerar que el tamaño del dataset podría no ser suficiente para que YOLOv8x generalizara de forma adecuada, incluso con las técnicas de aumento de datos aplicadas automáticamente por el modelo. Incrementar el número de imágenes se planteó como una solución potencial, pero esto resultaba inviable en ese momento debido al tiempo necesario para etiquetar las imágenes, especialmente considerando la alta densidad de lubinas en cada una de ellas. A pesar de estas limitaciones, las tendencias observadas en las pérdidas hacia el final de las 300 épocas sugirieron que un mayor número de iteraciones podría ser clave para corregir el problema de sobreajuste. Además, los resultados mostraron un avance significativo, alcanzando un mAP50 de 0.858. Este progreso se atribuye principalmente a un cambio crucial en el entrenamiento: aumentar el tamaño de las imágenes (imgsz) de 1920 a 2208, el máximo que admitido por el equipo del CITSEM. Este ajuste mejoró de una manera considerable los resultados, aunque incrementó el consumo de memoria VRAM, lo que obligó a reducir el tamaño del lote (batch size) de 2 a 1 para evitar errores durante el entrenamiento. De cara al próximo entrenamiento, se mantendrán los hiperparámetros actuales, salvo el número de iteraciones que será de 600 o 1000 con el objetivo de ver si realmente los valores de pérdidas disminuyen o se producen intervalos de subida y bajada. En caso de persistir el problema, se deducirá de manera definitiva que se necesitaría un aumento del dataset para evitar este problema.

■ Gráficas:

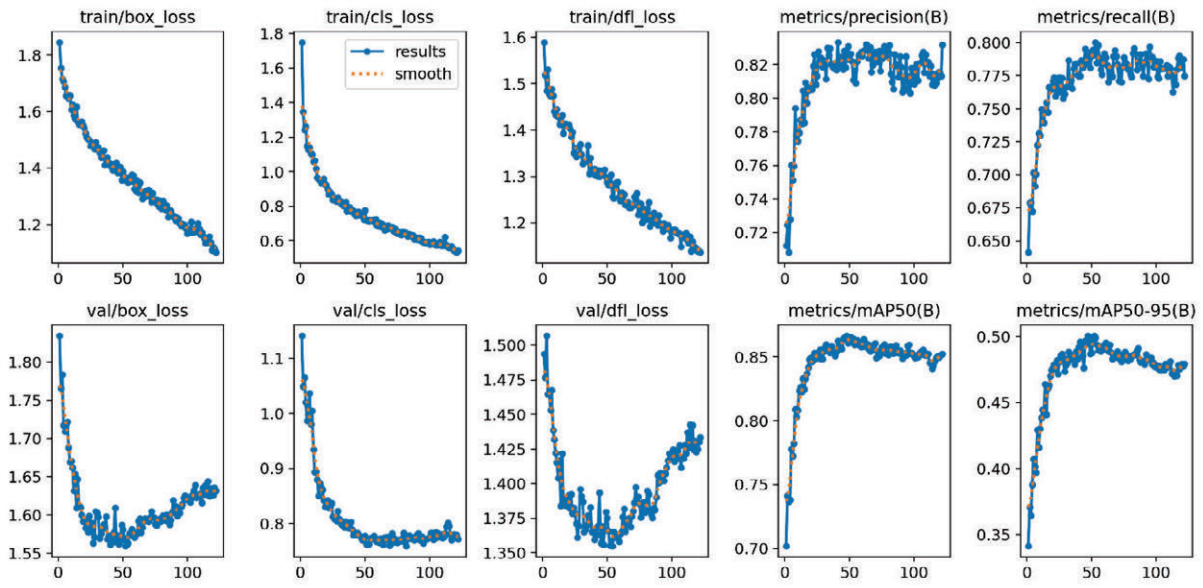


Figura 38 Gráficas de métricas del quinto entrenamiento

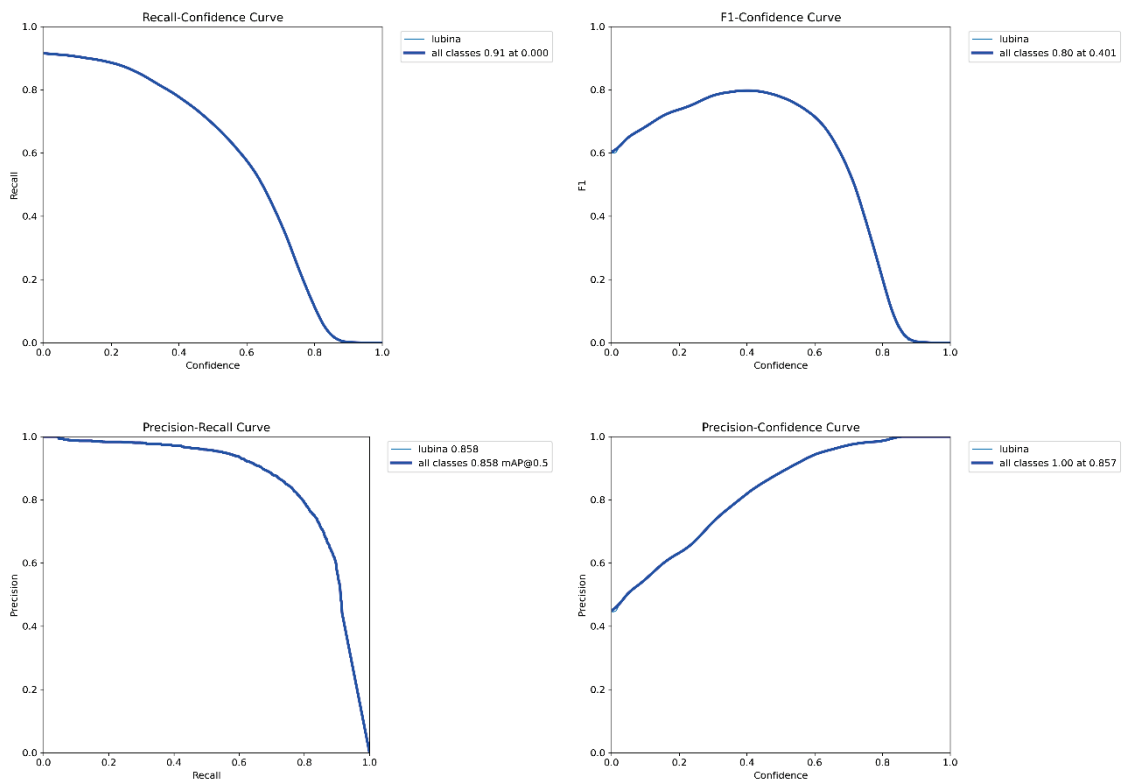


Figura 39 Gráficas de precisión y recall del quinto entrenamiento

5.1.6 Sexto entrenamiento

- **Fecha:** 8 de noviembre de 2024
- **Número de imágenes:** 160 imágenes en entrenamiento, 30 en validación y 10 de test.
- **Conclusiones:** Se decidió extender el entrenamiento a 1000 épocas con la idea de corregir la tendencia de overfitting que se había observado en entrenamientos anteriores. Sin embargo, esta estrategia no funcionó como se esperaba, ya que el sobreajuste continuó incrementándose conforme avanzaba el proceso. Esto quedó reflejado en una disminución de las métricas de precisión y recall, así como en los valores de mAP50 y mAP50-95, mostrando que el modelo no era capaz de aprender nuevos patrones después de cierto punto. Este resultado confirmó lo que ya se había anticipado en análisis previo y es que el dataset con solo 200 imágenes disponibles, no es suficiente para que el modelo pueda generalizar correctamente. Por lo tanto, queda claro que, para solucionar este problema, será necesario ampliar significativamente el número de imágenes. A pesar de estas limitaciones, este modelo ha conseguido los mejores resultados obtenidos hasta ahora. Se consiguió un mAP50 de 0.865, el valor más alto logrado en todos los entrenamientos, junto con mejoras notables en precisión y recall. Este avance se debe a los ajustes que se hicieron a lo largo de las iteraciones previas, como aumentar el tamaño de las imágenes (imgsz) a 2208 y prolongar el entrenamiento a 1000 épocas, lo que permitió aprovechar al máximo los datos disponibles. Dado el tiempo limitado para ampliar el dataset, este modelo será finalmente el que se utilice para la detección de lubinas, tanto en los vídeos de 360° como en los de vista panorámica que agrupa tres vistas en una. Aunque no es un modelo perfecto, ofrece un buen equilibrio entre las limitaciones del conjunto de datos y los resultados obtenidos siendo la mejor opción posible.
- **Gráficas:**

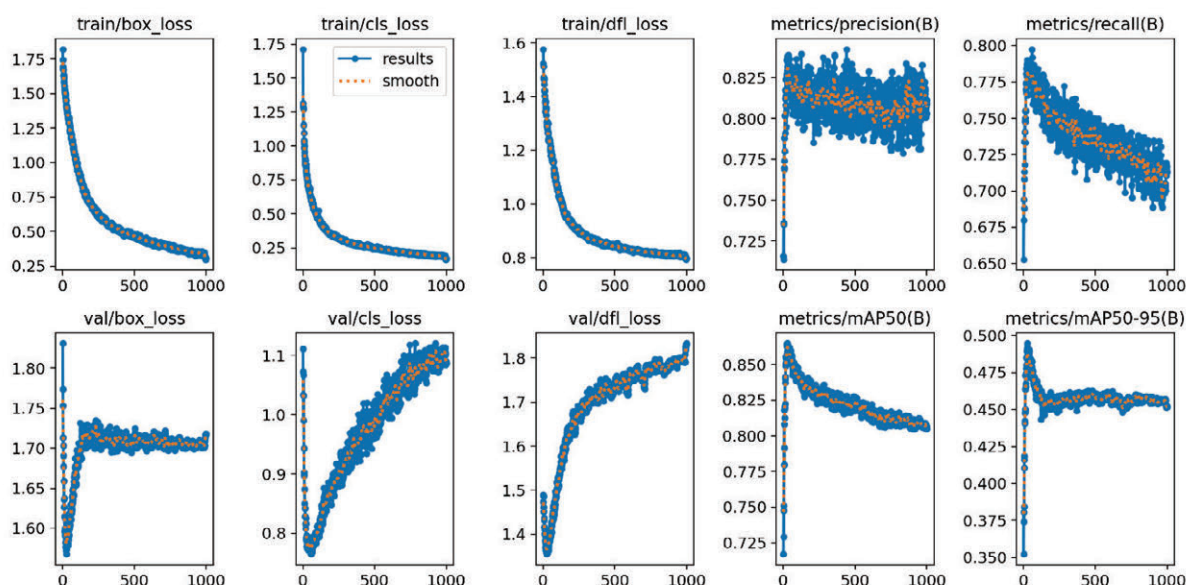


Figura 40 Gráfica de métricas del sexto entrenamiento

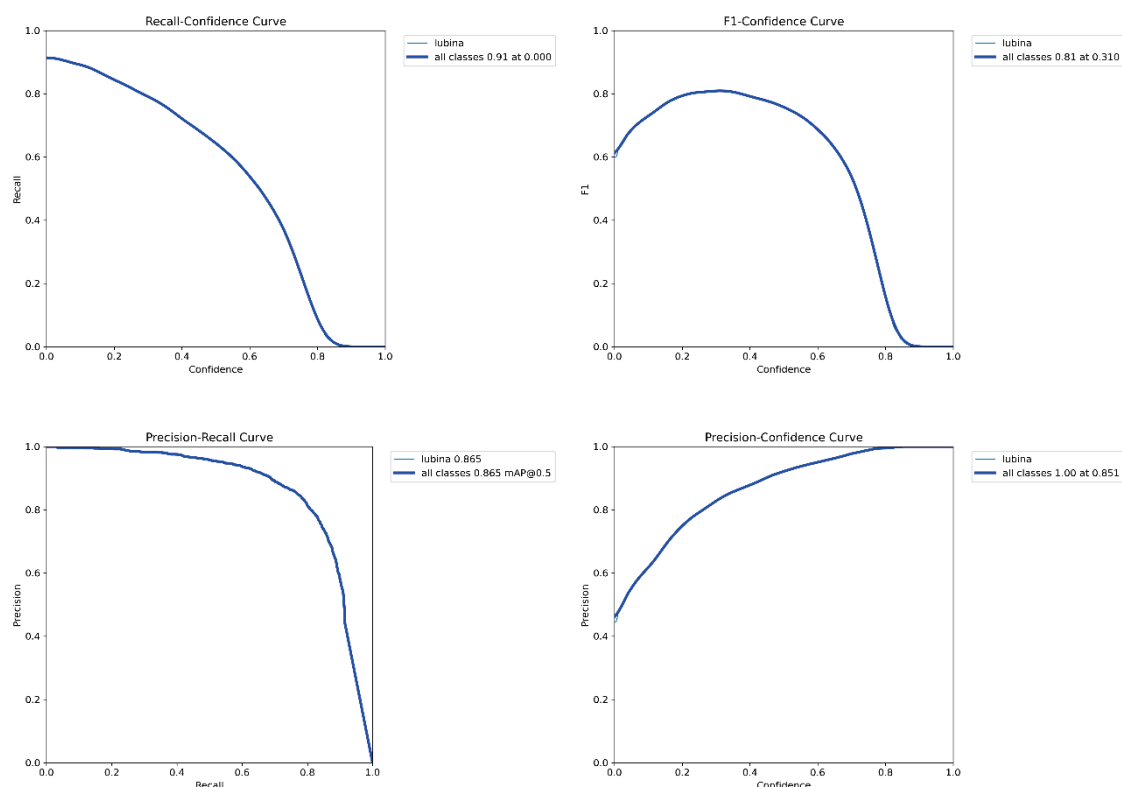


Figura 41 Gráficas de precisión y recall del sexto entrenamiento

5.1.7 Conclusiones etapas de entrenamiento

A lo largo del punto anterior, se han detallado las fases clave de todos los entrenamientos realizados. En cuanto al optimizador, el principal utilizado fue Adam. No obstante, una vez que surgió el problema del sobreajuste, se llevaron a cabo pruebas con AdamW, dado que este optimizador realiza un ajuste más avanzado y podría haber ofrecido mejores resultados. Sin embargo, los resultados obtenidos fueron prácticamente idénticos a los de Adam y no se logró resolver el problema del overfitting, por lo que se decidió continuar trabajando con Adam como la opción más eficiente. En lo que respecta al tamaño de las imágenes (imgsz), se experimentó con varias configuraciones, como 980, 1024, 1280 y 1560, entre otras. Sin embargo, solo al aumentar el tamaño a 1920 y posteriormente a 2208 se observaron mejoras significativas. Este cambio estuvo motivado por las características de las imágenes originales, que tenían un tamaño de 3840x1920. Reducir demasiado el tamaño afectaba negativamente las capacidades del modelo para realizar detecciones precisas, especialmente dado el alto número de elementos presentes en cada imagen. Por ello, aumentar este parámetro se convirtió en un paso crucial para alcanzar los resultados actuales.

El modelo final es capaz de lograr un IoU promedio del 74,39% en el conjunto de test. A pesar de estos logros, se concluye que el próximo paso para mejorar el rendimiento debería ser ampliar el dataset, ya que esta sigue siendo una de las principales limitaciones. Asimismo, sería ideal trabajar con una versión más actualizada de YOLO que ofrezca mejoras en la detección de objetos en condiciones complejas, como baja luminosidad.

Finalmente, se muestran cuatro gráficas que muestran la evolución de los valores de mAP50, mAP50-95, Precision(P) y Recall(R) a lo largo del proyecto:

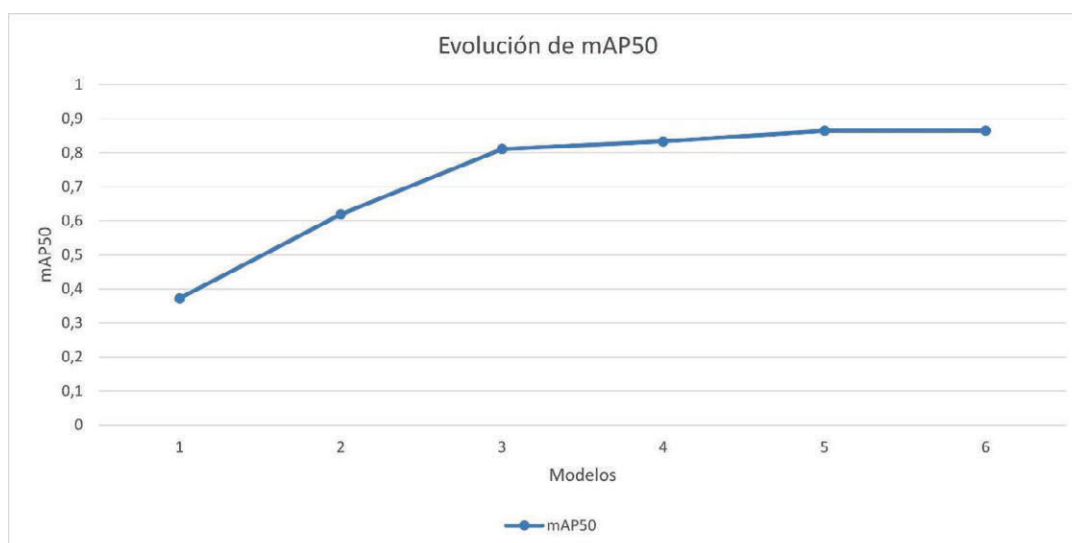


Figura 42 Gráfica Evolución mAP50 para los entrenamientos

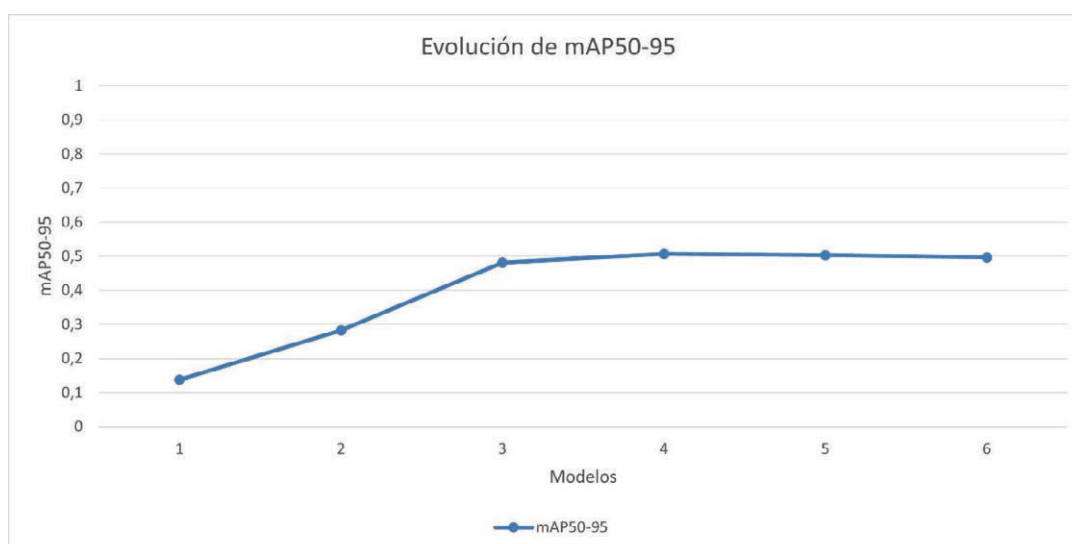


Figura 43 Gráfica Evolución mAP50-95 para los entrenamientos

Como se aprecia estos dos valores de mAP50 y mAP50-95 han ido mejorando con el tiempo según se han ido modificando los valores de tamaño de imagen (imgsz) y las tasas de aprendizaje. De hecho, en el modelo final los valores de precisión y recall son muy altos y oscilan el 80%. Es cierto que el overfitting ha impedido que pudiese aprender durante todas las etapas, lo que hubiera permitido haber obtenido un modelo con valores de precisión mucho mayores, aun así, los resultados obtenidos para la gran densidad de elementos por imagen y las diferentes condiciones lumínicas se consideran mucho más que aceptables al lograr la detección de la inmensa mayoría de las lubinas en cada frame del vídeo, ya sea en vista panorámica o en vista 360°.

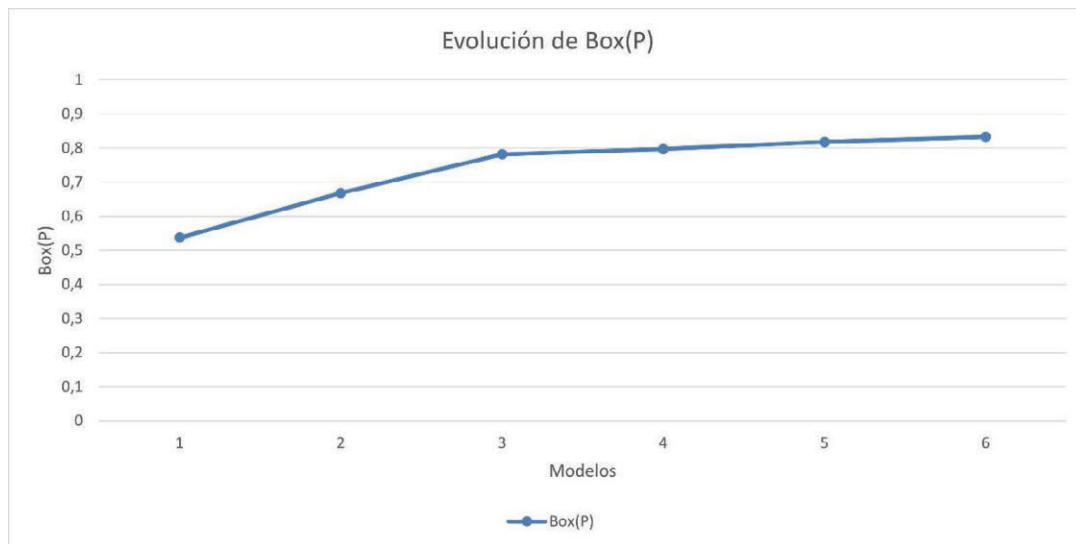


Figura 44 Gráfica Evolución Box(P) en los entrenamientos

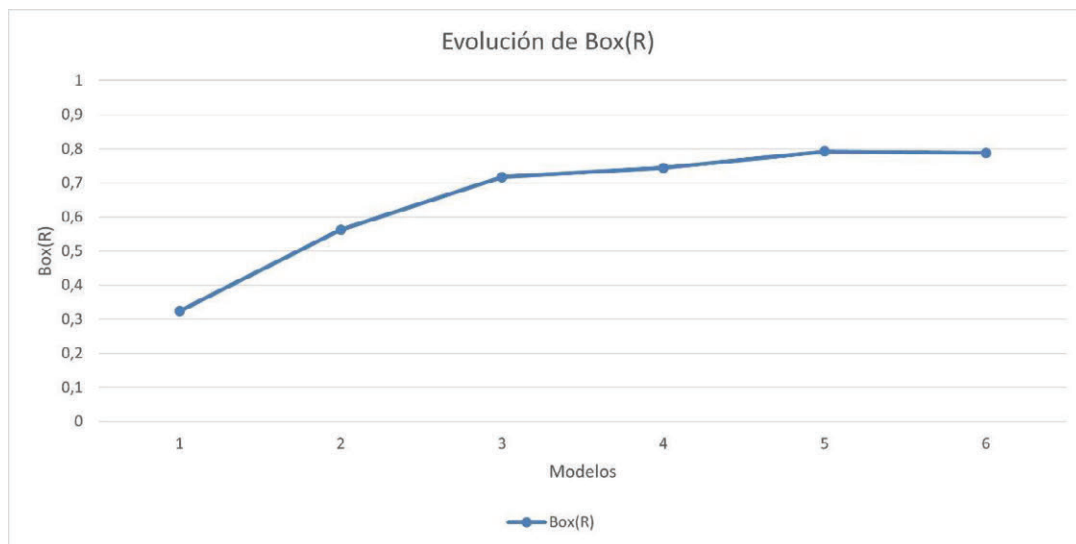


Figura 45 Gráfica Evolución Box(R) en los entrenamientos

Como se aprecia los dos últimos modelos ofrecen resultados muy parecidos; sin embargo, el último ofrece una precisión de 0,833 lo cual es la principal mejora respecto al anterior y de ahí que sea el modelo finalmente escogido. Finalmente, se muestran todos los resultados numéricos recogidos en esta tabla de valores:

Tabla 2 Tabla resumen métricas principales de entrenamiento

Modelo	Box(P)	Box(R)	mAP50	mAP50-95
1	0,537	0,324	0,372	0,138
2	0,668	0,562	0,619	0,283
3	0,782	0,717	0,811	0,481
4	0,797	0,744	0,834	0,507
5	0,818	0,792	0,865	0,503
6	0,833	0,788	0,865	0,497

5.2 Resultados de IoU (Intersection over Union)

Tras completar las etapas de entrenamiento, se decidió evaluar el modelo final. Para ello, una métrica clave en este proceso es el IoU (explicado teóricamente en el apartado 4.3 de este documento), el cual es ampliamente utilizado para medir la precisión en modelos de redes neuronales convolucionales. El valor de IoU compara las predicciones del modelo con las etiquetas de referencia, calculando el porcentaje de superposición entre el área predicha y el área etiquetada en relación con el área total combinada. Es una forma clara de cuantificar si está funcionando bien el modelo. Si bien métricas como el mAP50 y el mAP50-95 están estrechamente relacionadas con el IoU y se obtienen durante las fases de validación y entrenamiento, la evaluación con IoU se realizó para poner a prueba al modelo. Con lo cual, se utilizó un conjunto de datos de otros vídeos de otro tipo, que fueran distintos a los empleados durante el entrenamiento y validación, con el objetivo de observar cómo se comporta el modelo frente a nuevas imágenes, similares a las que podría encontrar en un escenario real. Esta estrategia buscaba no solo comprobar su precisión, sino también evaluar su capacidad de generalización. [45] [50] En la figura 48, se van a mostrar los resultados acumulados para todas las imágenes analizadas:

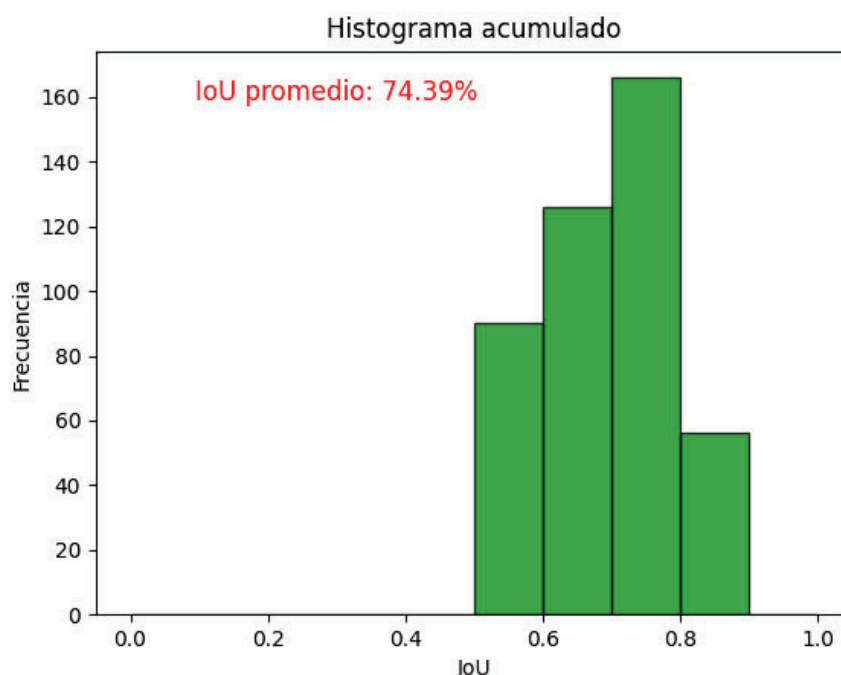


Figura 46 Histograma de valor de IoU acumulado

En cuanto al valor de IoU promedio acumulado, se calcula sumando todos los valores de los histogramas individuales y los representa en un único histograma, teniendo una visión global de la precisión del modelo. El resultado es de 74,39% que es un gran resultado no solo por superar el inicialmente 70% que se estableció como objetivo sino porque a la hora de probar el modelo frente a los vídeos se puede apreciar un buen rendimiento. A continuación, se van a mostrar por cada una de las 10 imágenes tanto el histograma como la imagen resultante con la representación de las etiquetas, que son la representación de las detecciones:

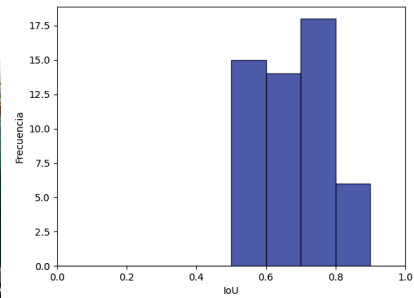
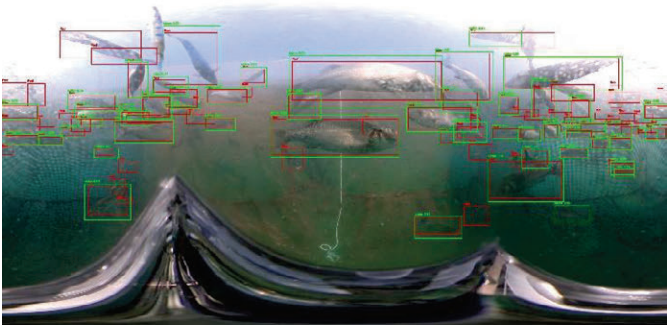


Figura 47 Gráficas de Resultados IoU para la imagen 1

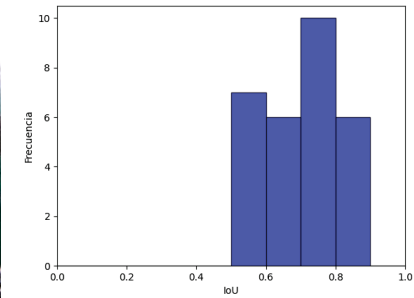
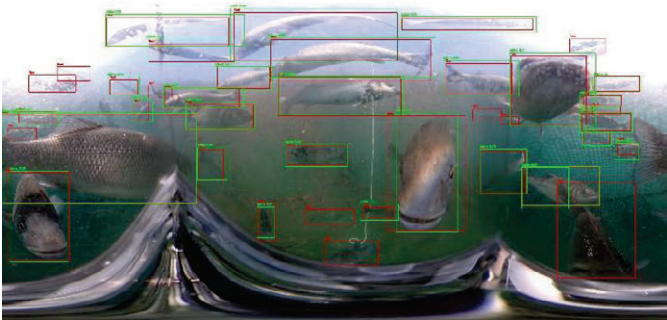


Figura 48 Gráficas de resultados IoU para la imagen 2

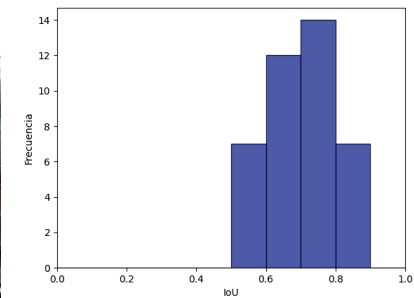
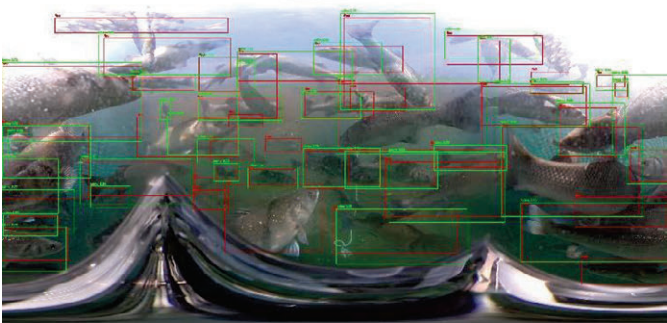


Figura 49 Gráficas de resultados IoU para la imagen 3

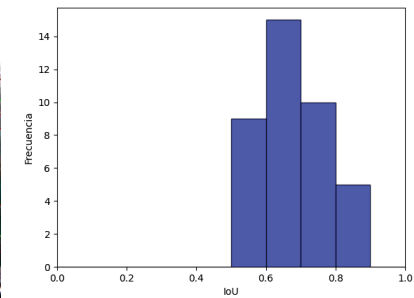
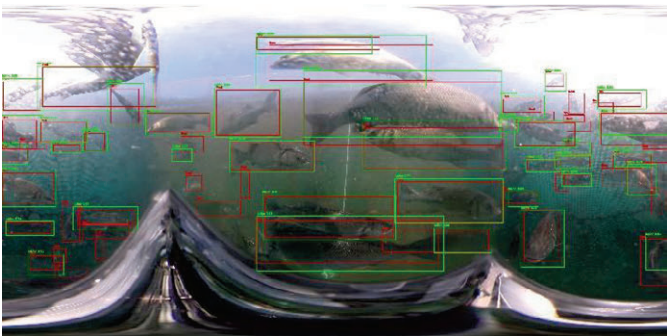


Figura 50 Gráficas de resultados IoU para la imagen 4

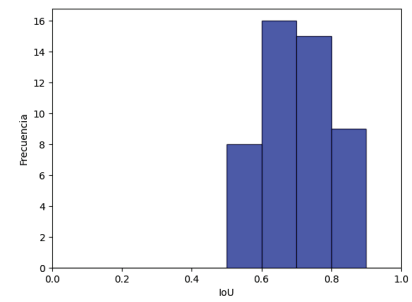
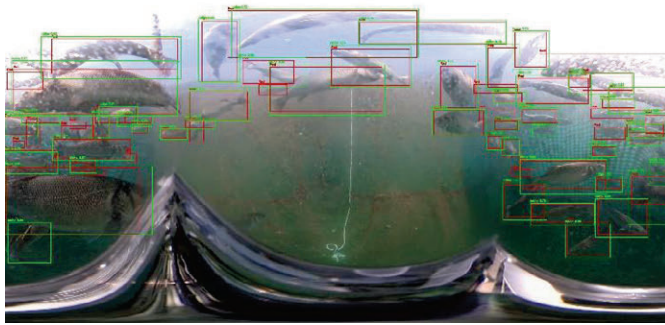


Figura 51 Gráficas de resultados IoU para la imagen 5

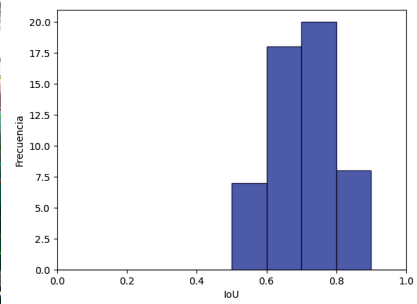
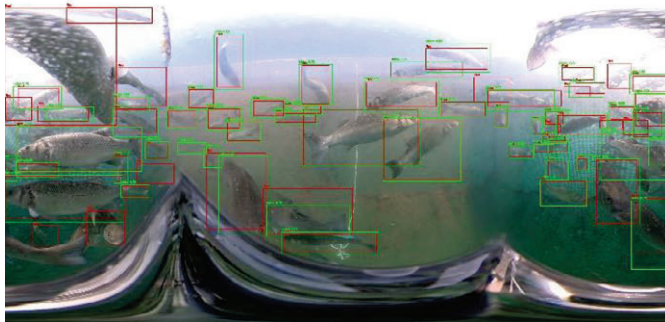


Figura 52 Gráfica de resultados IoU para la imagen 6

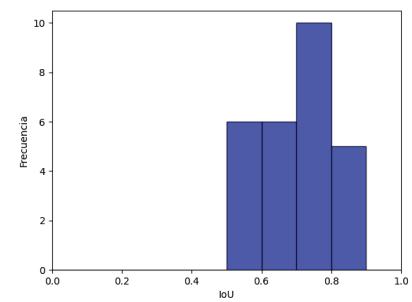
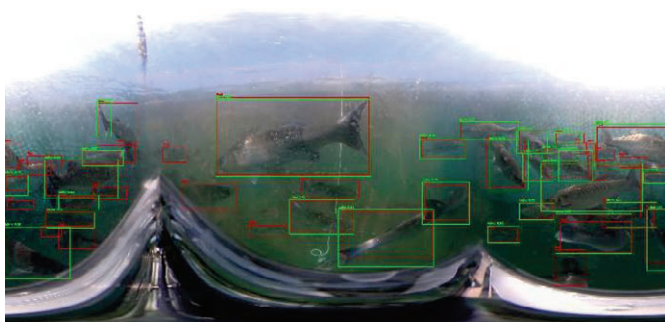


Figura 53 Gráfica de resultados IoU para la imagen 7

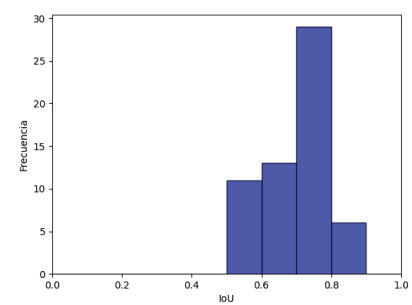
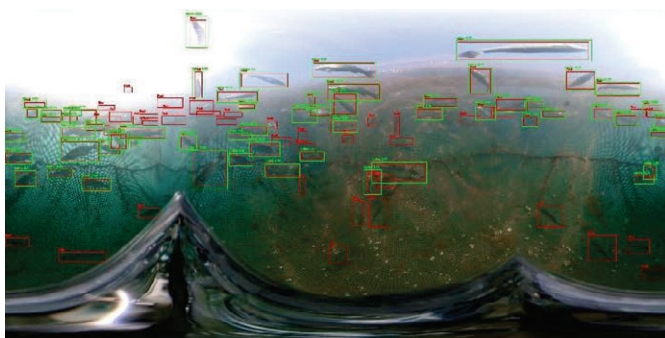


Figura 54 Gráfica de resultados IoU para la imagen 8

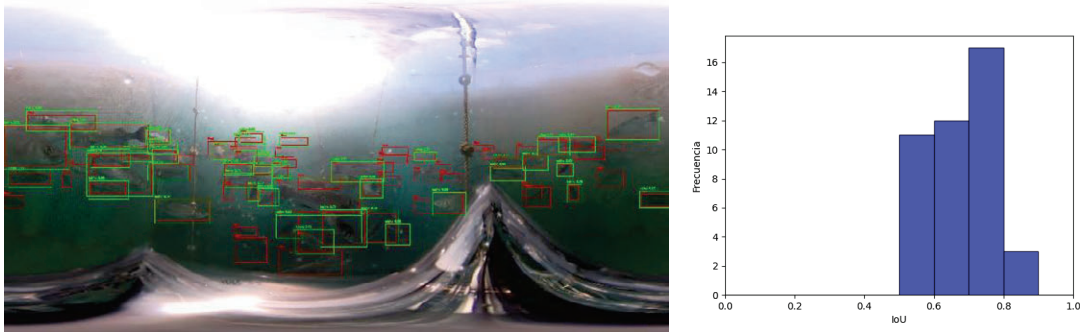


Figura 55 Gráfica de resultados IoU para la imagen 9

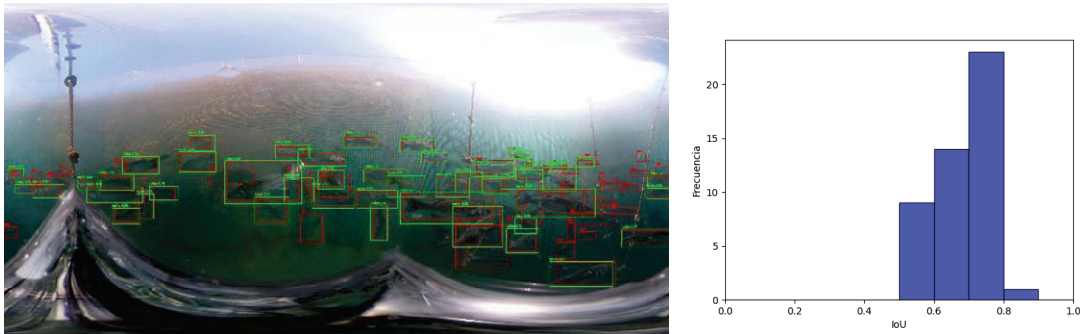


Figura 56 Gráfica de resultados IoU para imagen 10

5.3 Resultados en entornos reales

Una vez finalizadas las dos fases previas, se diseñó una aplicación para probar el modelo en un entorno real. Esta aplicación consta de dos módulos principales: el primero guarda el vídeo con las detecciones realizadas por el modelo, lo que permite observar visualmente los resultados en todo momento y el segundo almacena un archivo de texto (.txt) que registra las detecciones realizadas en cada fotograma. Este archivo sirve como base para analizar, según ciertos comportamientos, las reacciones de las lubinas. A partir de este registro, es posible generar gráficas e interpretar patrones que ayudan a comprender cómo las lubinas responden ante diferentes estímulos en el entorno de pruebas. A continuación, se describen las principales situaciones detectadas durante el análisis.

Situación 1: Perturbación Humana en el medio

En este primer caso, se observó que las lubinas reaccionan ante perturbaciones humanas. Por ejemplo, cuando se genera una sombra o hay algún movimiento cercano que no está relacionado con la alimentación, las lubinas tienden a agruparse en una zona concreta, normalmente en las áreas más alejadas de la superficie. Este cambio en su comportamiento afecta directamente a las métricas registradas. El porcentaje de área ocupada baja drásticamente, ya que los peces se concentran en regiones de los laterales del plano normalmente. Además, el tamaño de las cajas delimitadoras se reduce debido al aumento de la distancia entre las lubinas y la cámara, y esto provoca que el modelo tenga más dificultades para realizar detecciones. Esto también se refleja en una caída notable en el número total de detecciones. En términos de posición, los centroides se desplazan hacia el lugar donde las

lubinas se han agrupado. Un ejemplo de esta reacción puede observarse en la Figura 59, donde la mayoría de los peces se concentran en la parte inferior izquierda del recinto.

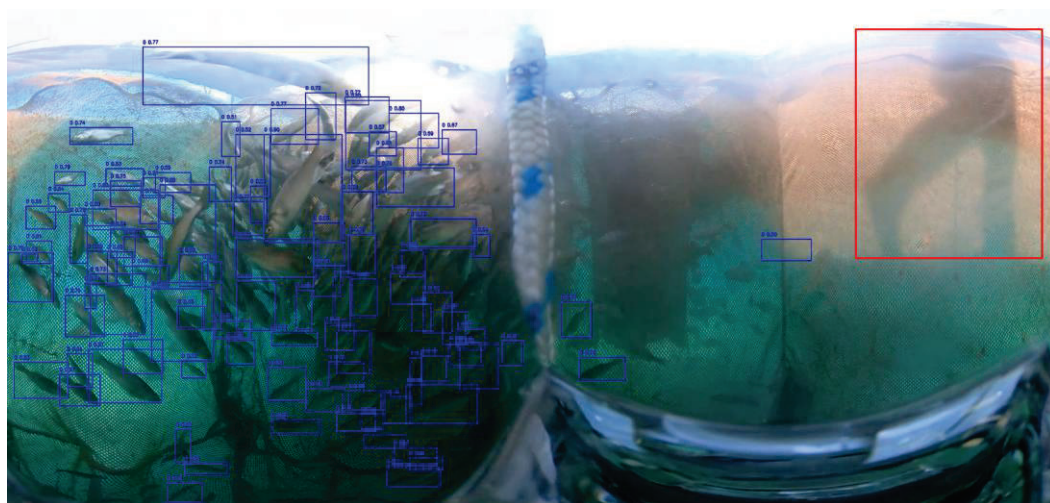


Figura 57 Perturbación por sombra humana en el medio acuático

Situación 2: Fase previa a la alimentación o posterior a una perturbación

En estos casos, cuando las lubinas están a punto de ser alimentadas o después de haber algún tipo de intervención en el medio, tienden a ascender, lo que hace que el porcentaje de área ocupada por las detecciones aumente significativamente durante unos instantes. Esto también lleva a un incremento notable en el número de detecciones, ya que los peces se distribuyen por todo el marco en lugar de concentrarse en una región específica. Además, los centroides se ubican en posiciones más centrales al extenderse por todas las zonas del recinto. Al acercarse a la cámara, el tamaño promedio de las cajas también aumenta de forma considerable. Aunque esta situación ocurre principalmente durante la alimentación, también puede darse si se produce una perturbación en ese momento. Por ejemplo, en algunas ocasiones, mientras las lubinas están siendo alimentadas, una sombra humana provoca que se alejen temporalmente. Sin embargo, una vez que el evento concluye, vuelven a subir para comer. Este comportamiento genera algunos fotogramas en los que las lubinas ascienden de nuevo y ocupan un gran porcentaje del espacio visible, tal y como se aprecia en la Figura 60.

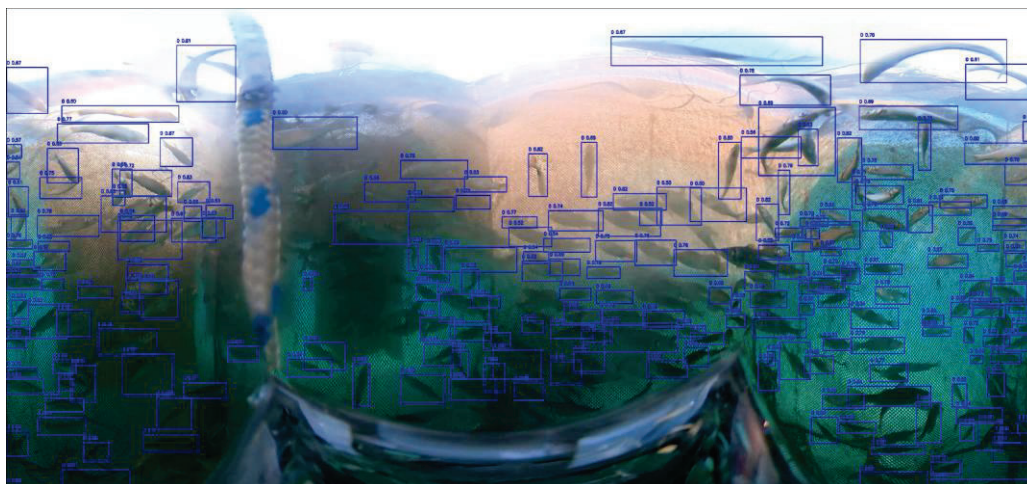


Figura 58 Situación previa a la alimentación o posterior a una perturbación humana

Situación 3: Fase de alimentación

Durante la alimentación, el comportamiento de las lubinas cambia. En esta fase, los peces se agrupan en la zona donde se ha esparcido la comida, lo que produce una disminución en el porcentaje de área ocupada, ya que la concentración ocurre en una región específica. Sin embargo, el tamaño promedio de las cajas delimitadoras aumenta debido a la proximidad de las lubinas a la cámara, y el número de detecciones sigue siendo alto porque el modelo puede capturar bien a los peces agrupados. Los centroides, por su parte, se desplazan hacia la región donde se acumulan las lubinas alrededor de la comida, reflejando esta concentración. Este comportamiento se ilustra claramente en la Figura 61, donde las lubinas se concentran en torno a la zona de alimentación.

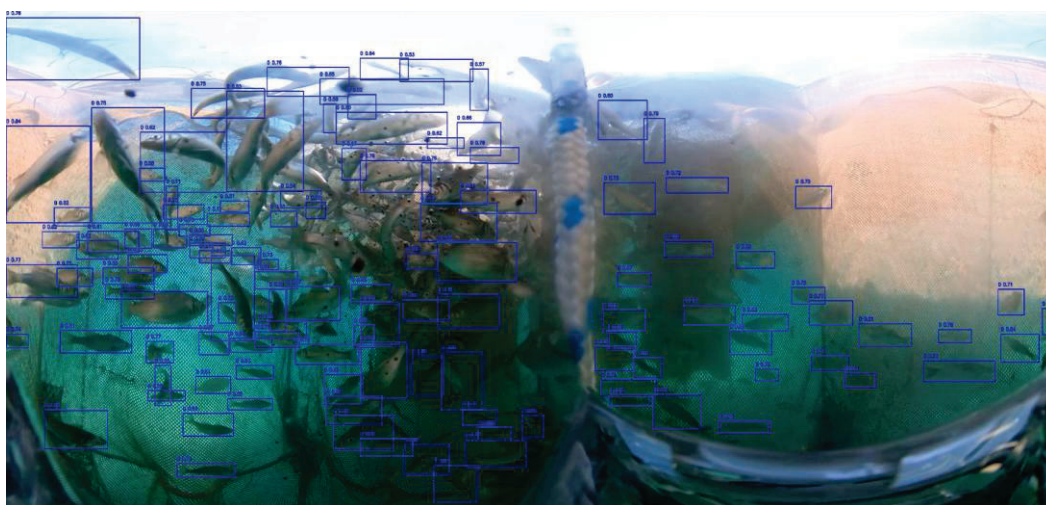


Figura 59 Situación de alimentación de lubinas

Situación 4: Situación estándar

En una situación estándar, que se produce cuando no hay alteraciones en el medio ni se está alimentando a las lubinas, se suelen agrupar en las zonas laterales del recinto, tanto a la izquierda como a la derecha. En este caso el modelo sigue detectando un alto número de peces, ya que, aunque están cerca no se concentran todas en un solo punto, lo que facilita su detección. El porcentaje de área ocupada cambia dependiendo de si las lubinas se acercan o se alejan de la cámara, y el tamaño promedio de las cajas suele ser pequeño debido a la mayor distancia de las lubinas respecto a la cámara. Los centroides también varían según el fotograma, ya que, en algunos momentos se observa una mayor concentración de lubinas en uno de los laterales frente al otro. Esta situación se refleja en la Figura 62, que muestra este comportamiento característico de los peces en ausencia de estímulos externos.

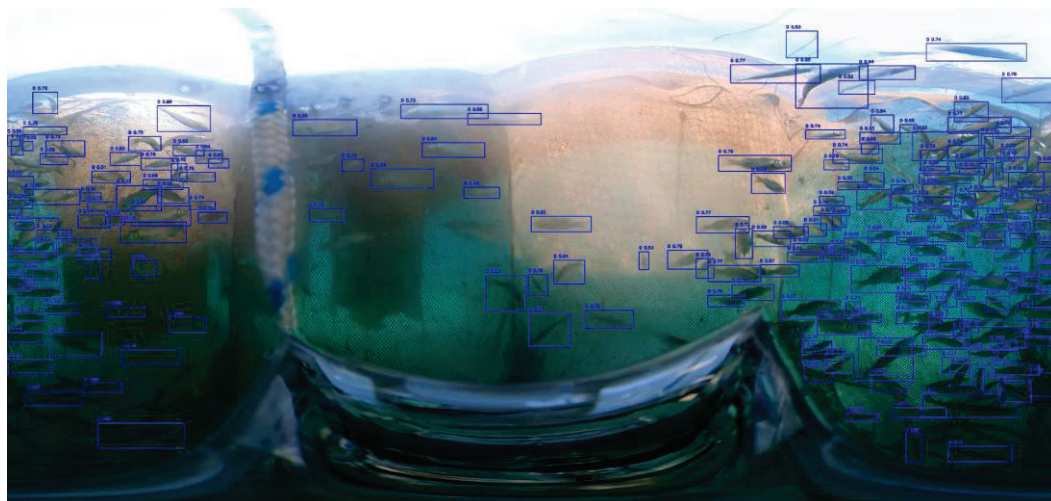


Figura 60 Situación estándar de las lubinas

5.3.1 Detecciones por Frame

Con este gráfico se pretende mostrar el número exacto de detecciones que hace el modelo por cada instante de tiempo. Se puede apreciar que en momentos en los que las lubinas están en una situación de “normalidad”, son en los que no se produce un número elevado de detecciones, mientras que en las situaciones en las que el número de detecciones disminuye se debe en casi todos los casos a algún estímulo como perturbaciones humanas en el medio o el momento de alimentar a las lubinas. Cuando se produce alguno de estos eventos la reacción suele ser la del movimiento de todas las lubinas hacia una zona concreta; por ejemplo, en el caso de haber una perturbación humana tienden a ocupar posiciones de la zona más baja posible del recinto acuícola.

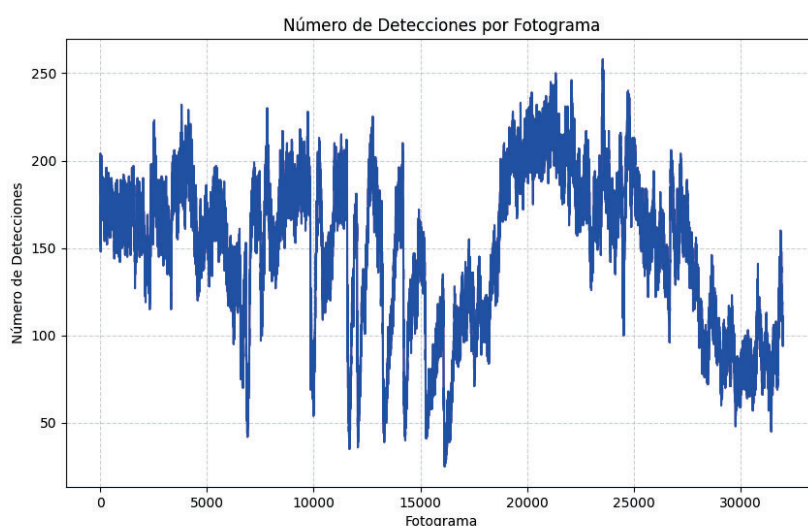


Figura 61 Gráfica de detecciones por fotograma de lubinas en un entorno real

5.3.2 Porcentaje de área ocupada por cajas delimitadoras

El porcentaje de área ocupada se utiliza principalmente para evaluar cómo las lubinas se distribuyen dentro del recinto en cada momento. Cuando estas se reparten por todas las zonas, el porcentaje será más alto, lo que indica que no ocurre ningún evento destacable. Sin embargo, si el porcentaje disminuye, supone que la mayoría de los peces se concentran en una zona específica por alguna razón particular. Este análisis es útil para identificar momentos en los que se producen cambios significativos en la posición de los peces, lo que permite investigar las posibles causas observando el vídeo correspondiente. A lo largo del análisis, se pueden apreciar variaciones notables en el porcentaje de área ocupada, con aumentos y disminuciones en ciertas regiones del recinto. Los momentos de mayor valor se corresponden con el momento previo a la alimentación o los fotogramas posteriores a una intervención en el medio acuático. Lo contrario ocurre durante el transcurso de estos eventos.

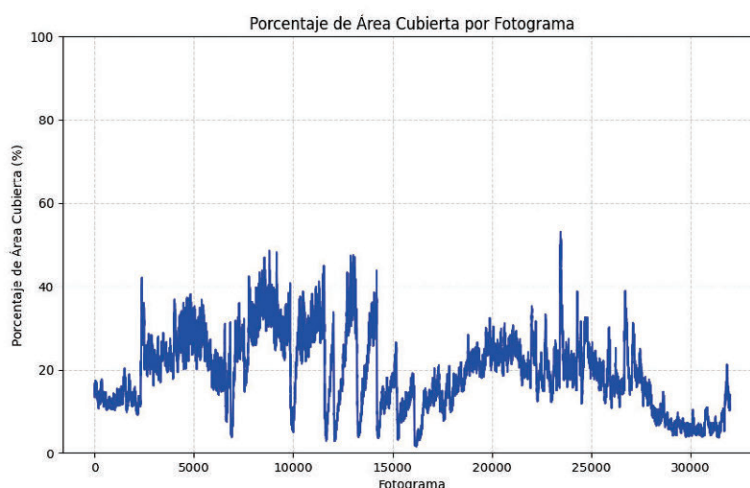


Figura 62 Gráfica de porcentaje de área cajas delimitadoras por cada fotograma

5.3.3 Tamaño medio de las cajas delimitadoras

El tamaño de las cajas proporciona información sobre la distancia de las lubinas con respecto a la cámara. Una caja grande indica que el objeto está muy cerca, mientras que una más pequeña sugiere que se encuentra a mayor distancia. Este análisis busca calcular el tamaño medio de las cajas delimitadoras para identificar si en cada momento la tendencia del grupo de lubinas es acercarse o alejarse de la cámara. Esta información resulta útil como complemento para entender cómo reaccionan las lubinas ante distintas situaciones. En el caso actual, se observan valores altos en el tamaño de las cajas cuando las lubinas tienden a dispersarse y ocupar un mayor espacio en el recinto, acercándose más a la zona de grabación. Esto contrasta claramente con lo que ocurre después de la alimentación o tras una alteración en el medio acuático, donde las cajas delimitadoras son más pequeñas debido a que los peces se alejan de la cámara como reacción instintiva a estos eventos.

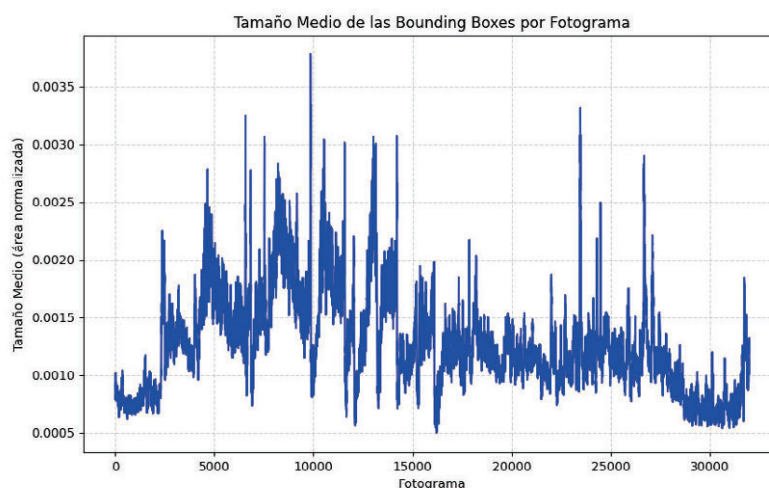


Figura 63 Gráfica tamaño medio cajas delimitadoras en cada fotograma

5.3.4 Centroides X e Y por fotograma

Los valores de los centroides son de utilidad para conocer las tendencias de cambio de posición de las lubinas por cada fotograma, de tal modo que, si se producen grandes cambios de posición, esto se va a ver reflejado en los valores correspondientes en los ejes X e Y (en el apartado 4.6 se muestra el funcionamiento de los ejes al no funcionar como los ejes cartesianos tradicionales). Para este caso, hay dos situaciones que realmente modifican de manera drástica dichos valores y son:

- **Alteración humana.** Esto supone cambios ya que las lubinas tienden a juntarse en una región concreta de la parte inferior que suele ser la parte inferior izquierda, para lo cual el centroide en X sube o baja dependiendo de si la zona de aglutinamiento se produce a la izquierda(valor de X bajo) o a la derecha(valor de X alto) y el centroide Y sube ya que cuanto mayor es su valor, más hacia abajo se concentran las lubinas y viceversa.
- **Fase de alimentación.** El patrón que siguen es el de ascender hacia la superficie y esto hace que el valor del centroide X al igual que en la situación anterior su valor dependa de la zona en la que se produce la acumulación de lubinas y el valor del centroide Y baja al centrarse en zonas superiores de la escena.

De hecho, tal y como se aprecia en la Figura 66, los valores de caídas considerables se corresponden principalmente con eventos en los que o bien se produce una alteración en el medio y las lubinas tienden a refugiarse en la parte inferior izquierda o bien en las fases de alimentación, que, aunque suelen echar comida por la parte izquierda y derecha en la mayoría de las ocasiones se suele dar el alimento por la parte izquierda del vídeo panorámico.

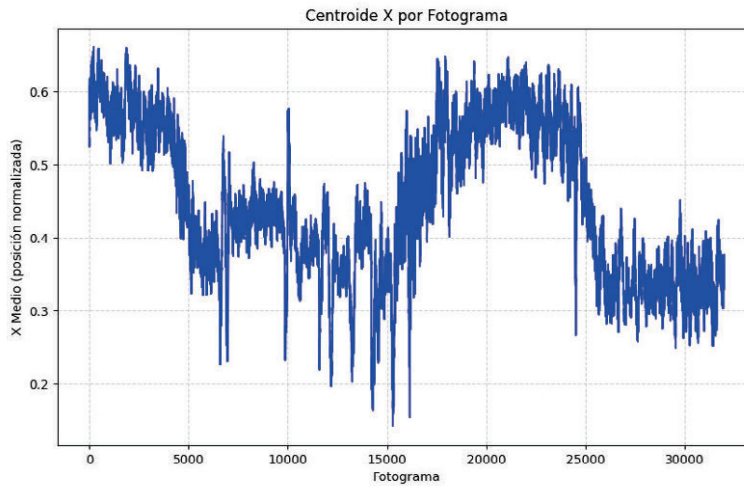


Figura 64 Gráfica centroide X por fotograma

En el caso del centroide Y las fluctuaciones tienen lugar por las mismas razones que en el caso de X, de hecho, en este caso los valores de Y altos suponen que las lubinas se concentran por la parte inferior y se deben por perturbación humana y lo contrario tiene lugar fundamentalmente en los momentos de alimentación de las lubinas.

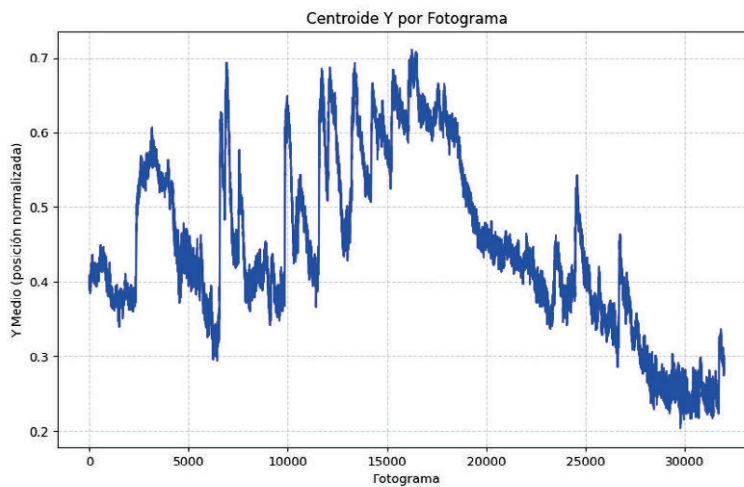


Figura 65 Gráfica centroide Y por fotograma

6. Presupuesto

Este documento presenta el desglose de actividades, recursos y costos necesarios para llevar a cabo el proyecto de manera eficiente. Para la ejecución de este proyecto, se ha llevado a cabo un análisis detallado de las actividades necesarias y los costos asociados, tomando como referencia el salario medio de un recién egresado en Ingeniería de Telecomunicaciones, estimado en 15 € por hora [51]. Este valor ha sido empleado como base para calcular los costos laborales, asegurando un presupuesto realista.

6.1 Recursos Hardware

El proyecto ha requerido el uso de un equipo informático con las siguientes especificaciones:

- Procesador: Intel Core i7-12700, que cuenta con 12 procesadores físicos y 20 lógicos, con una frecuencia base de 2,1 GHz.
- Memoria RAM: 16 GB.
- Tarjeta gráfica: Nvidia RTX A2000 con 12 GB de VRAM.

6.2 Recursos Software

El entorno de trabajo y las herramientas empleadas para el desarrollo del proyecto han sido las siguientes:

- Entorno de desarrollo: Visual Studio Code, una herramienta gratuita que ha permitido la escritura y depuración del código de manera eficiente.
- Licencia de Insta360 Studio. Utilizada para la manipular y exportar vídeos 360°. La versión estándar tiene un costo de 329 €, aunque esta cifra podría incrementarse si se incorporan características adicionales. [52]
- Lenguaje de programación y librerías: Python y todas las librerías empleadas en este proyecto han sido herramientas gratuitas, lo que ha contribuido a optimizar los costos del desarrollo.

6.3 Costos finales

En total, se estiman 352 horas de trabajo (12 ECTS), con un coste de 7.079 € (sin IVA). Con el IVA del 21%, el coste final del proyecto asciende a 8.565,59 €. El desglose queda:

- Costos laborales: 5280 €.
- Costos de hardware y software: 1799 €.
- Total, sin IVA: 7079 €.
- Total, con IVA (21%): 8565,59 €.

Tabla 3 Presupuesto total del proyecto

Activo	Descripción	Duración	Horas	Precio (€/hora)	Precio Total(€)
Investigación y Objetivos					
Planificación del proyecto	Crear un plan de desarrollo	1 semana	8	15	120
Investigación del entorno de trabajo	Consulta de herramientas necesarias para el proyecto	1 semana	8	15	120
Adaptación a las tecnologías de entrenamiento					
Consulta de material	Búsqueda y lectura de documentación	2 semanas	18	15	270
Análisis de modelos CNN	Decidir cuáles es el mejor modelo de CNN a usar	2 semanas	18	15	270
Investigación tipo modelo final	Profundizar en el modelo elegido	3 semanas	27	15	405
Entrenamiento del modelo					
Fases de entrenamiento	Obtener el modelo de red neuronal	4 semanas	55	15	825
Análisis y evaluación del modelo					
Análisis de resultados	Validación de resultados	2 semanas	30	15	450
Desarrollo de la aplicación					
Fase de investigación	Búsqueda de librerías Python que sirvan para solventar el caso de uso	1 semana	8	15	120
Fase de selección de herramientas	Selección de herramientas finales para el desarrollo de la aplicación final	1 semana	8	15	120
Fase de desarrollo	Desarrollo de la aplicación	3 semanas	27	15	405
Elaboración de la memoria final					
Escribir documentación	Escribir memoria final del proyecto	4 meses	118	15	1770
Conclusiones					
Resultados finales	Recopilación de resultados y análisis final	2 semanas	27	15	405
Hardware y Software					
Ordenador	Ordenador con Hardware de última generación	1 unidad			1400
Licencias de Software	Herramientas para desarrollar todo el proyecto (Insta360)				329
Unidad de Almacenamiento	Disco Duro para almacenamiento de versiones				70
FACTURA PROYECTO				Total (SIN IVA)	7079
				Total (IVA 21%)	8565,59

7. Impacto del proyecto

El sistema de detección de lubinas en vídeos 360° mediante redes neuronales convolucionales (CNN) supone un importante paso adelante en la aplicación de la tecnología para afrontar retos globales como la producción sostenible de alimentos, la protección del medio ambiente y el desarrollo económico. Este proyecto no solo mejora los procesos en la acuicultura al automatizar y hacer más precisa la detección de especies, sino que también impulsa prácticas responsables que ayudan a conservar los ecosistemas marinos y a gestionar los recursos de forma más eficiente.

7.1 Impacto social

Este proyecto impulsa una acuicultura más sostenible al usar tecnologías avanzadas que mejoran la seguridad alimentaria y optimizan la gestión de recursos. Al automatizar el monitoreo de lubinas, se reducen tareas manuales como el conteo y análisis de comportamientos, ahorrando tiempo y recursos. Además, fomenta la formación en herramientas tecnológicas, mejorando las competencias y las oportunidades laborales en el sector.

7.2 Impacto de salud

La automatización de las tareas de monitorización reduce el estrés de los peces y minimiza el riesgo de enfermedades. Además, permite detectar problemas en el comportamiento de las lubinas a tiempo, mejorando su bienestar y la calidad de los productos para consumo humano.

7.3 Impacto ambiental

El uso de tecnologías como YOLOv8 mejora la gestión de recursos en acuicultura al optimizar la alimentación y otros procesos, que supone una reducción en el desperdicio y minimiza el impacto ambiental de los desechos. Además, fomenta prácticas sostenibles, ofreciendo una alternativa a la pesca intensiva y ayudando a conservar los ecosistemas marinos.

7.4 Impacto económico

Este sistema tiene el potencial de reducir costos en la acuicultura al automatizar tareas clave como la detección y el conteo de especies. Esto mejora la eficiencia, beneficia tanto a grandes empresas como a pequeños productores y reduce la dependencia de mano de obra en tareas repetitivas. A largo plazo, una mejor gestión de los recursos puede aumentar la rentabilidad y la competitividad en el mercado.

7.5 Impacto tecnológico

Este proyecto combina tecnologías como la visión por computadora, el aprendizaje profundo y los vídeos 360°, promoviendo su adopción en el sector acuícola. También abre la puerta a

avances como la detección de comportamientos anómalos. Además, sienta las bases para futuros proyectos, fortaleciendo el uso de herramientas tecnológicas en la industria.

7.6 Contribución a los Objetivos de Desarrollo Sostenible (ODS)

El proyecto se alinea con varios ODS, de los cuales tenemos: [53]

- **ODS 2: Hambre Cero.** Contribuye a la seguridad alimentaria mediante una gestión más eficiente de la acuicultura. El proyecto promueve la producción sostenible de alimentos para una población en constante crecimiento.
- **ODS 9: Industria, innovación e infraestructura.** La implementación de tecnologías avanzadas como las redes neuronales convolucionales fomenta la innovación y refuerza la infraestructura tecnológica en la acuicultura.
- **ODS 12: Producción y consumo responsables.** Al optimizar recursos y reducir el impacto ambiental, el proyecto promueve prácticas de producción responsables que benefician tanto al medio ambiente como a la sociedad.
- **ODS 14: Vida Submarina.** El enfoque en una acuicultura sostenible ayuda a mitigar la sobreexplotación de los recursos marinos y contribuye a la conservación de los ecosistemas acuáticos.



Figura 66 Objetivos de desarrollo sostenible de la ONU [53]

8. Conclusiones

8.1 Conclusiones

Tras la elaboración de este proyecto final de carrera, se puede afirmar que se han alcanzado con éxito los objetivos propuestos. El desarrollo del sistema de detección de lubinas en vídeos 360° mediante redes neuronales convolucionales, ha permitido no solo abordar de manera eficaz el problema planteado, sino también establecer una base para futuras investigaciones en el campo de la acuicultura.

En la actualidad, debido a las crecientes necesidades de alimentación de la población, es crucial disponer de alternativas sostenibles que no comprometan los ecosistemas acuáticos como ocurre con la pesca intensiva. En este contexto, la visión por ordenador se presenta como una solución prometedora para optimizar procesos mediante el uso del procesamiento de imágenes. Las redes neuronales convolucionales (CNN) destacan dentro de este campo por su capacidad para el manejo de grandes volúmenes de datos y su escalabilidad, algo esencial en acuicultura, donde frecuentemente se observan altas concentraciones de ejemplares. Entre estas redes, YOLOv8 ha demostrado ser una opción de referencia debido a su alto rendimiento y facilidad de uso.

El sistema desarrollado ha demostrado ser capaz de realizar detecciones con una alta precisión en un entorno controlado, logrando un IoU promedio de 74,39%, lo cual es un resultado muy bueno ya que supera el valor de 70% que se planteó inicialmente. Este logro ha sido posible gracias a la correcta aplicación de técnicas de aprendizaje profundo y a un riguroso proceso de creación del dataset, que aseguró la diversidad de las imágenes utilizadas para el entrenamiento y validación del modelo. No obstante, cabe destacar que, aunque los resultados han sido muy positivos, se observaron algunas limitaciones relacionadas con la capacidad del modelo para generalizar en ciertos casos. Esto se debe en parte a un leve overfitting detectado durante los entrenamientos, lo que podría ser mejorado en el futuro mediante la ampliación del dataset. Además, uno de los principales logros del proyecto ha sido la implementación de un flujo de trabajo eficiente, desde la preparación de los datos hasta el despliegue del sistema final. Esto incluye el uso de herramientas de etiquetado locales, como CVAT, para garantizar la privacidad de los datos, y la optimización del modelo mediante técnicas de validación y evaluación de métricas clave, como la intersección sobre unión (IoU), mAP50 y mAP50-95. Todo ello, no solo han permitido obtener un modelo muy preciso, sino que también han asegurado su aplicabilidad en condiciones reales.

Por otro lado, el proyecto ha destacado la importancia del aprendizaje automático en la acuicultura, mostrando cómo estas tecnologías pueden transformar un sector tradicional en uno más eficiente, preciso y sostenible. La capacidad del modelo para detectar lubinas de manera no invasiva elimina la necesidad de intervenciones manuales que podrían alterar el comportamiento natural de las especies o comprometer su estado de salud.

8.2 Trabajos futuros

Como se aprecia los resultados han sido satisfactorios y esto supone que el proyecto servirá de base para investigaciones del comportamiento de lubinas, una vez que se ha logrado su detección. En cuanto a trabajos futuros, tenemos los siguientes:

- **Ampliación del dataset.** Aunque el modelo ha mostrado un buen rendimiento, se ha detectado cierto overfitting en algunos casos. Esto podría resolverse aumentando el tamaño y la diversidad del dataset, lo que permitiría al modelo generalizar mejor y adaptarse a un mayor número de escenarios.
- **Optimización del rendimiento de la aplicación.** Si bien la aplicación actual funciona correctamente, un objetivo clave sería lograr que las detecciones se realicen en tiempo real.
- **Análisis de comportamiento.** Ampliar el alcance del proyecto para incluir el estudio de patrones de comportamiento de las lubinas, como estados de estrés, hábitos de alimentación o interacciones entre las mismas. Este tipo de análisis podría optimizar las condiciones de cría y mejorar el bienestar de las especies.
- **Exploración de nuevas librerías 360°.** Aunque el entorno 360° implementado ha sido efectivo, sería interesante investigar nuevas librerías o herramientas que puedan simplificar y mejorar la creación de entornos inmersivos más pulidos y funcionales.
- **Evaluación de nuevos modelos de CNN.** YOLOv8 ha demostrado ser una excelente opción, pero es importante explorar modelos más recientes en los que el modelo sea capaz de trabajar mejor en entornos con poca luz o condiciones de alta turbidez.
- **Ampliación del sistema a otras especies.** Una posible evolución del proyecto sería entrenar el modelo para detectar y analizar otras especies acuáticas, ampliando su aplicabilidad y contribuyendo a una gestión más completa de los ecosistemas acuáticos.

9. Referencias

- [1] M. L. Monja, «Acuicultura: definición, historia, importancia y tipos,» [En línea]. Available: <https://aquahoy.com/acuicultura-definicion-historia-importancia-clasificacion/>. [Último acceso: 11 Noviembre 2024].
- [2] Acuicultura Today, «La historia y evolución de la acuicultura: Desde las antiguas civilizaciones hasta la industria global en crecimiento,» 19 Marzo 2024. [En línea]. Available: <https://acuicultura.es/la-historia-y-evolucion-de-la-acuicultura-desde-las-antiguas-civilizaciones-hasta-la-industria-global-en-crecimiento/>. [Último acceso: 12 Noviembre 2024].
- [3] «Acuicultura de España,» [En línea]. Available: <https://acuiculturadeespana.es/>. [Último acceso: 12 Noviembre 2024].
- [4] H. Farrow, «¿Qué es la acuicultura?,» [En línea]. Available: <https://www.nationalgeographic.es/medio-ambiente/acuicultura>. [Último acceso: 12 Noviembre 2024].
- [5] S. Zhang, X. Yang, Y. Wang, Z. Zhao, J. Liu, Y. Liu, C. Sun y C. Zhou, «Automatic Fish Population Counting by Machine Vision and a Hybrid Deep Neural Network Model,» 24 Febrero 2020. [En línea]. Available: <https://www.mdpi.com/2076-2615/10/2/364>. [Último acceso: 13 Noviembre 2024].
- [6] M. Cui, X. Liu, H. Liu, J. Zhao, D. Li y W. Wang, «Seguimiento, recuento y análisis del comportamiento de los peces en la acuicultura digital: una revisión exhaustiva,» [En línea]. Available: <https://ar5iv.labs.arxiv.org/html/2406.17800v1>. [Último acceso: 13 Noviembre 2024].
- [7] ATRIA, «Conteo automático por visión artificial,» [En línea]. Available: <https://atriainnovation.com/blog/conteo-automatico-por-vision-artificial/>. [Último acceso: 13 Noviembre 2024].
- [8] Iberdrola, «Qué es el 'machine learning',» [En línea]. Available: <https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico>. [Último acceso: 15 Noviembre 2024].
- [9] Hewlett Packard, «¿Qué es el aprendizaje automático?,» [En línea]. Available: <https://www.hpe.com/es/es/what-is/machine-learning.html>. [Último acceso: 15 Noviembre 2024].
- [10] N. H. En, «Aplicaciones del aprendizaje profundo en la acuicultura,» [En línea]. Available: <https://medium.com/@nguhe/deep-learning-applications-in-aquaculture-5d8e0cdce9b0>. [Último acceso: 15 Noviembre 2024].
- [11] A. G. Fuentes, «¿Está la acuicultura preparada para el aprendizaje computacional?,» [En línea]. Available: <https://www.mispecies.com/reportajes/Esta-la-acuicultura-preparada-para-el-aprendizaje-computacional/>. [Último acceso: 15 Noviembre 2024].

- [12] C. R. China, «Five machine learning types to know,» 20 Diciembre 2023. [En línea]. Available: <https://www.ibm.com/think/topics/machine-learning-types>. [Último acceso: 15 Noviembre 2024].
- [13] ResearchGate, «Standard steps in supervised learning for classification,» [En línea]. Available: https://www.researchgate.net/figure/Standard-steps-in-supervised-learning-for-classification_fig1_353381668. [Último acceso: 15 Noviembre 2024].
- [14] Y. Zhang y Y. Wang, «Unsupervised Machine Learning,» [En línea]. Available: <https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/unsupervised-machine-learning>. [Último acceso: 15 Noviembre 2024].
- [15] GeeksForGeeks, «Tipos de aprendizaje automático.,» 29 Noviembre 2023. [En línea]. Available: <https://www.geeksforgeeks.org/types-of-machine-learning/>. [Último acceso: 15 Noviembre 2024].
- [16] Amazon AWS, «¿Qué es una red neuronal?,» [En línea]. Available: <https://aws.amazon.com/es/what-is/neural-network/>. [Último acceso: 16 Noviembre 2024].
- [17] IBM, «¿Qué son las redes neuronales?,» [En línea]. Available: <https://www.ibm.com/es-es/topics/neural-networks>. [Último acceso: 16 Noviembre 2024].
- [18] Datascientest, «Perceptrón: ¿qué es y para qué sirve?,» 7 Marzo 2022. [En línea]. Available: <https://datascientest.com/es/perceptron-que-es-y-para-que-sirve>. [Último acceso: 16 Noviembre 2024].
- [19] J. Durán, «Todo lo que Necesitas Saber sobre el Descenso del Gradiente Aplicado a Redes Neuronales,» 4 Septiembre 2019. [En línea]. Available: <https://medium.com/metadatos/todo-lo-que-necesitas-saber-sobre-el-descenso-del-gradiente-aplicado-a-redes-neuronales-19bdbb706a78>. [Último acceso: 21 Noviembre 2024].
- [20] G. Garau, «Perceptrón Simple: Definición matemática y propiedades,» [En línea]. Available: <https://blog.damavis.com/perceptron-simple-definicion-matematica-y-propiedades/>. [Último acceso: 16 Noviembre 2024].
- [21] ScienceDirect, «Multilayer Perceptron,» [En línea]. Available: <https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron>. [Último acceso: 16 Noviembre 2024].
- [22] F. Ab Ghani, M. Rivaie, M. Yusoff y M. Puteh, «A Review of Artificial Neural Network Applications in Variants of Optimization Algorithms,» 02 Noviembre 2022. [En línea]. Available: <https://ieeexplore.ieee.org/document/10033339>. [Último acceso: 16 Noviembre 2024].
- [23] R. R. Abril, «El descenso del gradiente,» [En línea]. Available: <https://lamaquinaoraculo.com/deep-learning/el-descenso-del-gradiente/>. [Último acceso: 16 Noviembre 2024].
- [24] I. Goodfellow, Y. Bengio y A. Courville, «Deep Learning,» [En línea]. Available: <https://www.deeplearningbook.org/>. [Último acceso: 16 Noviembre 2024].

-
- [25] Encord, «Mean Square Error (MSE),» [En línea]. Available: <https://encord.com/glossary/mean-square-error-mse/>. [Último acceso: 08 December 2024].
- [26] Great Learning, «Types of Neural Networks and Definition of Neural Network,» 4 Agosto 2022. [En línea]. Available: <https://www.mygreatlearning.com/blog/types-of-neural-networks/>. [Último acceso: 17 Noviembre 2024].
- [27] IBM, «¿Qué son las redes neuronales convolucionales?,» [En línea]. Available: <https://www.ibm.com/es-es/topics/convolutional-neural-networks>. [Último acceso: 17 Noviembre 2024].
- [28] Z. Keita, «Introducción a las redes neuronales convolucionales (CNN),» 15 Abril 2024. [En línea]. Available: <https://www.datacamp.com/es/tutorial/introduction-to-convolutional-neural-networks-cnns>. [Último acceso: 17 Noviembre 2024].
- [29] MathWorks, «¿Qué son las redes neuronales convolucionales?,» [En línea]. Available: <https://es.mathworks.com/discovery/convolutional-neural-network.html>. [Último acceso: 17 Noviembre 2024].
- [30] D. Calvo, «Red Neuronal Convolucional CNN,» 20 Julio 2017. [En línea]. Available: <https://www.diegocalvo.es/red-neuronal-convolucional/>. [Último acceso: 17 Noviembre 2024].
- [31] javi, «La Función Unidad Rectificada Uniforme (RELU): Una Herramienta Esencial para el Aprendizaje Profundo,» 24 Marzo 2023. [En línea]. Available: <https://jacar.es/la-funcion-unidad-rectificada-uniforme-relu-una-herramienta-esencial-para-el-aprendizaje-profundo/>. [Último acceso: 06 Diciembre 2024].
- [32] Nvidia, «Convolutional Neural Network (CNN),» Nvidia, [En línea]. Available: <https://developer.nvidia.com/discover/convolutional-neural-network>. [Último acceso: 18 Noviembre 2024].
- [33] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» 12 Diciembre 2016. [En línea]. Available: <https://ieeexplore.ieee.org/document/7780460>. [Último acceso: 18 Noviembre 2024].
- [34] Ultralytics, «Ultralytics YOLO11 Tasks,» [En línea]. Available: <https://docs.ultralytics.com/es/tasks/>. [Último acceso: 18 Noviembre 2024].
- [35] Ultralytics, «Understanding the Key Steps in a Computer Vision Project,» [En línea]. Available: <https://docs.ultralytics.com/guides/steps-of-a-cv-project/>. [Último acceso: 18 Noviembre 2024].
- [36] Z. Zhang, J. Li, C. Su, Z. Wang, Y. Li, D. Li y Y. Chen, «A method for counting fish based on improved YOLOv8,» [En línea]. Available: <https://www.sciencedirect.com/science/article/pii/S014486092400061X>. [Último acceso: 19 Noviembre 2024].
- [37] D.-A. Pham y S.-H. Han, «Deploying a Computer Vision Model Based on YOLOv8 Suitable for Drones in the Tuna Fishing and Aquaculture Industry,» 16 Mayo 2024. [En línea].

- Available: <https://www.mdpi.com/2077-1312/12/5/828>. [Último acceso: 19 Noviembre 2024].
- [38] X. Ran, B. Li, Y. Zhang, M. Kong y Q. Duan, «Anomalous white shrimp detection in intensive farming based on improved YOLOv8,» [En línea]. Available: <https://www.sciencedirect.com/science/article/pii/S0144860924000840>. [Último acceso: 19 Noviembre 2024].
- [39] Amazon AWS, «¿Qué es Python?,» [En línea]. Available: <https://aws.amazon.com/es/what-is/python/>. [Último acceso: 19 Noviembre 2024].
- [40] P. J. Gallardo, «Los lenguajes de programación más demandados en 2024,» 15 Enero 2024. [En línea]. Available: <https://www.hackaboss.com/blog/lenguajes-programacion-mas-demandados>. [Último acceso: 19 Noviembre 2024].
- [41] D. Bergmann y C. Stryker, «¿Qué es PyTorch?,» 23 Noviembre 2023. [En línea]. Available: <https://www.ibm.com/es-es/topics/pytorch>. [Último acceso: 19 Noviembre 2024].
- [42] J. Gallagher, «What is a Confusion Matrix? A Beginner's Guide.,» 25 Noviembre 2022. [En línea]. Available: <https://blog.roboflow.com/what-is-a-confusion-matrix/>. [Último acceso: 20 Noviembre 2024].
- [43] Ultralytics, «Performance Metrics Deep Dive,» [En línea]. Available: <https://docs.ultralytics.com/guides/yolo-performance-metrics/>. [Último acceso: 20 Noviembre 2024].
- [44] E. Akbarnezhad, «YOLOv8 Projects #1 "Metrics, Loss Functions, Data Formats, and Beyond",» 9 Octubre 2023. [En línea]. Available: <https://www.linkedin.com/pulse/yolov8-projects-1-metrics-loss-functions-data-formats-akbarnezhad/>. [Último acceso: 20 Noviembre 2024].
- [45] D. Shah, «Intersection over Union (IoU): Definition, Calculation, Code,» 30 Mayo 2023. [En línea]. Available: <https://www.v7labs.com/blog/intersection-over-union-guide>. [Último acceso: 21 Noviembre 2024].
- [46] D. Wei, «Demystifying the Adam Optimizer in Machine Learning,» 30 Enero 2024. [En línea]. Available: <https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e>. [Último acceso: 26 Noviembre 2024].
- [47] K. Pykes, «Tutorial del Optimizador AdamW en PyTorch,» 22 Octubre 2024. [En línea]. Available: <https://www.datacamp.com/es/tutorial/adamw-optimizer-in-pytorch>. [Último acceso: 26 Noviembre 2024].
- [48] M. Fernando Jean Dijkstra, «The SGD optimizer,» 13 Abril 2024. [En línea]. Available: <https://medium.com/@fernando.dijkstra/the-sgd-optimizer-4764cc0f0493>. [Último acceso: 26 Noviembre 2024].
- [49] Roboflow, «YOLOv8 PyTorch TXT,» [En línea]. Available: <https://roboflow.com/formats/yolov8-pytorch-txt>. [Último acceso: 04 Diciembre 2024].
- [50] Ultralytics, «Profundización en las métricas de rendimiento,» [En línea]. Available: <https://docs.ultralytics.com/es/guides/yolo-performance-metrics/>. [Último acceso: 07 December 2024].

- [51] Talent, «Salario medio para Ingeniero Telecomunicaciones en España, 2024,» [En línea]. Available: <https://es.talent.com/salary?job=ingeniero+telecomunicaciones>. [Último acceso: 23 Noviembre 2024].
- [52] Insta360, «Insta360 Ace Pro & Insta360 Ace,» [En línea]. Available: <https://store.insta360.com/product/ace-pro>. [Último acceso: 25 Noviembre 2024].
- [53] United Nations, «Objetivos de Desarrollo Sostenible,» [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>. [Último acceso: 30 Noviembre 2024].
- [54] Amazon AWS, «¿En qué consiste el ajuste de hiperparámetros?,» [En línea]. Available: <https://aws.amazon.com/es/what-is/hyperparameter-tuning/>. [Último acceso: 30 Noviembre 2024].
- [55] ProspexAI, «Latest YOLOv8 & YOLOv9 Guide for hyperparameter tuning and data augmentation 2024,» 24 Mayo 2024. [En línea]. Available: <https://medium.com/@tarekeesa7/latest-yolov8-yolov9-guide-for-hyperparameter-tuning-and-data-augmentation-2024-469c69f295e0>. [Último acceso: 30 Noviembre 2024].
- [56] Ultralytics, «Configuration,» [En línea]. Available: <https://docs.ultralytics.com/usage/cfg/>. [Último acceso: 30 Noviembre 2024].
- [57] Ultralytics, «Tamaño del lote,» [En línea]. Available: <https://www.ultralytics.com/es/glossary/batch-size>. [Último acceso: 30 Noviembre 2024].
- [58] S. Kumar, «Data Augmentation Increases Accuracy of your model — But how ?,» 21 Julio 2019. [En línea]. Available: <https://medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722>. [Último acceso: 30 Noviembre 2024].
- [59] Ultralytics, «Ultralytics YOLOv8,» [En línea]. Available: <https://docs.ultralytics.com/es/models/yolov8/>. [Último acceso: 30 Noviembre 2024].
- [60] Insta360, «Insta360 X3 & RS: Pro Tips & Best Settings for High-Quality 360 Videos,» 26 Noviembre 2024. [En línea]. Available: <https://www.insta360.com/blog/tips/insta360-one-x2-one-r-pro-tips.html>. [Último acceso: 03 Diciembre 2024].
- [61] A. Afdhal, K. Saddami, S. Sugiarto, Z. Fuadi y N. Nasaruddin, «Real-Time Object Detection Performance of YOLOv8 Models for Self-Driving Cars in a Mixed Traffic Environment,» 19 Septiembre 2023. [En línea]. Available: <https://ieeexplore.ieee.org/document/10249521>. [Último acceso: 20 Noviembre 2024].

Anexo

A.1 Hiperparámetros de YOLOv8

Al entrenar modelos de machine learning, cada modelo tiene un conjunto específico de hiperparámetros y para encontrarlos se deben realizar varias pruebas hasta encontrar la combinación óptima para un determinado caso de uso. [54] De entre todos los hiperparámetros destacan los siguientes: [55] [56]

- **Número de épocas (epochs).** El número de épocas de entrenamiento depende del conjunto de datos. Se recomienda que se utilicen 300 épocas, aunque en caso de producirse overfitting o sobreajuste, es preferible o bien reducir el número de épocas o utilizar la técnica que se denomina “early stopping”, que se utiliza para detener el entrenamiento de manera anticipada cuando ya no se consigue después de un número de épocas que el modelo mejore sus resultados.
- **Tamaño de la imagen (imgsz).** El tamaño de imagen indica el cambio de dimensiones previo al entrenamiento. De manera predeterminada su valor es 640. En el caso de haber imágenes de alta resolución en los conjuntos de datos se puede o bien aumentar su valor o aplicar el enfoque de ventana deslizante, dividiendo estas imágenes en fragmentos más pequeños.
- **Tamaño del lote (batch_size).** Cuanto mayor sea su valor mejor, pero está limitado a las capacidades del equipo. Su función es la de dividir el conjunto de datos en lotes más pequeños y así se puede agilizar el proceso de entrenamiento. [57]
- **Patience.** Se utiliza para evitar el overfitting y lo que hace es parar el entrenamiento cuando no se producen mejoras después de un número determinado de épocas.
- **Optimizador.** Se indica el optimizador a utilizar. Destacan Adam y AdamW.
- **Tasa de aprendizaje inicial(lr0).** Especifica la velocidad de actualización de los pesos. Si se aumenta el valor, las oscilaciones son más altas y al disminuirlo, la convergencia es más lenta. Se recomiendan valores de 0.001 o 0.01 de partida.
- **Tasa de aprendizaje final (lrf).** Tasa de aprendizaje final del modelo.
- **Momentum.** Ayuda al modelo a salir de mínimos locales y su ajuste puede mejorar la convergencia. Su valor por defecto es 0.937.
- **Weight_decay.** Regularización utilizada para evitar el sobreajuste u overfitting. Su valor inicial es 0.0005.
- **Warmup_epochs.** Se realizan unas épocas previas al inicio del entrenamiento. Se actualiza la tasa de aprendizaje inicial. Su valor es de 3 por defecto.
- **Warmup_momentum.** Se ajusta lentamente hasta converger en el valor que se ha estipulado para el entrenamiento. Su valor inicial es de 0.8.
- **Data Augmentation(Aumentado de datos).** YOLOv8 aumenta las imágenes para mejorar los resultados del entrenamiento. Para ello, se realizan todo tipo de técnicas como cambio de luminosidad en las imágenes o realizar giros de la imagen de manera aleatoria y cambiar el tamaño de los objetos para ser capaz de detectar objetos a

diferentes distancias. En la siguiente figura se va a ilustrar de manera visual cómo funciona el proceso de aumentado de datos:

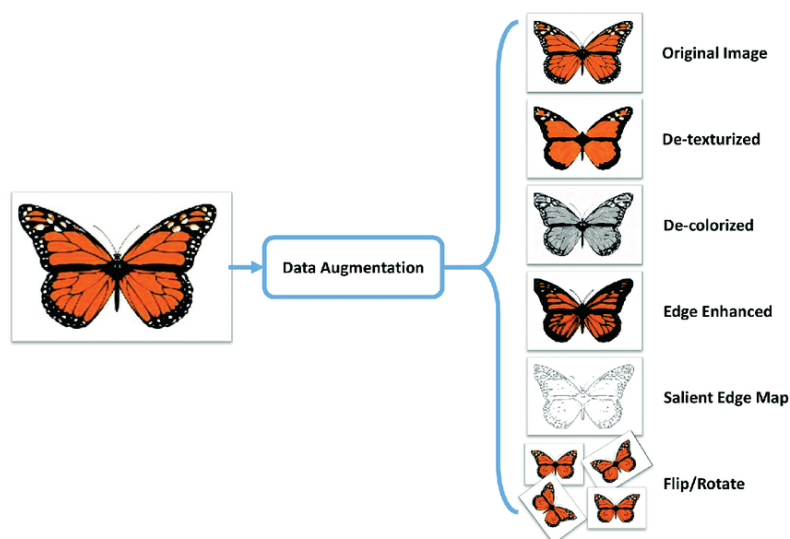


Figura 67 Ejemplo de Data Augmentation [58]

En Data Augmentation, destacan los siguientes hiperparámetros: [56]

- **Hsv_h.** Ajusta el valor de Hue de la imagen. Es muy útil para que el modelo generalice mejor en diferentes condiciones lumínicas. Su valor por defecto es 0.015.
- **Hsv_s.** Ajusta el valor de saturación, lo cual permite simular diferentes condiciones de iluminación. Su valor por defecto es 0.7.
- **Hsv_v.** Modifica el valor del brillo de la imagen para que el modelo generalice mejor en distintas condiciones de luz. Inicialmente, tiene un valor de 0.4.
- **Degrees.** Rota la imagen para mejorar el reconocimiento de objetos en diferentes ángulos. Su valor por defecto es 0.
- **Translate.** Mueve la imagen en vertical y horizontal. Su valor inicial es 0.1.
- **Scale.** Modifica el tamaño de los objetos para simular diferentes distancias. Por defecto es 0.5.
- **Fliplr.** Rota la imagen para aprender de objetos simétricos. Por defecto es 0.5.
- **Mosaic.** Combina cuatro imágenes de entrenamiento en una sola, simulando interacciones de objetos. Es muy útil para escenas complejas. Inicialmente, tiene valor 1.
- **Erasing.** Borra partes de la imagen durante el entrenamiento de clasificación, para que se centre en otras características. Su valor inicial es 0.4.
- **Crop_fraction.** Recorta la imagen de clasificación a una fracción de su tamaño para resaltar las características principales y adaptarse mejor a las escalas de los objetos. Por defecto, tiene valor 1.

A.2 Versiones de YOLOv8

YOLOv8 cuenta con diferentes versiones que dependiendo del contexto se usará una u otra. En la figura 61 se muestra la relación entre el número de parámetros y precisión, junto con la latencia necesaria para cada modelo. Se aprecia que YOLOv8x es el que mayor precisión tiene al contar con un mayor número de parámetros, pero esto se ve afectado en el mayor tiempo que necesita para realizar la detección. [59]

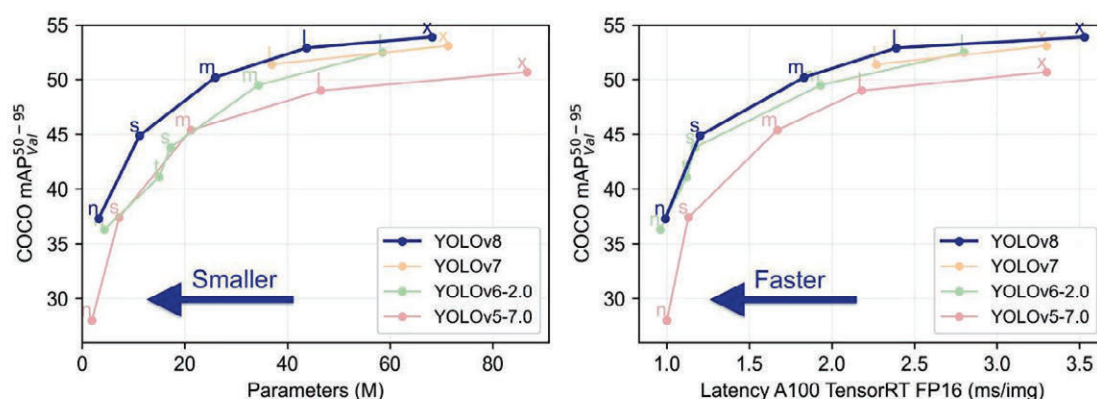


Figura 68 Gráficas de relación entre parámetros y latencia frente a precisión [59]

Como se aprecia, el modelo YOLOv8n (versión nano) es la versión más simple y rápida, pero también la menos precisa, lo cual dependiendo del proyecto puede ser un aspecto a tener en cuenta. En la figura 62, se muestran las capacidades de los modelos:

Modelo	Nombres de archivo	Tarea	Inferencia	Validación	Formación	Exportar
YOLOv8	yolov8n.pt yolov8s.pt yolov8m.pt yolov8l.pt yolov8x.pt	Detección	✓	✓	✓	✓
YOLOv8-seg	yolov8n-seg.pt yolov8s-seg.pt yolov8m-seg.pt yolov8l-seg.pt yolov8x-seg.pt	Segmentación de instancias	✓	✓	✓	✓
YOLOv8-pose	yolov8n-pose.pt yolov8s-pose.pt yolov8m-pose.pt yolov8l-pose.pt yolov8x-pose.pt yolov8x-pose-p6.pt	Pose/Puntos clave	✓	✓	✓	✓
YOLOv8-obb	yolov8n-obb.pt yolov8s-obb.pt yolov8m-obb.pt yolov8l-obb.pt yolov8x-obb.pt	Detección orientada	✓	✓	✓	✓
YOLOv8-cls	yolov8n-cls.pt yolov8s-cls.pt yolov8m-cls.pt yolov8l-cls.pt yolov8x-cls.pt	Clasificación	✓	✓	✓	✓

Figura 69 Tareas compatibles con modelos YOLOv8 [59]

Para el caso de estudio actual, se realiza detección de objetos en imágenes y al ser etiquetas que no tienen ángulo, se utiliza la tarea Detección sin orientación, en la cual vemos que todos los modelos son útiles para esta función. YOLOv8n es la más ágil de todas, pero en este caso al centrarnos fundamentalmente en precisión y no tanto en velocidad se prefiere utilizar la versión más alta posible y con ello la versión más adecuada sería YOLOv8x. [59]

Manual de usuario

En este apartado se van a describir los pasos a seguir para el exportado del vídeo 360°, siguiendo por la técnica empleada para la obtención de imágenes de manera aleatoria del vídeo final, el etiquetado de las imágenes, la elaboración de los conjuntos de datos y el funcionamiento de la aplicación final.

A.3 Exportado del vídeo 360°

Los videos en formato 360° no se guardan en un único archivo, sino que están formados por dos: un archivo .LRV y otro .VID. El .LRV es una versión de baja resolución que se utiliza principalmente para generar vistas previas rápidas, mientras que .VID contiene el video principal en alta calidad. Por esta razón, para trabajar con el material original, siempre se debe abrir el archivo .VID. [60]

Para exportar un vídeo en Insta360Studio, hacemos los siguientes pasos:

1. Hacer doble click sobre el vídeo que queremos abrir que tenga extensión .VID.
2. Pulsa el botón “Exportar”, ubicado en la parte superior del panel, resaltado en amarillo. Esto abrirá un menú desplegable:

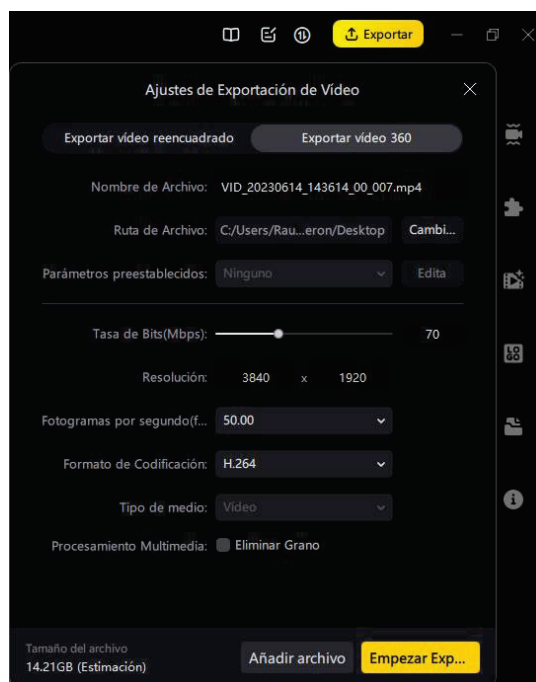


Figura 70 Desplegable de opciones de exportado de vídeos 360°

3. Seleccionar la opción “Exportar vídeo 360”.
4. Definir el nombre del archivo y la ubicación donde se desea guardar. El resto de los campos se dejan tal y como salen por defecto.
5. Pulsar en “Empezar Exportación”.

Una vez completado este proceso, se pasa a la etapa de obtención de imágenes de manera aleatoria. Para ello, se utiliza un programa que se encarga de extraer imágenes a partir de los

videos, basándose en las rutas especificadas en un archivo Excel. Este archivo debe contener las ubicaciones de todos los videos de los que se obtendrán imágenes. A continuación, se muestra cómo debe estar estructurado el archivo Excel:

	A	B	C	D
1				
2		W:\RaulCalderon\VID_20230614_153438_00_009.mp4		
3		W:\RaulCalderon\VID_20231011_164530_00_009.mp4		
4		W:\RaulCalderon\VID_20240312_151115_00_008.mp4		
5		W:\RaulCalderon\VID_20240312_125900_00_008.mp4		
6		W:\RaulCalderon\VID_20230614_181531_00_004.mp4		
7		W:\RaulCalderon\VID_20240313_103747_00_005.mp4		
8		W:\RaulCalderon\VID_20230615_131545_00_005.mp4		
9		W:\RaulCalderon\VID_20230615_134456_00_006.mp4		
10				
11				

Figura 71 Archivo Excel con los paths de los vídeos

Una vez que todo está preparado, se debe indicar al programa la ubicación del archivo Excel y una vez que se tenga todo listo, se ejecuta el programa(`random_frames.py`), el cual genera una carpeta con las imágenes extraídas de cada uno de los vídeos que figuren en el Excel. Es importante procesar cada vídeo de manera individual para evitar mezclar las fotos y mantener clara la separación entre los conjuntos de datos de entrenamiento y validación. Esto es fundamental para garantizar que no existan imágenes similares en ambos conjuntos, evitando así sesgos que puedan comprometer la fiabilidad de los resultados obtenidos.

A.4 Etiquetado de imágenes

En este apartado se va a mostrar el proceso a seguir para el etiquetado de imágenes con la herramienta CVAT. Antes que nada, debemos acceder al CVAT local del CITSEM, introduciendo en un navegador la ruta del servidor NAS , una vez que accedamos nos pedirá las credenciales de acceso, la cual serán proporcionadas por el profesor encargado. Una vez que hemos accedido a la herramienta, se pasa a crear todo lo que es necesario para el etiquetado de imágenes. Los pasos a realizar son:

1. Crear un proyecto nuevo.

- Dentro del menú “Projects” de la parte superior, haz clic en el desplegable “+” y selecciona la opción “Create a new project”.
- Una vez que nos encontramos en el menú principal de los proyectos, indicamos el nombre del proyecto y le damos el nombre a la etiqueta que en este caso se decide llamar “lubina”.
- Una vez completados estos pasos, pulsa el botón “Submit and Open” para finalizar la creación del proyecto y proceder al siguiente paso.

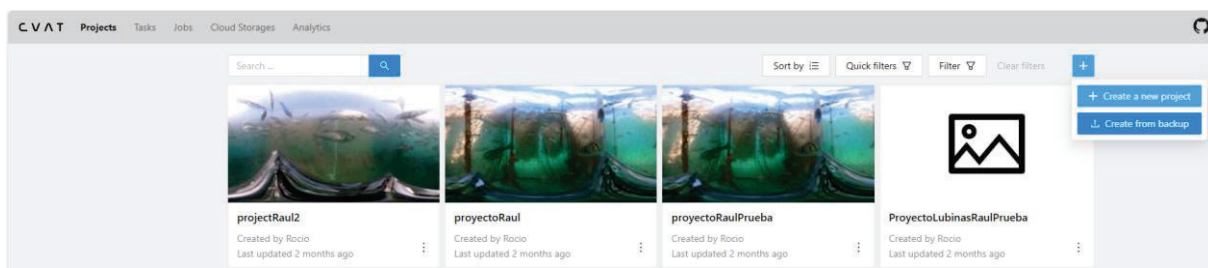


Figura 72 Interfaz principal de CVAT

Figura 73 Campos para creación de un proyecto en CVAT

2. Crear una nueva tarea.

Pulsamos en la parte superior a la parte de “Projects”, accedemos al proyecto que hemos creado y una vez dentro, se hace click en el desplegable “+” y selecciona la opción “Create a new task”.

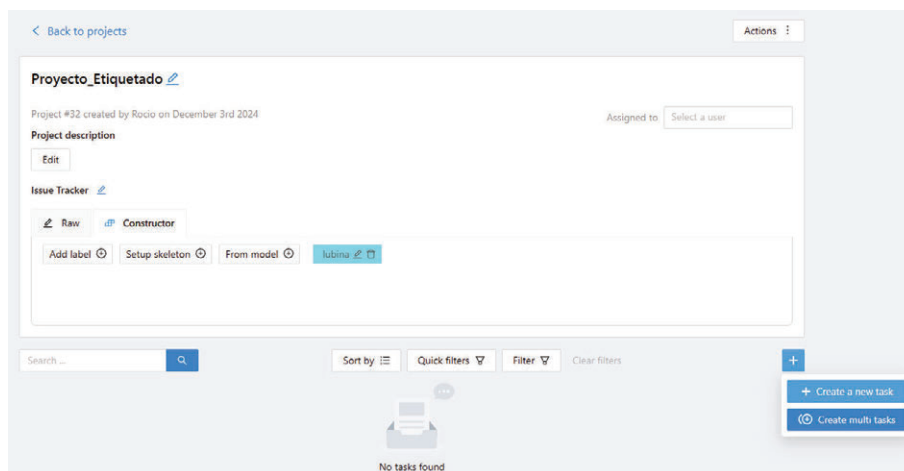


Figura 74 Menú de un proyecto en CVAT

Se definen los campos de tarea como el nombre. Además, se deben subir las imágenes que hemos obtenido previamente para realizar el posterior etiquetado, para ello simplemente seleccionamos las imágenes y las arrastramos hasta donde pone: “Click or drag files to this

area”. En este caso, no se debe indicar el tipo de etiqueta, ya que se indica que va a utilizar las etiquetas del propio proyecto, que previamente se había indicado como “lubina”.

Figura 75 Menú de una tarea (task) en CVAT

En la parte superior del CVAT y a la derecha de la opción “Projects”, encontramos la opción “Tasks”, en la que se definen las tareas y en la nueva tarea que hemos creado se va a pulsar en “Open”.



Figura 76 Menú de las tareas en CVAT

Una vez dentro de la tarea, para acceder al portal de etiquetado de imágenes se debe pulsar “Job #Valor” en el apartado de “Jobs”.

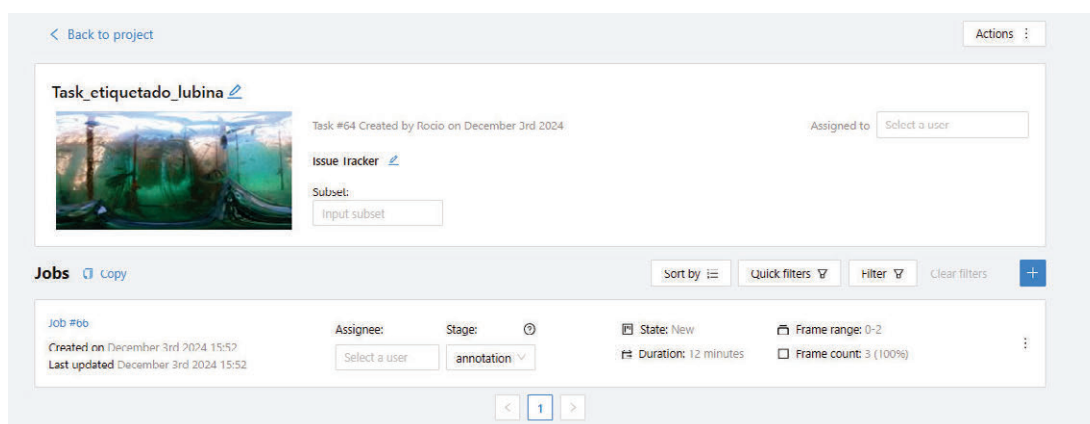


Figura 77 Menú de la tarea creada en CVAT

Al pulsar al trabajo, se abrirá automáticamente, la ventana principal en la cual se realiza el etiquetado de imágenes.

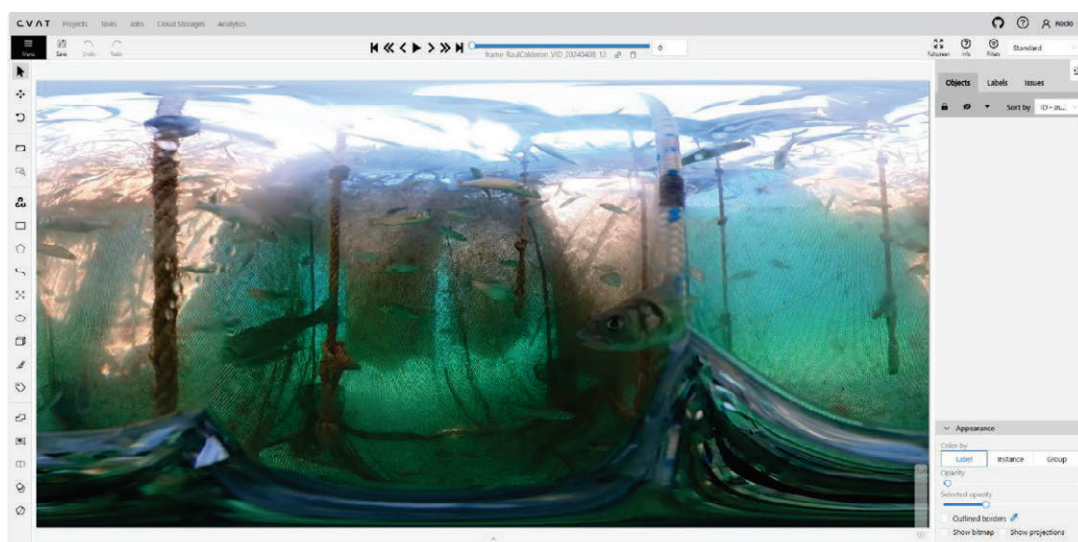


Figura 78 Menú de etiquetado de imágenes en CVAT

Para etiquetar imágenes seleccionamos en el panel de la izquierda la opción de dibujar un rectángulo y debe estar seleccionada la etiqueta que hemos definido en el proyecto y en el caso de este proyecto se ha utilizado la función "Shape" y como "Drawing method" se selecciona "By 2 points" y se dibuja el contorno de las lubinas de cada una de las imágenes.

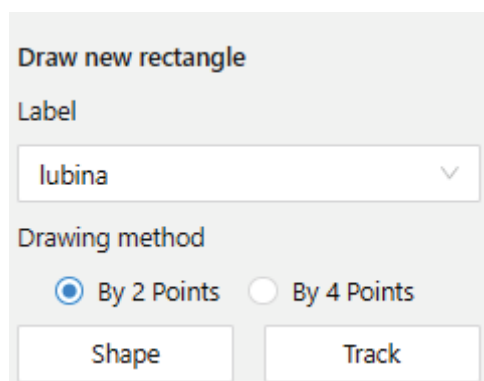


Figura 79 Desplegable opción "Draw a new rectangle" en CVAT

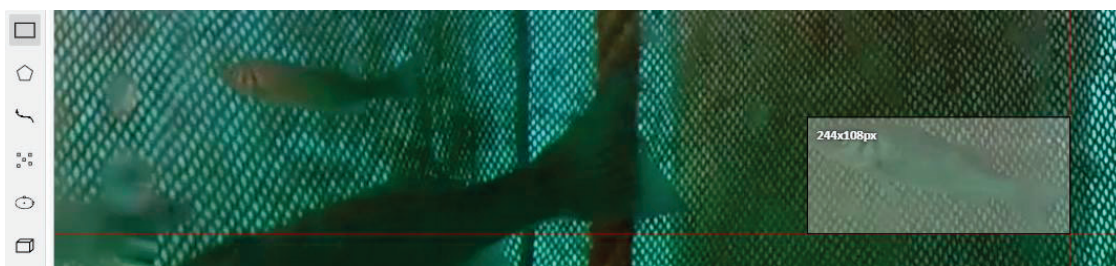


Figura 80 Etiquetado de una lubina con la opción de dibujar un rectángulo

Una vez que una imagen está completamente etiquetada, pasamos a la siguiente mediante el siguiente cuadro de opciones, en el cual las opciones que más interesan son “>” para avanzar hacia la siguiente imagen o “<” para volver a una imagen previa. En este punto se recomienda tener todas las imágenes etiquetadas antes de pasar al exportado final.

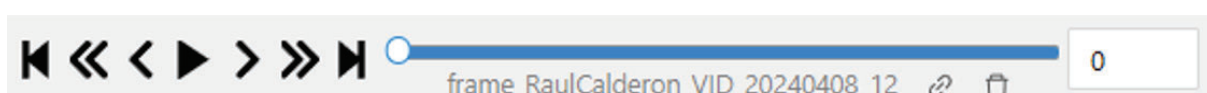


Figura 81 Cuadro de opciones para navegar entre imágenes en CVAT

Una vez que todas las imágenes están etiquetadas, procedemos a exportar las imágenes en formato YOLO; para ello, volvemos de nuevo a pulsar a la opción “Tasks” en la parte superior de la pantalla y pulsamos en “Actions” y dentro del menú desplegable pulsamos “Export task dataset”.

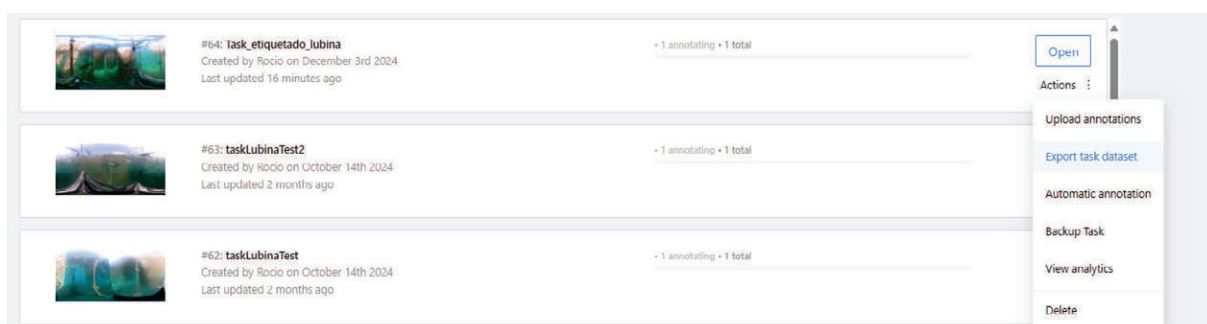


Figura 82 Exportado de imágenes de una tarea en CVAT

Nos aparecerá por pantalla un panel en el que nos pedirá especificar el formato de los datos de salida y seleccionamos YOLO 1.1 y le damos nombre al fichero .zip final.

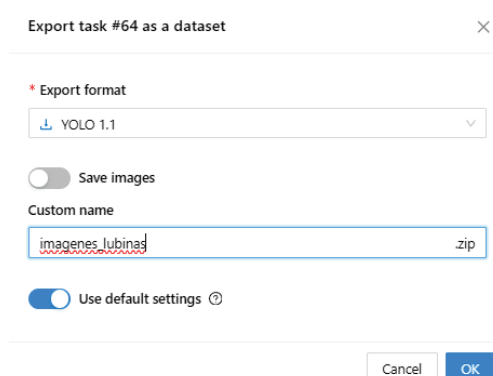


Figura 83 Selección del formato de salida de las etiquetas con CVAT

A.5 Conjuntos de datos para el entrenamiento

Una vez que hemos descargado las etiquetas tendremos un fichero, en el cual tenemos los siguientes elementos:

obj_train_data	Carpeta de archivos			
obj.data	Archivo DATA	1 KB	No	1 KB
obj.names	Archivo NAMES	1 KB	No	1 KB
train	Documento de texto	1 KB	No	1 KB

Figura 84 Archivos en el fichero descargado de etiquetas de CVAT

Lo más importante es el contenido de la carpeta “obj_train_data”, la cual contiene todos los archivos de texto, que cada uno de ellos cuenta con las etiquetas de cada foto por individual.

VID_20230211_143502_00_004_2024...	Documento de texto	4 KB	No	4 KB
VID_20230211_143502_00_004_2024...	Documento de texto	2 KB	No	2 KB
VID_20230211_143502_00_004_2024...	Documento de texto	2 KB	No	2 KB
VID_20230211_143502_00_004_2024...	Documento de texto	3 KB	No	3 KB
VID_20230211_143502_00_004_2024...	Documento de texto	3 KB	No	3 KB
VID_20230211_143502_00_004_2024...	Documento de texto	3 KB	No	3 KB
VID_20230211_143502_00_004_2024...	Documento de texto	2 KB	No	2 KB
VID_C5_D1_4J23_01_2024-04-16_18-...	Documento de texto	4 KB	No	4 KB
VID_C6_D1_4J23_02_2024-04-16_18-...	Documento de texto	3 KB	No	3 KB
VID_C6_D1_4J23_02_2024-04-16_18-...	Documento de texto	3 KB	No	3 KB

Figura 85 Contenido de la carpeta obj_train_data

Una vez que tenemos tanto etiquetas como las imágenes originales se deben distribuir los datos entre los diferentes conjuntos. En este proyecto, la distribución entre el conjunto de entrenamiento y de validación se hace de la siguiente forma:

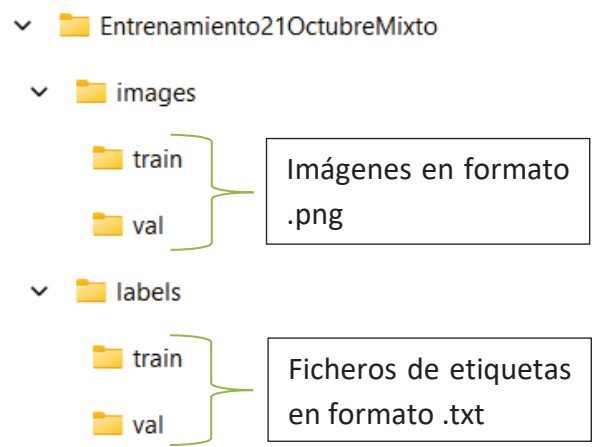


Figura 86 Sistemas de ficheros para el entrenamiento

Antes de comenzar a entrenar, se debe configurar un fichero dataset.yaml que va a ser utilizado por el programa de entrenamientos para tener claros dónde se ubican ambos

conjuntos. En este caso, donde pone “train”, se debe indicar la ubicación de images/train, ofreciendo el conjunto de imágenes del conjunto de entrenamiento y en “val” se debe indicar la ubicación de images/val, apuntando al conjunto de imágenes del conjunto de validación. En este caso, no es necesario indicar explícitamente las ubicaciones de las etiquetas, ya que, si nuestro sistema de archivos sigue la estructura de la Figura 79, será capaz de asociarlas correctamente. Se muestra un ejemplo del fichero dataset.yaml utilizado para este caso:

```
! dataset.yaml X
Entrenamiento21Octubre2024Mixto > ! dataset.yaml
1 train: D:\TRABAJO FINAL DE GRADO (YOLOv8-lubinas)\tfg\Entrenamiento21Octubre2024Mixto\images\train
2 val: D:\TRABAJO FINAL DE GRADO (YOLOv8-lubinas)\tfg\Entrenamiento21Octubre2024Mixto\images\val
3 nc: 2
4 names: ["lubina", "Lubina"]
```

Figura 87 Fichero dataset.yaml

A.6 Entrenamiento y ficheros de métricas

En este caso, se van a explicar para qué se utilizaron cada uno de los programas relacionados con el entrenamiento y obtención de métricas. En primer lugar, para realizar el entrenamiento se hizo uso del código train.py. Una vez que se obtuvo el mejor modelo, se pasó a analizar la carpeta de resultados que genera YOLO, que es la siguiente:

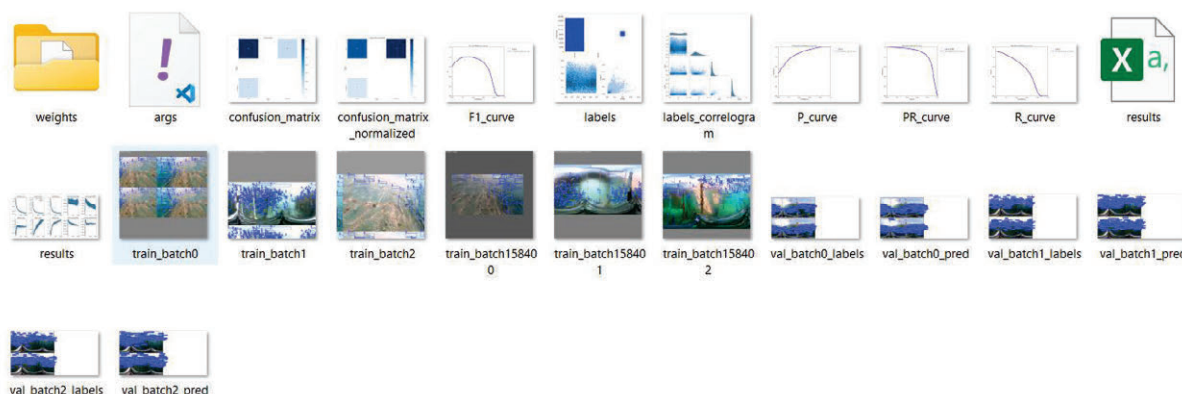


Figura 88 Resultados de entrenamiento generados por YOLOv8

Una vez que tenemos estos resultados, se pasa a calcular el IoU, para el cual se tiene la aplicación prediccionesIoUacum.py, la cual calcula tanto el IoU individual de cada una de las imágenes como el global uniendo los resultados de todas para obtener un valor global, que permita ver de una mejor manera cómo se realizan las detecciones en conjunto. Este programa requiere de la ruta al modelo de YOLOv8, el path de las imágenes de test y el de las etiquetas de test. Una vez terminada la ejecución de dicho programa, genera una carpeta en la que almacena el histograma de todas las detecciones, una nueva carpeta en la que genera el histograma por individual y cada una de las imágenes con las figuras de las cajas delimitadoras que predice el modelo frente a las etiquetadas inicialmente.

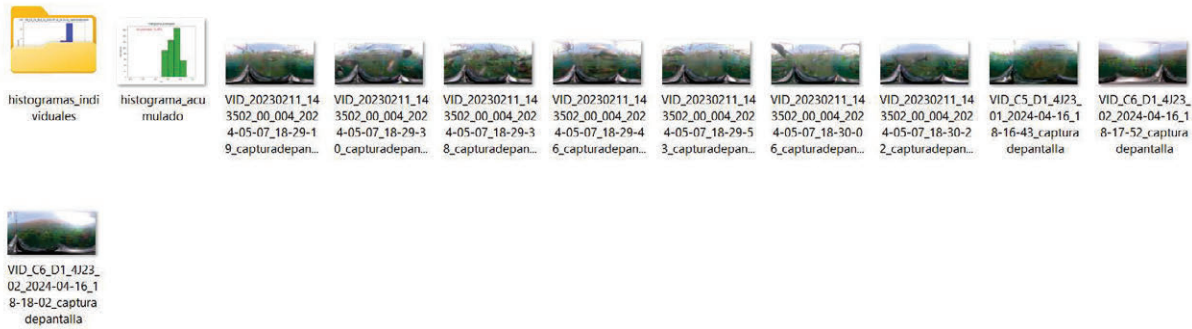


Figura 89 Resultados obtenidos por el programa de IoU

A.7 Funcionamiento de la aplicación

En este caso, se cuenta con dos versiones: la aplicación que reproduce el vídeo en vista panorámica se llama `app_360.py` y la que lo hace en vista panorámica se llama `app_panoramica.py`. Para ambos casos, una vez que termina el proceso de ejecución se almacenan un vídeo que sirve para comprobar las detecciones que realiza el modelo de manera visual y también se almacena un fichero de texto en el que se relacionan las detecciones por cada fotograma. Tanto en la aplicación que reproduce el vídeo en formato 360° como en la que lo hace con la vista panorámica, se pretende reproducir el vídeo por completo salvo que decidamos parar el proceso, para lo cual se pulsa la tecla “q” minúscula.

Los requisitos de ejecución del programa es pasarle tanto el modelo YOLOv8 entrenado (`best.pt`) y la ubicación del vídeo que ya esté transformado según se indicaba en el anexo “A.3 Exportado del vídeo 360°” de este documento.