PROYECTO FIN DE GRADO

TÍTULO: Sistema automatizado de segmentación de truchas en vídeo para el estudio de su comportamiento

AUTORA: Ruth Báez Perandones

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTORA: Juana María Gutiérrez Arriola

DEPARTAMENTO: Departamento de Ingeniería Telemática y Electrónica (DTE)

VºBº TUTORA

Miembros del Tribunal Calificador:

PRESIDENTE: Francisco Cano Broncano

TUTORA: Juana María Gutiérrez Arriola

SECRETARIO: Nicolás Sáenz Lechón

Fecha de lectura: 18 de Julio de 2024

Calificación:

El Secretario,

Agradecimientos

Quiero agradecer a mis padres y mi hermano por su amor incondicional y estar presentes en todo el proceso.

A Rafa por estar siempre a mi lado. Tu apoyo constante, paciencia infinita y tu capacidad de hacerme reír en los peores momentos.

A Rijana por acompañarme en cada palabra escrita.

A Puchun por todos los momentos vividos, las épocas de estrés y todas las risas compartidas.

Aroia, Jakey, Pablo y Maca sin vosotros todo este camino habría sido imposible.

A Samuel por acompañarme en la recta final.

A mi tutora por su guía y apoyo.

Resumen

El objetivo de este proyecto, titulado "Sistema automatizado de segmentación de truchas en vídeo para el estudio de su comportamiento", es desarrollar un sistema automático de segmentación y seguimiento de truchas en vídeos, para facilitar estudios sobre el comportamiento de estos peces en acuicultura. La segmentación de imágenes es una herramienta central en la visión artificial, especialmente útil para solventar problemas donde las regiones a segmentar tienen características muy similares. A través del uso de inteligencia artificial, específicamente el aprendizaje profundo y las redes neuronales convolucionales (CNN), se ha demostrado una gran eficacia en la segmentación de imágenes debido a su capacidad para aprender y reconocer patrones.

El principal objetivo de este proyecto es la implementación de un sistema capaz de localizar una trucha en un entorno controlado y realizar un seguimiento de sus movimientos, identificando exactamente la sección en la que se encuentra en cada momento. Para lograr esto, se han establecido varias metas: desarrollar un sistema que permita segmentar el pez completo y localizar partes específicas como la cabeza.

Desde una perspectiva tecnológica, el sistema ha sido codificado en Python y utiliza una red neuronal para agilizar el proceso de segmentación y seguimiento. La entrada al sistema consiste en vídeos que muestran una trucha en un entorno controlado. Las condiciones finales esperadas son que el sistema sea capaz de detectar la presencia de la trucha, rastrear sus movimientos y generar un vídeo de salida que resalte la trucha mediante un recuadro y una máscara que destaca su figura.

Los resultados obtenidos evidencian que el método basado en el uso del centroide con un tamaño de imagen de 640 ofrece la mayor precisión, con una eficacia del 90,38%. Sin embargo, para aplicaciones que requieren una mayor rapidez de procesamiento, el uso de un tamaño de imagen de 320 también es viable, sacrificando solo un 2,09% de eficacia. La flexibilidad del sistema permite ajustar este valor según las necesidades específicas.

En conclusión, el sistema desarrollado es una herramienta eficaz para el seguimiento de truchas, proporcionando una base sólida para estudios detallados sobre su comportamiento. A pesar de los desafíos tecnológicos y de precisión, los resultados obtenidos son prometedores y demuestran la viabilidad de utilizar técnicas de inteligencia artificial en el ámbito de la acuicultura.

Abstract

The aim of this project, titled "Automated trout segmentation system in video for studying their behaviour," is to develop an automatic system for segmenting and tracking trout in videos, with the goal of facilitating studies on the behaviour of these fish in aquaculture. Image segmentation is a central tool in computer vision, particularly useful for solving problems where the regions to be segmented have very similar characteristics. Using artificial intelligence, specifically deep learning and convolutional neural networks (CNN), significant effectiveness in image segmentation has been demonstrated due to their ability to learn and recognize patterns.

The main goal of this project is to implement a system capable of locating a trout in a controlled environment and tracking its movements, precisely identifying the section it is in at any given moment. To achieve this, several goals have been established: developing a system that can segment the entire fish and locate specific parts such as the head.

From a technological perspective, the system has been coded in Python and uses a neural network to speed up the segmentation and tracking process. The system's input consists of videos showing a trout in a controlled environment. The expected final conditions are that the system can detect the presence of the trout, track its movements, and generate an output video that highlights the trout with a bounding box and a mask that emphasize its figure.

The results show that the method based on using the centroid with an image size of 640 offers the highest accuracy, with a success rate of 90.38%. However, for applications requiring faster processing, using an image size of 320 is also viable, sacrificing only 2.09% accuracy. The system's flexibility allows this value to be adjusted according to specific needs.

In conclusion, the developed system is an effective tool for tracking trout, providing a solid foundation for detailed studies on their behaviour. Despite the technological and accuracy challenges, the results obtained are promising and demonstrate the feasibility of using artificial intelligence techniques in the field of aquaculture.

Índice de figuras

Figura 1. Estructura de la inteligencia artificial 7
Figura 2. Estructura de las capas y los nodos de una red neuronal[12] 9
Figura 3. Funciones de activación 10
Figura 4. Esquema del proceso de un nodo[13] 11
Figura 5. Función de pérdida[14] 12
Figura 6. Proceso de la capa convolucional[16] 13
Figura 7. Proceso de la capa de agrupación[16] 14
Figura 8. Proceso de las R-CNN[17] 15
Figura 9. Proceso Faster R-CNN[17] 16
Figura 10. Método aplicado por YOLO[17] 17
Figura 11. Ejemplo de cajas delimitadoras y máscaras[19] 18
Figura 12. Fotograma de un vídeo 22
Figura 13. Sistema de etiquetado LabelMe 23
Figura 14. Imágenes de prueba del modelo entrenado 25
Figura 15. Numeración de las secciones 26
Figura 16. Ejemplo de seguimiento con centroide 27
Figura 17. Detección de la cabeza por parte del modelo 28
Figura 18. Detección de la cabeza basándose en el movimiento 29
Figura 19. Diagrama del sistema 30
Figura 20. Matriz de confusión 34
Figura 21. Gráficas obtenidas del entrenamiento del modelo 35
Figura 22. Comparativa del clip 87 con centroide 37
Figura 23. Comparativa del clip 78 con centroide 38
Figura 24. Comparativa del clip 71 con centroide 39
Figura 25. Comparativa del clip 87 con la cabeza como referencia 40
Figura 26. Comparativa del clip 65 con la cabeza como referencia 41
Figura 27. Comparativa del clip 85 con imgsz=320 42
Figura 28. Comparativa gráfica 43
Tabla 1. Eficacia promedio de las propuestas 43
Figura 29. Comparación clip 67 44
Figura 30. Comparación imgsz=640 e imgsz=320 45
Tabla 2. Presupuesto general 47
Tabla 3. Porcentajes de coincidencia para cada clip 57
Figura 31. Path del modelo de YOLO 59
Figura 32. Entorno de ejecución 59
Figura 33. Interfaz del programa 60
Figura 34. Selección de las rutas 60
Figura 35. Confirmación de los datos seleccionados 61
Figura 36. *Frames* en ejecución 61
Figura 37. Salida del sistema 62

Figura 38. Errores de selección 62

Lista de acrónimos

AA	Aprendizaje automático
AP	Aprendizaje Profundo
cls	Pérdidas de clasificación
CNN	Redes neuronales convoluciones
DL	*Deep Learning*
dls	Pérdidas de regresión
FN	False negative
FP	False positive
IA	Inteligencia Artificial
Imgsz	Tamaño de la imagen
Iou	Intersección sobre unión
mAP	Precisión media
ML	*Machine Learning*
ODS	Objetivos de Desarrollo Sostenible
OpenCV	Open Source Computer Vision Library
R-CNN	Region-based Convolutional Neural Networks
RPN	Red de Propuestas de Regiones
ReLU	Rectified Linear Unit
RNA	Redes Neuronales Artificiales
SVM	Suport Vector Machine
TN	True negative
TP	True positive
YOLO	You Only Look Once

Índice de contenidos

Resumen iii

Abstract v

Índice de figuras vii

Lista de acrónimos ix

1. Introducción 1

1.1 Marco y motivación del proyecto 1

1.2 Objetivos técnicos y académicos 1

1.3 Estructura del resto de la memoria 2

2. Marco tecnológico 5

2.1 Sistemas de detección en acuicultura [2] 5

2.1.1 Sensores y dispositivos 5

2.1.2 Tecnologías especificas 5

2.2 Inteligencia artificial 6

2.2.1 Aprendizaje automático 7

2.2.2 Aprendizaje profundo 8

2.3 Redes neuronales 8

2.3.1 Función de perdida y descenso de gradiente[14] 11

2.3.2 Redes neuronales convolucionales 12

2.4 Detección de objetos 15

2.4.1 R-CNN (Region-based Convolutional Neural Networks) 15

2.4.2 Fast R-CNN 15

2.4.3 Faster R-CNN 16

2.4.4 YOLO (You Only Look Once) 16

2.4.5 Cajas delimitadoras y máscaras 17

3. Especificaciones y restricciones de diseño 19

3.1 Especificaciones de diseño 19

3.2 Restricciones de diseño 19

4. Descripción de la solución propuesta 21

4.1 Descripción de los datos 21

4.2 Etiquetado de las imágenes 22

4.3 Entrenamiento del sistema 23

4.4 Seguimiento de la trucha 25

4.4.1 Seguimiento con centroide 26

4.4.2 Seguimiento con la cabeza como referencia 27

4.5 Parámetros de detección [30] 29

4.6 Arquitectura del sistema 30

5. Resultados 33

5.1 Resultados del entrenamiento 33

5.1.1 Matriz de confusión 33

5.1.2 Rendimiento del modelo 35

5.2 Resultados obtenidos 36
5.2.1 Seguimiento con centroide 36
5.2.2 Seguimiento con la cabeza como referencia 39
5.2.3 Cambio de parámetros en la detección 41
5.3 Comparativa de resultados 42
6. Presupuesto 47
7. Impacto del proyecto[36] 49
7.1 Implicaciones del proyecto 49
7.2 Contribución a los Objetivos de Desarrollo Sostenible (ODS)[37] 49
8. Conclusiones 51
8.1 Trabajos futuros 51
9. Referencias 53
Anexo 57
A.1 Resultados obtenidos 57
Manual de usuario 59
A.2 Funcionamiento del sistema 59
A.3 Posibles errores 62

1. Introducción

1.1 Marco y motivación del proyecto

Este proyecto se enmarca en la visión artificial y la inteligencia artificial, con un enfoque específico en la segmentación y el seguimiento de objetos en imágenes y vídeos. La segmentación de imágenes es una herramienta fundamental en la visión artificial que permite identificar y separar diferentes objetos o regiones dentro de una imagen. Esta capacidad es particularmente importante en entornos donde los objetos a segmentar tienen características visuales muy similares, lo que dificulta su identificación precisa.

La necesidad de este proyecto surge en el ámbito de la acuicultura, donde es crucial monitorear el comportamiento de las truchas para promover su bienestar y optimizar su productividad. Actualmente, la segmentación y el seguimiento manual de truchas en vídeos es un proceso laborioso. Mediante la implementación de un sistema automatizado que utilice técnicas avanzadas de aprendizaje profundo y redes neuronales convolucionales (CNN), se busca desarrollar una solución eficiente y precisa para este problema.

El objetivo general del proyecto es desarrollar un sistema capaz de localizar una trucha en un entorno controlado y realizar un seguimiento de sus movimientos, identificando exactamente la sección en la que se encuentra en cada momento. Las soluciones propuestas incluyen el uso de redes neuronales para segmentar el pez completo y localizar partes específicas como la cabeza. Además, se han planteado dos enfoques para identificar la sección en la que se encuentra la trucha: uno utilizando el centroide de la detección y otro utilizando la posición de la cabeza, facilitando así el análisis detallado del comportamiento de las truchas.

1.2 Objetivos técnicos y académicos

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

- Desarrollar un sistema que permita segmentar el pez completo en vídeos tomados en un entorno controlado.
- Implementar un algoritmo de seguimiento que pueda rastrear los movimientos de la trucha a lo largo del vídeo.
- Evaluar y comparar diferentes métodos y parámetros de segmentación para optimizar la precisión y eficiencia del sistema.

Desde el punto de vista académico, el proyectista adquiere las siguientes competencias y habilidades,[1]:

- Capacidad de búsqueda y selección de información, de razonamiento crítico y de elaboración y defensa de argumentos dentro del área.
- Capacidad para expresarse correctamente de forma oral y escrita y transmitir información mediante documentos y exposiciones en público.
- Capacidad de abstracción, de análisis y de síntesis de resolución de problemas

- Capacidad de analizar y valorar el impacto social y medioambiental de las soluciones técnicas.
- Habilidades para la utilización de las Tecnologías de la Información y las Comunicaciones.
- Habilidades de aprendizaje con un alto grado de autonomía.
- Capacidad de realizar programación en tiempo real, concurrente, distribuida y basada en eventos, así como el diseño de interfaces persona-computador.

1.3 Estructura del resto de la memoria

En los próximos capítulos de la memoria se ofrece información sobre los distintos aspectos del proyecto. En primer lugar, se habla sobre el **marco tecnológico**, el cual presenta una revisión de las tecnologías existentes relacionadas con la segmentación de imágenes y el seguimiento de objetos en vídeos, con un enfoque específico en el uso de redes neuronales convolucionales (CNN). Este punto proporciona el contexto tecnológico necesario para comprender las bases teóricas y prácticas del proyecto.

Se continua con las **especificaciones y restricciones de diseño**, donde se describe las especificaciones de diseño y las restricciones que deben tenerse en cuenta para el desarrollo del sistema. Aquí se detallan los requisitos técnicos y las limitaciones del proyecto, estableciendo el marco dentro del cual se desarrollará la solución.

Seguidamente se hace una **descripción de la solución propuesta**, en la cual se detalla el proceso de etiquetado de las imágenes, el entrenamiento del sistema y las técnicas de seguimiento de la trucha. Este capítulo incluye una explicación de los métodos utilizados para el seguimiento, destacando el uso del centroide y la cabeza como referencias para la localización de la trucha en el entorno controlado.

Posteriormente, se analizan los **resultados** obtenidos con el sistema desarrollado, comparando los diferentes métodos y parámetros utilizados. Se investiga como influyen estos resultados y su relevancia para el estudio del comportamiento de las truchas, proporcionando una evaluación de la efectividad y precisión del sistema.

Después, se ofrece un análisis del **presupuesto**, indicando los gastos asociados con el desarrollo y la implementación del sistema. Este capítulo incluye una estimación financiera completa, abarcando desde el costo de los recursos tecnológicos hasta el tiempo invertido en el desarrollo del proyecto.

Más adelante, se analiza el **impacto del proyecto** en términos de sus implicaciones para la acuicultura y su contribución a los Objetivos de Desarrollo Sostenible (ODS). Este capítulo evalúa cómo el proyecto puede influir positivamente en prácticas sostenibles y en la mejora del bienestar de las truchas.

Finalmente, se analizan las **conclusiones** y los posibles **trabajos futuros** ofreciendo una visión general del trabajo realizado, resumiendo los problemas abordados, las soluciones propuestas y los resultados obtenidos. Además, se proponen futuras líneas de trabajo para mejorar y

expandir el sistema desarrollado, sugiriendo posibles mejoras y nuevos enfoques para abordar los desafíos restantes en la segmentación y seguimiento de truchas.

2. Marco tecnológico

2.1 Sistemas de detección en acuicultura [2]

En el ámbito de la acuicultura, el bienestar animal es un aspecto crucial que impulsa continuos avances en el estudio del comportamiento animal. Uno de los métodos comúnmente empleados es el experimento *open field*, el cual proporciona datos significativos sobre la respuesta de los peces en entornos con los que no están familiarizados. Automatizar este tipo de experimentos ofrece numerosos beneficios para los investigadores, permitiendo una recolección de datos más precisa y eficiente. La implementación de sistemas automatizados no solo optimiza los recursos, sino que también facilita la monitorización continua y detallada de las interacciones y comportamientos, contribuyendo así al avance del conocimiento en el campo del bienestar animal en acuicultura.

El desarrollo tecnológico en la detección de objetos, especialmente en aplicaciones de visión artificial para la acuicultura, ha avanzado significativamente en los últimos años. A continuación, se muestran algunas de las tecnologías más relevantes utilizadas en la detección y seguimiento de peces en entornos acuáticos.

2.1.1 Sensores y dispositivos

La detección de objetos, incluidos los peces, se ha beneficiado enormemente del uso de diversos sensores y dispositivos avanzados:

- **Cámaras**: Las cámaras han evolucionado considerablemente en términos de resolución y tamaño, lo que las hace ideales para capturar imágenes detalladas tanto en 2D como en 3D. La combinación de cámaras con otros sensores, como láseres 3D o sensores acústicos, proporciona sistemas más precisos y versátiles.
- **Sensores 3D**: Estos dispositivos generan una nube de puntos que representa la superficie de los objetos en la escena. Sensores activos como *Time of Flight*, luz estructurada y de triangulación son utilizados para mapear objetos en 3D con alta precisión y velocidad.
- **Tecnología acústica**: El uso de sónar y DIDSON ha revolucionado la detección de peces a largas distancias bajo el agua. Aunque excelente para detectar bancos de peces o grandes objetos, tiene limitaciones en la detección precisa de objetos pequeños o individuales.
- **Láser de infrarrojos**: Ideal para condiciones de poca luz bajo el agua, el láser de infrarrojos emite haces que, al ser interrumpidos por objetos, permiten calcular la distancia y detectar la presencia de peces con precisión.

2.1.2 Tecnologías especificas

Para desarrollar aplicaciones avanzadas de **visión artificial** en entornos submarinos, se utilizan herramientas como OpenCV y Halcon, que ofrecen algoritmos sofisticados para el procesamiento de imágenes y la detección de objetos. Estas bibliotecas son fundamentales

debido a su capacidad para trabajar con imágenes de baja visibilidad y condiciones ambientales desafiantes bajo el agua.

OpenCV (Open Source Computer Vision Library)[3] es una biblioteca de código abierto que proporciona una extensa colección de algoritmos para el procesamiento de imágenes y visión artificial. Es compatible con múltiples lenguajes de programación y plataformas, lo que facilita su integración en diversos proyectos de visión artificial.

Halcon,[4] por otro lado, es una biblioteca comercial de software desarrollada por MVTec Software GmbH, especializada en aplicaciones de visión artificial. Halcon ofrece un conjunto de herramientas y algoritmos avanzados para la detección de objetos y el procesamiento de imágenes en entornos submarinos. Su capacidad para abordar condiciones ambientales difíciles y su rendimiento optimizado la convierten en una opción preferida para aplicaciones industriales y de investigación que requieren soluciones de visión artificial de alta precisión y fiabilidad.

Además, el uso de **redes neuronales convolucionales** (CNN) en el contexto de la visión artificial ha mejorado significativamente la detección y clasificación de peces. Las CNN permiten analizar imágenes capturadas por cámaras bajo el agua, identificando peces con alta precisión y velocidad. Esta tecnología combina el poder de la inteligencia artificial con algoritmos diseñados específicamente para reconocer patrones complejos en datos visuales, lo que es crucial para aplicaciones como la monitorización de especies marinas y la gestión de ecosistemas acuáticos.

2.2 Inteligencia artificial

La inteligencia artificial (IA)[5] es una disciplina de la ciencia y la ingeniería cuyo objetivo es crear tecnologías capaces de ejecutar tareas cognitivas como el reconocimiento de voz e imágenes, la toma de decisiones y la resolución de problemas, de manera similar a la de un ser humano.

El concepto de IA se introdujo por primera vez en 1950 por Alan Turing, quien planteó si las máquinas podían pensar. Entre 1957 y 1974, el avance tecnológico permitió el desarrollo de algoritmos de aprendizaje automático (*machine learning*, ML). Durante la década de 1980, se publicaron numerosos artículos sobre aprendizaje profundo (*deep learning,* DL) demostrando que las maquinas eran capaces de aprender de la experiencia. Hoy en día, la IA se ha vuelto cada vez más accesible y es capaz de realizar tareas cognitivas complejas que antes solo eran posibles para la inteligencia humana.

En la Figura 1 se presenta una representación de la estructura de la inteligencia artificial. Dentro de este amplio campo, el aprendizaje automático se centra en sistemas que adquieren la capacidad de aprender a partir de datos. A su vez, dentro del aprendizaje automático, el aprendizaje profundo es una técnica específica que emplea redes neuronales para procesar los datos de manera más eficiente.

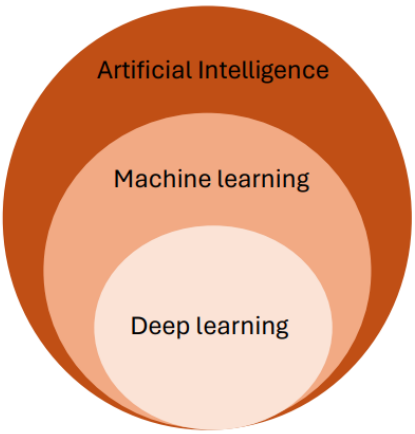


Figura 1. Estructura de la inteligencia artificial

2.2.1 Aprendizaje automático

El aprendizaje automático (AA)[6] conocido en inglés como *machine learning,* es un área de la IA que se enfoca en desarrollar sistemas capaces de aprender y abordar diversas cuestiones a partir de datos, sin estar específicamente programados para resolver cada problema en particular. Estos modelos son capaces de identificar patrones en los datos que reciben y tomar decisiones basadas en dichos patrones.[7]

A continuación, se presentan algunos de los métodos más reconocidos del AA, siendo el aprendizaje supervisado y el no supervisado los más frecuentes[8]:

- **Aprendizaje supervisado**: para el entrenamiento de algoritmos de aprendizaje supervisado se utilizan datos etiquetados, donde se conoce el resultado correcto. Al proporcionar estas respuestas correctas, el modelo aprende patrones que le permiten predecir la respuesta idónea cuando se le presentan datos no etiquetados. Estos sistemas se emplean para resolver problemas de regresión y clasificación. Por ejemplo, se puede entrenar un modelo para reconocimiento facial utilizando imágenes etiquetadas con expresiones faciales y emociones. A medida que la red procesa las respuestas correctas, aumenta gradualmente su conocimiento. Una vez completado el entrenamiento, el modelo es capaz de identificar las emociones en un rostro presente en una imagen que nunca ha visto. Una de sus limitaciones es que puede ser necesario la etiquetación de millones de datos para un correcto funcionamiento.
- **Aprendizaje no supervisado**: el sistema se entrena con datos no etiquetados previamente, por lo que el sistema no conoce la respuesta correcta de antemano. El modelo debe ser capaz de analizar todos los datos y encontrar la estructura común entre ellos. Estos sistemas son muy útiles para detectar anomalías, reconocer patrones y agrupar datos en categorías. Por ejemplo, pueden identificar atributos comunes en distintos clientes, lo que resulta útil para campañas de marketing posteriores. Una de sus limitaciones es que las predicciones que ofrece pueden ser imprecisas.
- **Aprendizaje semi-supervisado**: este método consiste en una combinación entre el aprendizaje supervisado y el aprendizaje no supervisado. Las aplicaciones son las mismas que las del aprendizaje supervisado. Para entrenar estos sistemas, se necesita

datos etiquetados y una mayor cantidad de datos sin etiquetar. El proceso comienza con un pequeño entrenamiento del sistema utilizando los datos etiquetados. Luego, se lleva a cabo el proceso denominado pseudoetiquetado, donde el propio algoritmo etiqueta los datos no etiquetados basándose en lo que ya ha aprendido. Por último, el sistema se vuelve a entrenar con todos los datos resultantes de los procesos anteriores. La principal ventaja de este método es que no requiere tanto tiempo para etiquetar todos los datos.

- **Aprendizaje por refuerzo**: este método se basa en un sistema de aprendizaje de prueba y error, donde algunas acciones resultan en una mayor recompensa. El aprendizaje incluye tres elementos clave: el agente, encargado de aprender y decidir; el entorno, con el cual el agente interactúa; y las acciones disponibles para el agente. El propósito es obtener la mayor recompensa posible, siendo estas recompensas definidas por el científico de datos, quien influye en los resultados. La aplicación de estos sistemas se enfoca principalmente en la robótica, los juegos y la navegación.

2.2.2 Aprendizaje profundo

El aprendizaje profundo (AP)[9] conocido en inglés *deep learning,* es un tipo de AA que busca simular el aprendizaje humano mediante el uso de redes neuronales artificiales (RNA) y con un enfoque de aprendizaje por capas. La primera capa aprende las características básicas, mientras que las capas siguientes analizan características progresivas más complejas, hasta que todas las capas del sistema están entrenadas.

Esto resulta en un sistema mucho más complejo, capaz de detectar patrones que podrían pasar inadvertidos con otras técnicas de AA, permitiendo a los modelos obtener mejores resultados en diversas tareas.

Los principales usos del AP incluyen la visión por computadora, el reconocimiento de voz, el procesamiento del lenguaje natural y sistemas de recomendación.

2.3 Redes neuronales

Las redes neuronales,[10] también conocidas como redes neuronales artificiales (RNA, por sus siglas en inglés ANN-Artificial Neural Network), buscan imitar el funcionamiento de las neuronas biológicas del cerebro humano a través de un conjunto de algoritmos.[11] Estas redes están formadas por múltiples capas, como se ilustra en la Figura 2: una capa de entrada, una o más capas ocultas y una capa de salida. Cada una de estas capas contiene nodos, o también conocidos como neuronas. Cada neurona cuenta con múltiples entradas, cada una con su respectivo peso, un umbral (también llamado bias o sesgo) y una salida. Una red neuronal se considera de aprendizaje profundo cuando tiene más de tres capas, es decir tiene más de una capa oculta.[12]

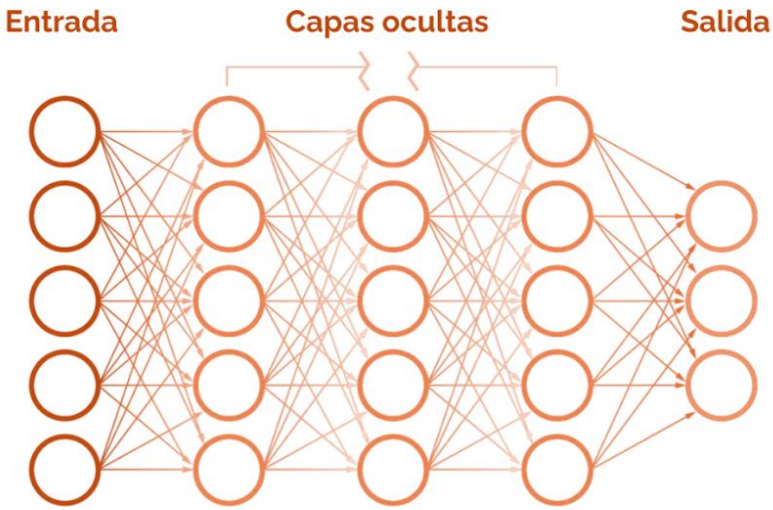


Figura 2. Estructura de las capas y los nodos de una red neuronal[12]

Para comprender el funcionamiento de las redes neuronales, es esencial analizar su proceso. En primer lugar, los datos de entrada llegan a los nodos de la capa de entrada, donde se procesan. Una vez finalizado el procesamiento, se asignan los diferentes pesos que determinan la importancia de cada variable, por lo que la variable con mayor relevancia obtendrá un valor mayor. Posteriormente, siguiendo la ecuación (1), cada nodo multiplica sus entradas por sus respectivos pesos y luego se suman.

$$z = \left(\sum_{i=1}^{n} w_i \cdot x_i \right) + b \tag{1}$$

Siendo w_i los pesos correspondientes a cada entrada, x_i los datos de entrada y b el umbral establecido.

Luego la salida se procesa a través de la función de activación (2), la cual normaliza el valor de la salida. Si este valor excede un umbral especifico, el nodo se activa y transmite la señal a la siguiente capa.

$$y = f(z) \tag{2}$$

Esta función tiene como objetivo resaltar la información relevante y disminuir la importancia de la información irrelevante. Normalmente, es la misma en las diferentes capas, excepto en la capa de salida, que utiliza una función diferente. Algunos de los tipos de funciones de activación más populares son las siguientes[13]:

- **Función de paso binario (binary step function)**: esta función se basa en umbral para determinar si un nodo debe activarse. La entrada se compara con el umbral y si es mayor, el nodo se activa; de lo contrario, se desactiva y la salida no se transmite a la siguiente capa. Algunas de sus limitaciones incluyen la incapacidad para manejar más de una salida, lo que la hace inadecuada para problemas de clasificación multiclase. Por otro lado, la función de paso binario dificulta la retropropagación debido a que su

derivada es cero en casi todas partes, impidiendo el ajuste de pesos durante el entrenamiento y, por lo tanto, el aprendizaje de la red neuronal.

- **Función de activación lineal (linear activation function)**: también conocida como "sin activación" o "función identidad" (cuando se multiplicada por 1.0), establece que la activación es directamente proporcional a la entrada, no realiza operaciones adicionales sobre la suma ponderada de la entrada, se limita a devolver el valor que se le dio. Tiene dos inconvenientes principales: no es compatible con la retropropagación debido a que la derivada de la función es una contante y no está relacionada con la entrada, y la última capa se convierte en una función lineal de la primera capa, por lo que la función de activación lineal hace que la red neuronal se reduzca a una sola capa. Por lo tanto, esta función limita la capacidad de la red para aprender mapeos complejos entre las entradas y las salidas.
- **Función de activación no lineal (Non-linear activation functions)**: estas funciones resuelven las limitaciones de las funciones de activación lineal. La función está relacionada con la entrada, lo que permite retroceder y entender que pesos de las entradas proporcionan una mejor predicción, lo que posibilita la retropropagación. La salida no es una combinación lineal de la entrada por lo que se pueden apilar numerosas capas de neuronas.

  Entre las funciones no lineales más comunes se encuentran: la función sigmoide la cual toma cualquier valor real como entrada y genera valores dentro del rango de 0 a 1, se utiliza principalmente en modelos donde se desea obtener un porcentaje como salida; ReLU (Rectified Linear Unit, unidad lineal rectificada) se caracteriza por no activar todas las neuronas a la vez, es muy eficiente computacionalmente para realizar comparaciones, es ampliamente utilizada en la detección de objetos; Softmax, que es una combinación de múltiples sigmoides y es muy útiles en problemas de clasificación multiclase.

En la Figura 3 se observa un ejemplo de cada tipo de función de activación:

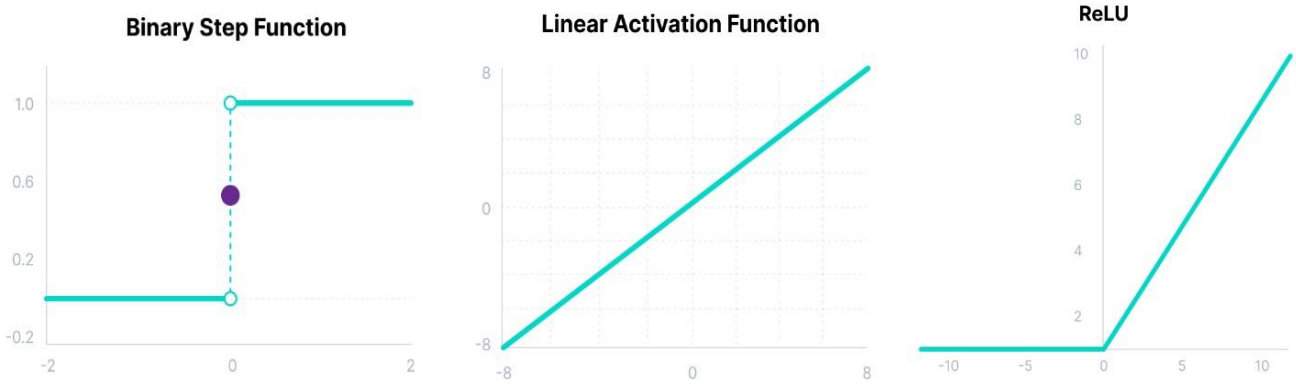


Figura 3. Funciones de activación

En resumen, un nodo recibe un conjunto de datos y realiza una suma ponderada con los pesos asignados a cada entrada. Posteriormente, este resultado se somete a una función de activación que ajusta el valor y lo compara con un umbral determinado para decidir si se activa el siguiente nodo, lo que permite la transición a la capa siguiente. De este modo, la entrada de una capa es la salida de la capa anterior. Este proceso se ilustra en la Figura 4.

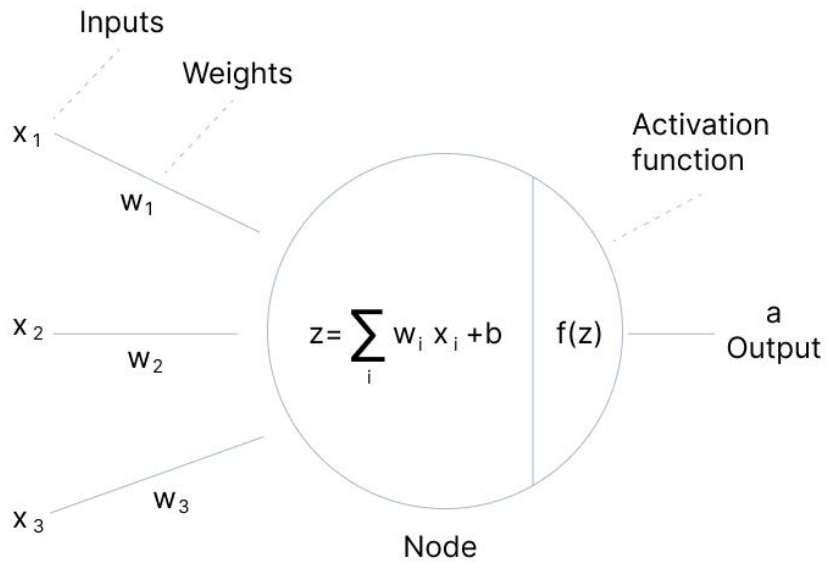


Figura 4. Esquema del proceso de un nodo[13]

Existen dos métodos distintos para la propagación de la información:

- **Propagación hacia adelante (Feedforward propagation)**: en este tipo de propagación, la información siempre avanza hacia las capas siguientes. En cada capa, los datos recibidos de la capa anterior se procesan mediante la función de activación y los resultados pasan a la siguiente capa. Durante este proceso, la salida final se determina en función de los pesos actuales de la red.
- **Retropropagación (Backpropagation)**: en este proceso, los pesos de la red neuronal se ajustan en función del error de predicción. Primero, se calcula el error de la salida comparándola con el valor deseado. Luego, este error se propaga hacia las capas de detrás, recalculando los gradientes de los pesos. Finalmente, se actualizan los pesos utilizando un algoritmo de optimización. Así, el error de predicción se minimiza ajustando los pesos de la red.

2.3.1 Función de perdida y descenso de gradiente[14]

El objetivo del entrenamiento de la red es lograr que el valor de la salida coincida con el valor real. Cuando los datos pasan por la red por primera vez, los pesos están definidos aleatoriamente, por lo que es probable que la predicción sea imprecisa. Para calcular dicho error se realiza una comparación entre el valor de la salida y el valor real mediante una función que se denomina **función de pérdida**. Esta función de pérdida se puede calcular utilizando diferentes métodos, como puede ser el valor absoluto o el valor cuadrático medio, entre otros.

Para minimizar la diferencia entre la predicción y el valor real, y permitir que la red neuronal aprenda automáticamente de los errores y ajuste los pesos, se emplean dos algoritmos: el descenso de gradiente y la retropropagación.

El objetivo del **descenso de gradiente** consiste en optimizar el modelo recalculando los pesos. Para ello, se debe alcanzar el valor mínimo de la función de perdida, como se observa en la Figura 5. En ese punto, el valor de la derivada es cero. Por lo tanto, es necesario calcular la derivada parcial correspondiente al peso de cada variable. De esta forma, se obtiene el vector gradiente, el cual apunta hacia los valores más altos de la curva, por lo que la dirección necesaria es la opuesta.

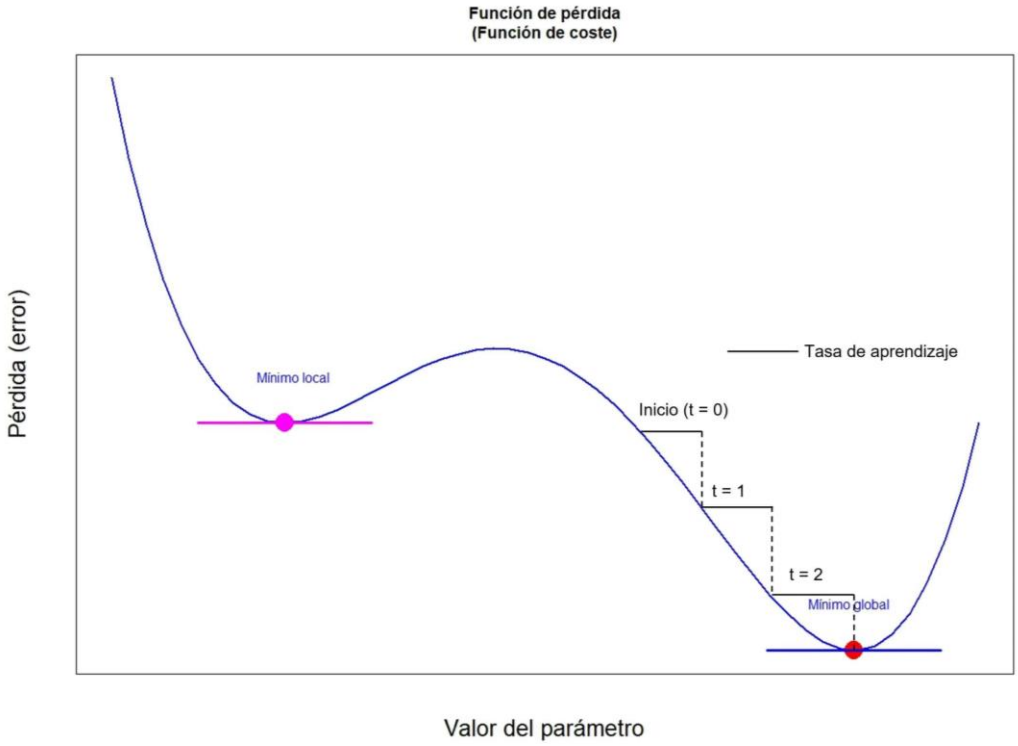


Figura 5. Función de pérdida[14]

Para determinar la cantidad exacta del ajuste se emplea la tasa de aprendizaje. Si la tasa es muy grande o muy pequeña, el modelo puede estancarse y no localizar nunca un mínimo, o puede que localice un mínimo local, que sea más alto que el global, como se muestra en la figura. Cada iteración en la que los datos pasan hacia adelante por la red y se actualizan los pesos en la retropropagación se denomina época.

2.3.2 Redes neuronales convolucionales

Las redes neuronales convoluciones (CNN, por sus siglas en inglés) son un tipo de red neuronal profunda que son efectivas para tareas de procesamiento y reconocimiento de imágenes. La arquitectura está inspirada en cómo se organiza y cómo funciona el cortex visual del ojo humano. Las CNN han demostrado ser altamente efectivas en una gran variedad de aplicaciones, desde el reconocimiento de objetos en imágenes hasta la visión por computadora y el análisis de vídeos.

Las redes convolucionales están formadas por una red multicapa en la que alternan capas convolucionales y de agrupación, y finalmente se encuentra la capa *fully conected*:[15]

- **Capa convolucional**: realiza operaciones entre la entrada y los filtros (kernel) con el objetivo de generar un mapa de características. Estos filtros son matrices de n dimensiones con pesos ajustables que se modifican durante el entrenamiento. La matriz de entrada, compuesta por los píxeles de la imagen en la primera capa y por la salida de la capa anterior en el resto de capas, se somete a un proceso conocido como convolución, donde grupos de pixeles continuos son operados matemáticamente utilizando la matriz *kernel* como filtro. Este filtro se mueve a lo largo de todas las neuronas de entrada, generando así una nueva matriz de salida que se utiliza como entrada para la siguiente capa. La convolución se describe mediante la ecuación (3):

$$u[m,n] = \sum_i \sum_j \sum_k s[i,j,k]h[m-i,n-j,o-k] \tag{3}$$

Aquí, *s* representa la matriz de entrada y *h* corresponde al *kernel* o respuesta al impulso del filtro. Este proceso se puede observar en la Figura 6:

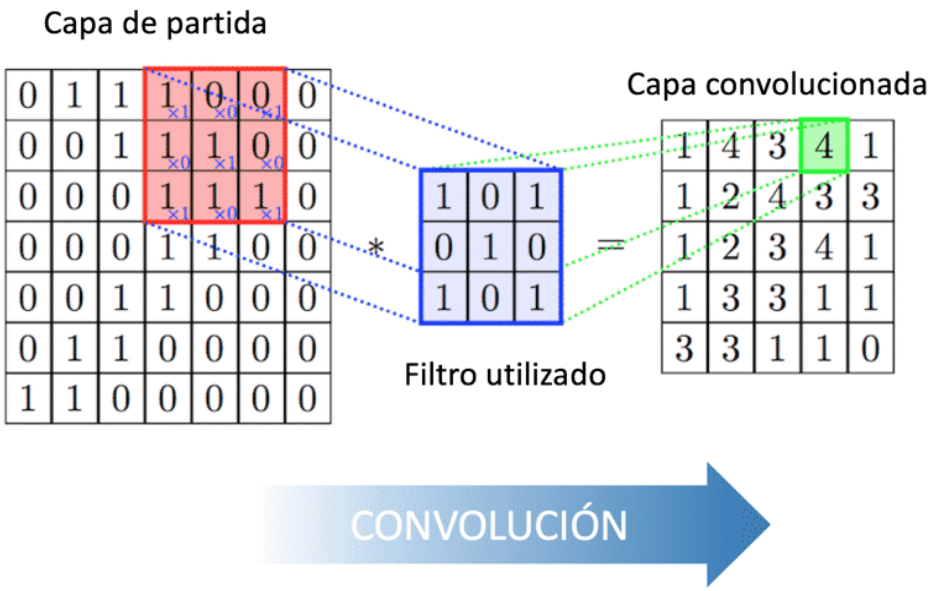


Figura 6. Proceso de la capa convolucional[16]

Una ventaja fundamental de la convolución es que el mismo filtro se emplea para extraer las características de cualquier parte de la imagen, lo que reduce significativamente la cantidad de conexiones y parámetros a entrenar en comparación con una red neuronal completamente interconectada.

Una vez aplicada la convolución, se aplica una función de activación a los mapas de características resultantes. La función más utilizada es la ReLU por su simplicidad y eficacia. Es crucial elegir una tasa de aprendizaje adecuada y supervisar las neuronas que quedan inactivas. Otras funciones como Leaky ReLU o Maxout también pueden ser utilizadas.

- **Capa de agrupación**: esta capa reduce la dimensionalidad y los parámetros de entrada aplicando filtros sin pesos que agregan información mediante funciones como máximo o promedio. Este proceso, conocido como reducción de muestreo, preserva características clave mientras mejora la eficiencia y evita el sobreajuste. Es una técnica esencial para simplificar modelos CNNs manteniendo su capacidad predictiva. Este proceso se observa en la Figura 7.

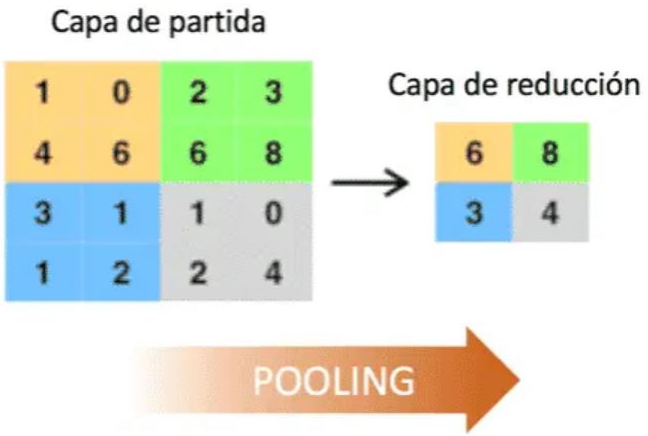


Figura 7. Proceso de la capa de agrupación[16]

- **Capa *fully conected* (FC)**: esta capa se caracteriza por conectar cada nodo de salida directamente con todos los nodos de la capa anterior. Esta capa final en la red neuronal realiza una clasificación utilizando las características obtenidas en las capas previas y sus filtros. Por otro lado, las capas convolucionales y de agrupación generalmente utilizan funciones de activación ReLU. Mientras que, las capas FC utilizan típicamente la función Softmax con el objetivo de producir salidas de clasificación probabilísticas entre 0 y 1.

Desde su aparición, las CNN han evolucionado significativamente como una clase de red neuronal profunda. Están especialmente diseñadas para el análisis de imágenes en diversas aplicaciones, incluyendo segmentación, clasificación y detección de objetos. Entre las arquitecturas más influyentes se encuentran: LeNet-5, AlexNet, VGGNet y ResNet.

LeNet-5, una de las primeras CNN desarrolladas por Yann LeCun y colaboradores, diseñada originalmente para la clasificación de dígitos manuscritos. Fue pionera en el uso de capas convolucionales y de submuestreo.

AlexNet, que ganó el desafío ImageNet en 2012, representó un avance significativo al introducir el uso generalizado de la función de activación ReLU y la técnica de *dropout* para mejorar el rendimiento y evitar el sobreajuste en redes profundas.

VGGNet, conocida por su estructura simple y el uso de múltiples capas convolucionales con filtros de tamaño reducido, estableció un estándar en la construcción de redes profundas para la clasificación de imágenes.

ResNet, destacada por sus conexiones residuales, permitió desarrollar redes mucho más profundas al mitigar el problema del desvanecimiento del gradiente. Esta arquitectura fue crucial para alcanzar nuevas alturas en la precisión de clasificación de imágenes.

Estas arquitecturas han sido fundamentales en el avance del procesamiento de imágenes mediante CNN, demostrando mejoras significativas en precisión y eficiencia en una amplia gama de aplicaciones prácticas.

2.4 Detección de objetos

La detección de objetos es un campo de la visión por computadora que se centra en localizar y clasificar objetos dentro de una imagen o un vídeo. Las aplicaciones de la detección de objetos abarcan desde la conducción autónoma hasta la vigilancia, la atención médica y más. Diversos algoritmos y técnicas se han desarrollado para abordar este desafío, entre los que se destacan R-CNN, Fast R-CNN, Faster R-CNN y YOLO.[17]

2.4.1 R-CNN (Region-based Convolutional Neural Networks)

R-CNN fue uno de los primeros enfoques que combinó técnicas de selección de regiones con CNN. El proceso de R-CNN consta de tres pasos principales, los cuales se pueden observar en la Figura 8: Generación de propuestas de regiones, donde se utiliza un método de búsqueda selectiva para generar numerosas propuestas de regiones de interés en una imagen; extracción de características, en la que cada una de estas regiones se redimensiona y se pasa a través de una CNN preentrenada para extraer características; clasificación y refinamiento, donde las características extraídas se introducen en un clasificador SVM (Suport Vector Machine) para determinar la clase del objeto y un modelo de regresión lineal para ajustar las cajas delimitadoras. Aunque es efectivo, R-CNN es computacionalmente intensivo y lento debido a la necesidad de aplicar la CNN a muchas regiones por imagen.

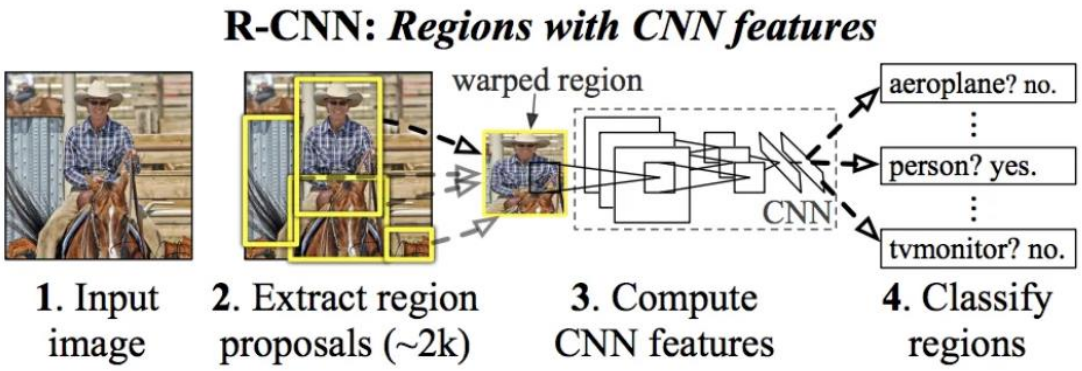


Figura 8. Proceso de las R-CNN[17]

2.4.2 Fast R-CNN

Fast R-CNN mejoró la eficacia de R-CNN integrando la generación de regiones de interés y la clasificación en un único proceso más cohesivo. El procesamiento de la imagen completa se realiza a través de una CNN para generar un mapa de características. Las regiones de interés

se proyectan sobre el mapa de características y se redimensiona a través de una operación de *RoI Pooling*. Finalmente, las características resultantes se pasan a través de capas totalmente conectadas y se utilizan para clasificar cada región y ajustar las cajas delimitadoras. Este método reduce significativamente el tiempo de procesamiento al evitar la necesidad de analizar múltiples regiones individualmente, mejorando tanto la velocidad como la precisión.

2.4.3 Faster R-CNN

Faster R-CNN introdujo una Red de Propuestas de Regiones (RPN) para mejorar aún más la eficacia. La RPN se entrena para generar propuestas de regiones de interés directamente desde los mapas de características de la CNN. Las regiones propuestas por la RPN se proyectan sobre el mapa de características, seguido por un refinamiento y clasificación similar a Fast R-CNN, el proceso se observa en la Figura 9. La integración de la RPN permite que el modelo aprenda a generar regiones de interés más efectivamente durante el entrenamiento, eliminando la necesidad de métodos de búsqueda selectiva y acelerando el proceso de detección.

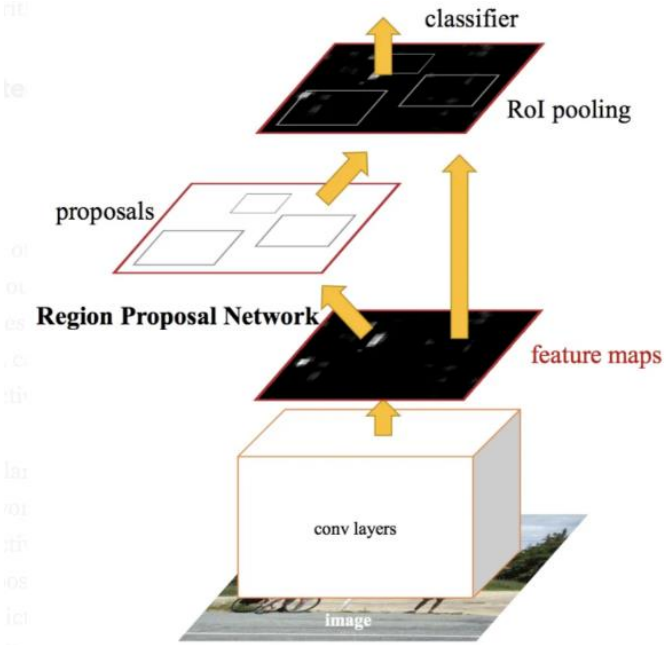


Figura 9. Proceso Faster R-CNN[17]

2.4.4 YOLO (You Only Look Once)

YOLO[18] adopta un enfoque completamente diferente a los sistemas basados en regiones. En lugar de generar propuestas de regiones, YOLO trata la detección de objetos como un problema de regresión única, dividiendo la imagen en una cuadricula de celdas y asignando una *bounding box* y una probabilidad de clase a cada celda. Esto permite a YOLO realizar detecciones extremadamente rápidas en tiempo real, esto se observa en la Figura 10. Cada celda de la cuadricula predice un número fijo de *bounding boxes* y las probabilidades de clase para cada caja. Cada predicción incluye una confianza que indica la precisión de la caja y la probabilidad de que contenga un objeto.

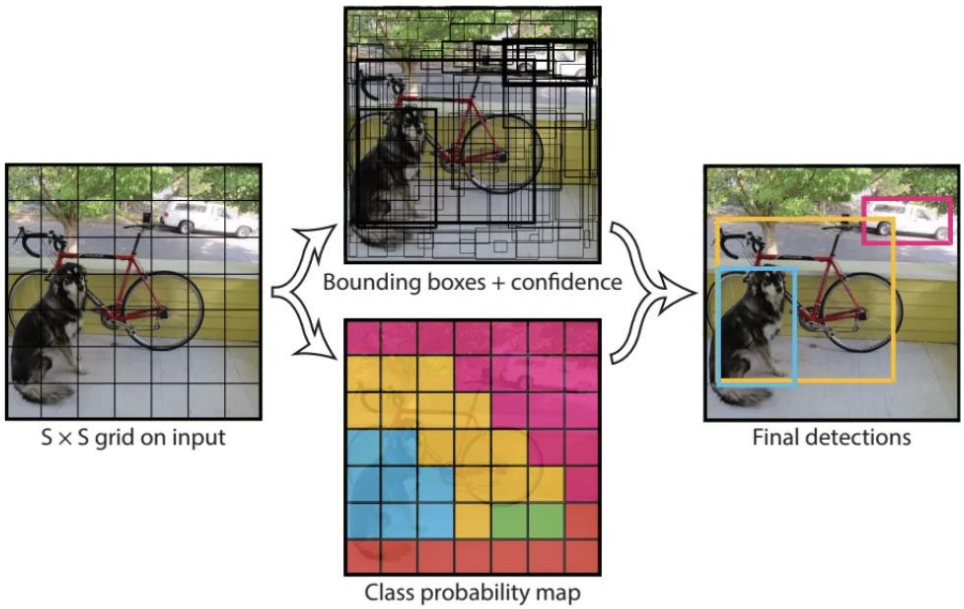


Figura 10. Método aplicado por YOLO[17]

YOLO presenta varias ventajas clave que lo hacen ideal para aplicaciones en tiempo real. La principal ventaja es su velocidad, ya que puede procesar hasta 45 frames por segundo en su versión original. Esto es posible porque YOLO considera la imagen completa de una sola vez, lo que significa que no se necesita una generación de propuestas de regiones, a diferencia de los métodos de R-CNN. Además, YOLO es altamente eficiente en términos de cálculo, lo que permite implementarlo en hardware con recursos limitados, como dispositivos móviles o cámaras inteligentes.

Otra ventaja significativa de YOLO es su capacidad para generalizar mejor a nuevas imágenes. Debido a su enfoque unificado, YOLO aprende representaciones globales de los objetos, lo que mejora su precisión al detectar objetos en diversas situaciones y contextos. Las versiones más recientes de YOLO han mejorado tanto en precisión como en velocidad, convirtiéndolo en uno de los métodos más populares para la detección de objetos.

2.4.5 Cajas delimitadoras y máscaras

Una vez identificado el objeto, es necesario marcarlo de alguna manera. Como se puede ver en la Figura 11, se pueden emplear cajas delimitadoras (*bounding boxes*) y máscaras para este propósito.

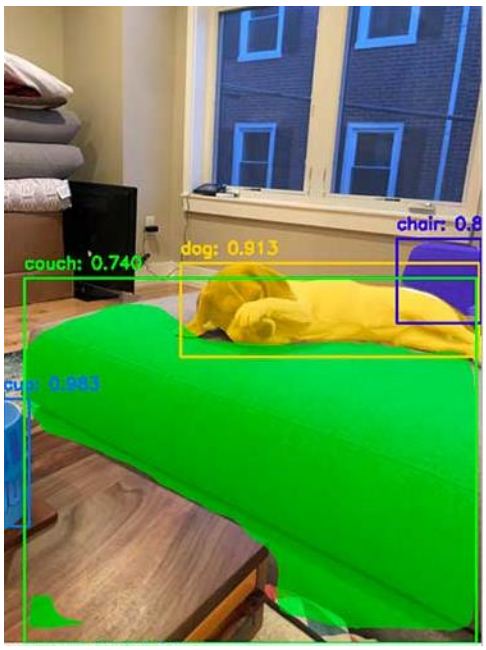


Figura 11. Ejemplo de cajas delimitadoras y máscaras[19]

Las cajas delimitadoras son esenciales para la detección de objetos, ya que permite al modelo aprender a identificar y localizar objetos en condiciones diversas.

En el contexto de YOLO, las cajas delimitadoras se definen utilizando un formato específico que optimiza la detección rápida y precisa. El formato YOLO utiliza coordenadas normalizadas y centradas para describir las cajas delimitadoras. Cada caja se representa mediante cuatro parámetros: ($x_{central}$, $y_{central}$, ancho, alto). Las coordenadas $x_{central}$ e $y_{central}$ indican el centro de la caja y están normalizadas en el rango [0, 1] en relación con el tamaño de la imagen. El ancho y el alto también están normalizados, lo que facilita la integración de las predicciones en una estructura de cuadrícula.[20]

Las máscaras son utilizadas principalmente en tareas de segmentación de imágenes, donde se necesita identificar la forma exacta de los objetos. [21]

Las máscaras de segmentación definen con precisión los contornos y formas de los objetos presentes en la imagen, lo cual es crucial para que los modelos aprendan a identificar y segmentar objetos de manera precisa. Esto no solo aumenta la variabilidad del conjunto de datos, sino que también ayuda a los modelos a aprender características más robustas y generalizables.

3. Especificaciones y restricciones de diseño

Este proyecto tiene como objetivo principal desarrollar un sistema automatizado basado en el aprendizaje profundo, capaz de segmentar y realizar el seguimiento de una trucha presente en diferentes vídeos.

3.1 Especificaciones de diseño

- El sistema ha sido entrenado utilizando aprendizaje por transferencia (*transfer learning)* para ajustar el modelo a los resultados deseados.
- El sistema está diseñado para localizar una sola trucha en un entorno especifico.
- El sistema divide la imagen en secciones utilizando las líneas negras visibles como guía.
- El sistema es capaz de segmentar la trucha y devolver como solución la sección donde se encuentra en cada momento.

3.2 Restricciones de diseño

- Para entrenar al modelo se ha utilizado la versión gratuita de *Google Colab*, lo que conlleva un tiempo limitado del uso de la GPU (Unidad de Procesamiento Gráfico).
- El sistema ha sido desarrollado en el lenguaje de programación Python 3.
- El proyecto ha sido implementado utilizando librerías de código abierto.
- El etiquetado no se ha sido realizado por un experto, por lo que puede ser impreciso.
- Para el entrenamiento se han empleado únicamente 105 imágenes de entrenamiento y 26 imágenes de validación. Las imágenes se han obtenido de *frames* de algunos vídeos.
- El sistema debe ser capaz de detectar a la trucha en cada uno de los *frames* del vídeo.
- El sistema debe ser autónomo sin la necesidad de la intervención humana.

4. Descripción de la solución propuesta

En este apartado se proporciona una visión detallada del desarrollo del sistema, abarcando tanto el modelo de detección como los distintos métodos empleados para el seguimiento de la trucha.

Para la implementación del código, se ha utilizado el lenguaje de programación Python y las siguientes librerías:

- **OpenCV**:[3] empleada para el procesamiento de imágenes y vídeos.
- **NumPy**:[22] para la manipulación de matrices y operaciones numéricas.
- **Ultralytics YOLO**:[23] utilizada para la detección de objetos mediante el algoritmo YOLO.
- **Pandas**:[24] para almacenar y manipular los datos de los fotogramas y las secciones, así como para guardar los resultados en un archivo Excel.
- **Tkinter**:[25] esta biblioteca se utiliza para crear una interfaz gráfica para el usuario.

Todo el proceso de entrenamiento se ha realizado utilizando *Google Colaboratory*,[26] en combinación con *Google Drive*. El resto del desarrollo se ha realizado empleando la herramienta *Visual Studio Code*. [27]

4.1 Descripción de los datos

Para este proyecto, se dispone de un conjunto de 30 vídeos en formato MTS, cada uno con una duración aproximada de 5 minutos y medio. Los vídeos están etiquetados como "Clip X", donde X es un número del 61 al 90. Cada vídeo presenta una resolución de 1920x1080 a 25 fps (fotogramas por segundo) y muestra una pecera con una trucha única. La pecera está dividida visualmente en secciones mediante cintas negras, con un fondo rocoso marrón que dificulta la visibilidad de la trucha, como se puede ver en la Figura 12.

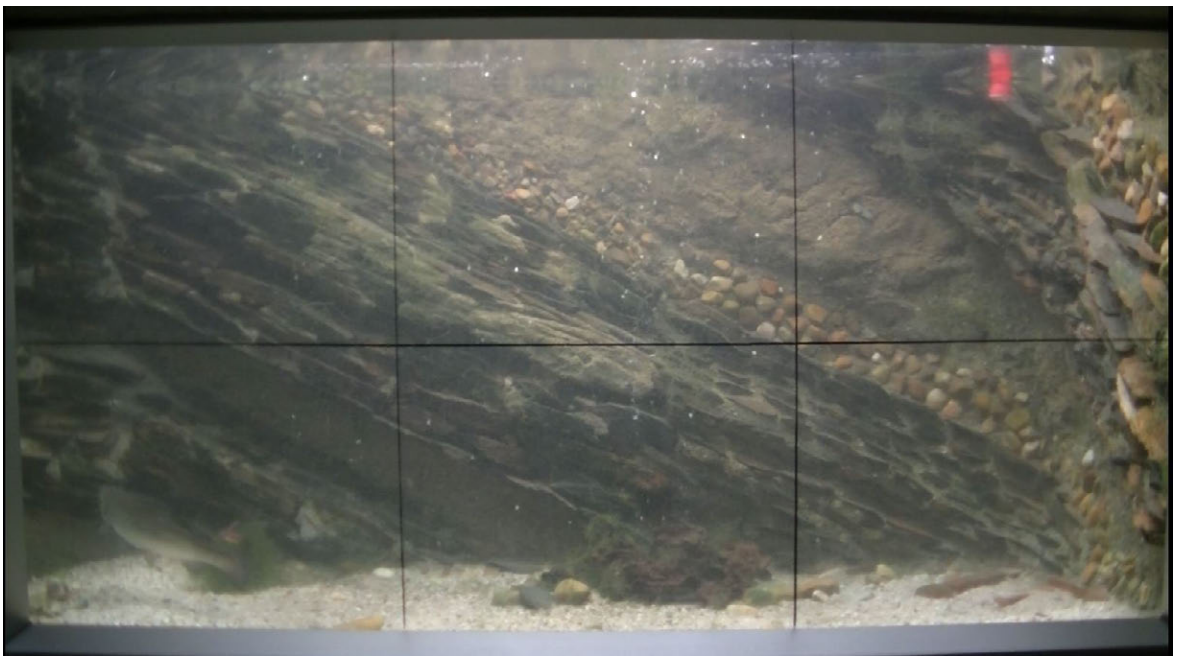


Figura 12. Fotograma de un vídeo

El propósito del experimento es realizar un estudio del comportamiento de las truchas mediante una prueba *open field*. Consiste en observar los vídeos y registrar el momento exacto en que la trucha cambia de una sección a otra. Actualmente, estas anotaciones se realizan manualmente por una persona que analiza cada vídeo.

Para la verificación de los resultados del sistema automático, se cuenta con un registro en Excel que contiene "el grabado y etiquetado manual se llevó a cabo por los técnicos Juan Enrique Barrios Sánchez (Barrios Sánchez, 2023) y Almudena Gallego Fernández, bajo supervisión de los investigadores Álvaro De la Llave-Propín y Morris Villarroel Robinson" [28]. Este archivo proporciona información crucial como el número de clip de cada vídeo y el fotograma en que la trucha cambia de sección, determinado por el momento en que la cabeza completa de la trucha cruza de una sección a otra.

4.2 Etiquetado de las imágenes

Para que la red entienda cual es la respuesta deseada en la salida, es fundamental realizar un etiquetado previo que indique la respuesta correcta. Este proceso de etiquetado es esencial para entrenar al modelo de detección de manera efectiva.

Para llevar a cabo el proceso de etiquetado, se han extraído numerosos fotogramas de varios vídeos y se ha realizado un etiquetado detallado sobre estas imágenes. Para esta tarea, se ha utilizado una herramienta especializada llamada LabelMe.[29] Como se puede observar en la Figura 13, LabelMe permite delinear con precisión el contorno de la trucha en cada imagen, facilitando un etiquetado exacto y fiable.

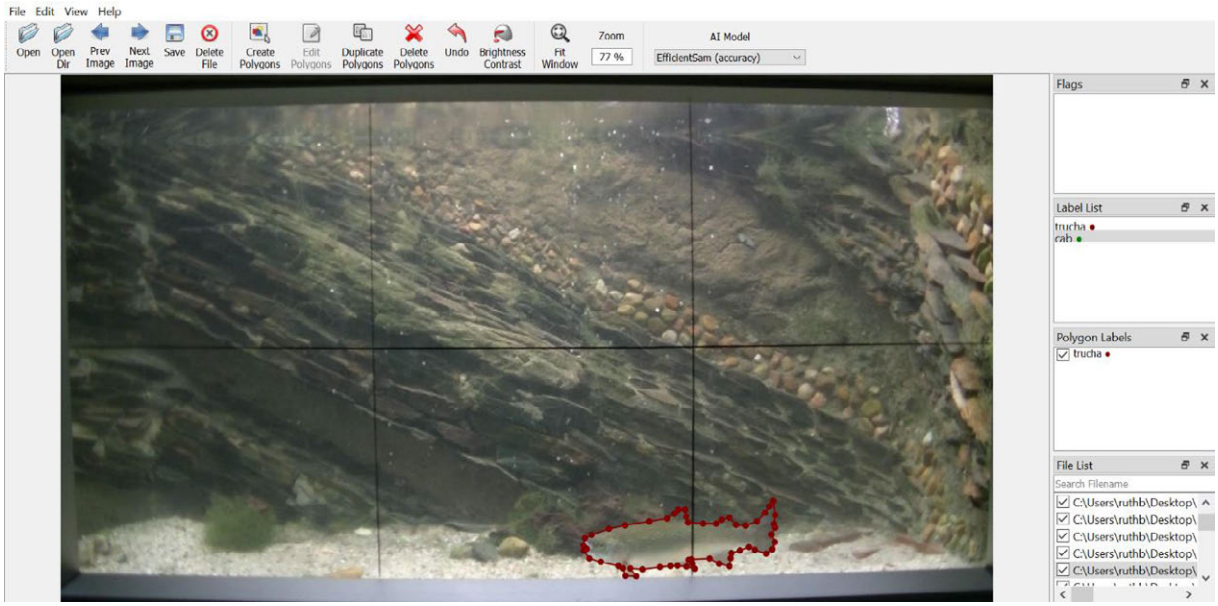


Figura 13. Sistema de etiquetado LabelMe

El proceso de etiquetado con LabelMe involucra varios pasos. Primero, se seleccionan los fotogramas relevantes de los vídeos que contienen las truchas que se desean detectar y seguir. A continuación, se carga cada fotograma en la herramienta LabelMe, donde se procede a dibujar cuidadosamente los contornos de las truchas utilizando la opción "*Create Polygons*". Este proceso es manual y requiere precisión para asegurarse de que las etiquetas sean lo más exactas posible, ya que la calidad del etiquetado impacta directamente en la precisión del modelo de detección.

Una vez que se ha delineado el contorno de la trucha, se le asigna una etiqueta para identificarla, en ese caso, se utiliza la etiqueta "*trucha".* Después de etiquetar las truchas en cada fotograma, se generan archivos de anotación en formato *JSON,* los cuales contienen las coordenadas de los contornos dibujados. Estos archivos de anotación son esenciales para entrenar el modelo de detección, ya que proporcionan la información necesaria para que la red neuronal aprenda a identificar y seguir las truchas en los vídeos.

Para el entrenamiento, es necesario cambiar el formato de anotación. Por ello, se ha utilizado un script llamado "*labelme2yolo",* que convierte las anotaciones de las imágenes del formato LabelMe, que guarda información sobre las formas y las etiquetas de los objetos en archivos *JSON,* al formato YOLO. En este formato, cada imagen tiene un archivo de texto asociado que contiene el tipo de objeto y las coordenadas, todos los valores normalizados con respeto a las dimensiones de la imagen, en un rango de 0 a 1.

4.3 Entrenamiento del sistema

El modelo empleado en el desarrollo de este proyecto es el yolov8n-seg. Este modelo pertenece a la subcategoría de YOLOv8 enfocada en la segmentación de instancias, lo que significa que no solo detecta y clasifica objetos dentro de una imagen, sino que también proporciona una máscara precisa delineando cada objeto detectado.

La denominación “n” indica que se trata de un modelo de tamaño pequeño dentro de la familia YOLOv8. Este modelo ha sido elegido por su alta velocidad de procesamiento, además de cumplir con las exigencias de detección requeridas. Esto ofrece un equilibrio adecuado entre la capacidad de detección y los recursos computacionales necesarios para este proyecto específico.

Para el entrenamiento del modelo final, se han utilizado un total de 105 imágenes para el entrenamiento y 26 imágenes para la validación. Ha sido necesario dividir las imágenes y los archivos de anotaciones en varias carpetas para facilitar el entrenamiento. Inicialmente, las imágenes y los archivos de texto se organizan en dos carpetas principales: *labels* e *images*. Dentro de cada una de estas carpetas, se ha realizado una subdivisión adicional en carpetas de entrenamiento (*train*) y validación (*val*). Además, se es necesario un *dataset* que contiene la información de los *paths* de las carpetas de imágenes y archivos de anotaciones, así como las etiquetas de los objetos. Este *dataset* estructurado permite al modelo acceder fácilmente a los datos necesarios durante las fases de entrenamiento y validación.

El proceso de entrenamiento se ha llevado a cabo con 250 épocas. Este número de épocas ha sido seleccionado para garantizar que el modelo pueda aprender adecuadamente las características y variaciones presentes en los datos, permitiendo así una segmentación y detección precisa de las truchas.

Se ha llevado a cabo una prueba para verificar la efectividad del entrenamiento. Para ello se han utilizado 37 imágenes variadas que incluyen distintas condiciones de agua, desde clara hasta turbia, así como diversas posiciones de las truchas. El objetivo es asegurar la correcta detección de las truchas utilizando el modelo entrenado.

Como se muestra en la Figura 14, se han obtenido resultados con una confianza superior a 0,7, por lo que se puede decir que son satisfactorios.

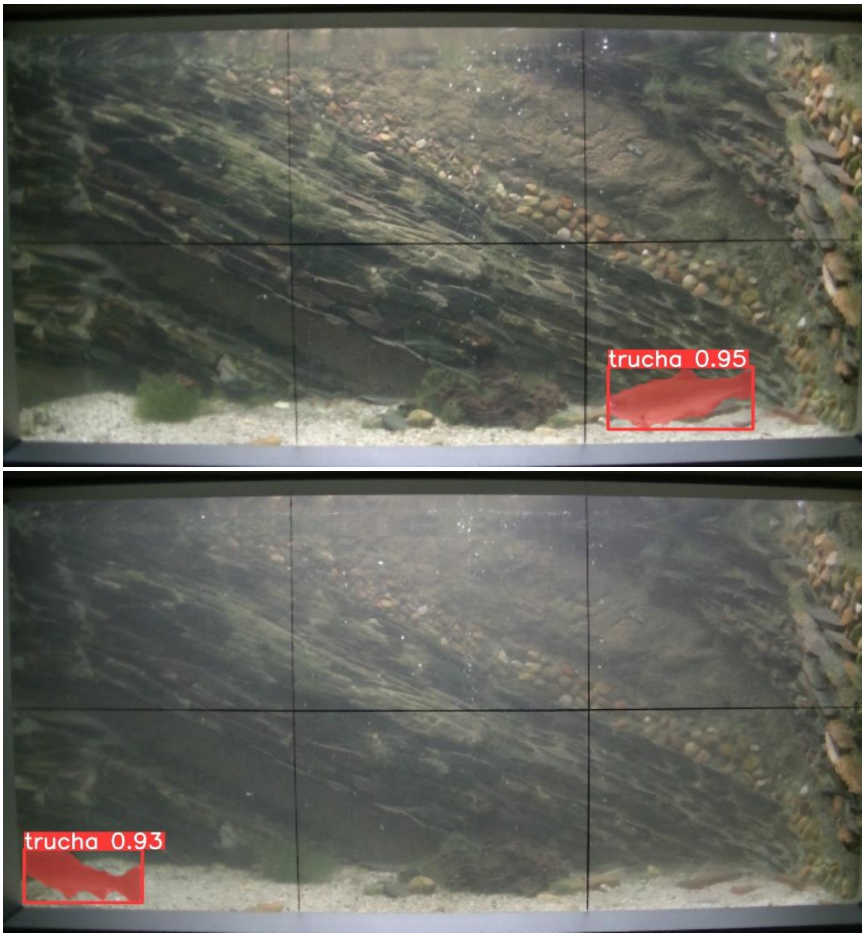


Figura 14. Imágenes de prueba del modelo entrenado

4.4 Seguimiento de la trucha

Los vídeos muestran una pecera dividida en seis secciones, marcadas por dos líneas verticales negras y una horizontal, numeradas de izquierda a derecha y de arriba hacia abajo, como se observa en la Figura 15. El objetivo de este proyecto es detectar la trucha e indicar en que sección se encuentra.

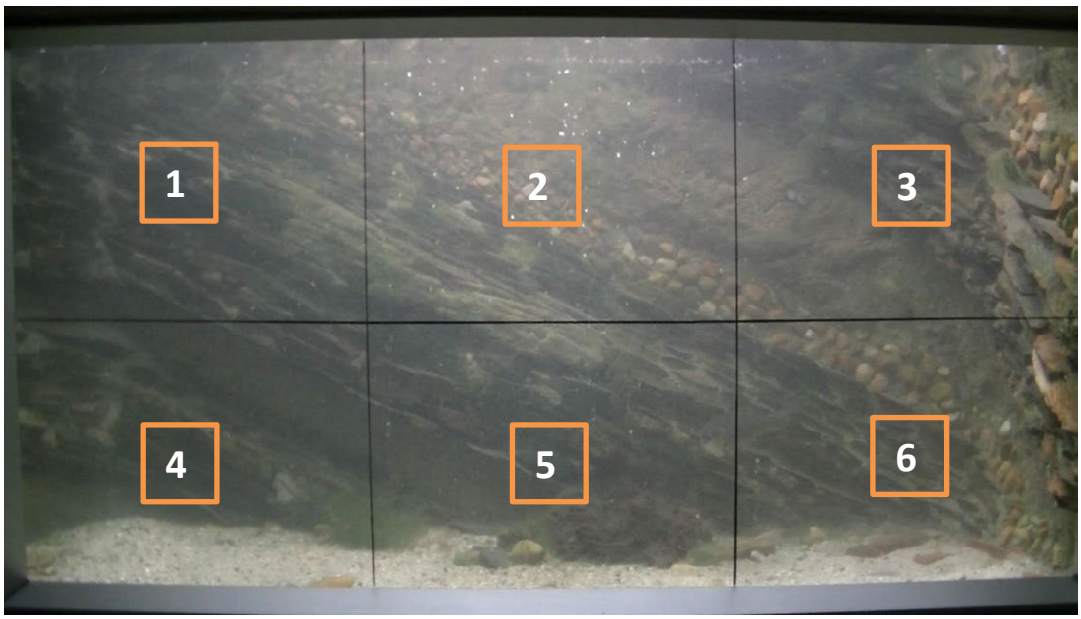


Figura 15. Numeración de las secciones

Para detectar las líneas, se ha utilizado el detector de bordes *Canny* sobre los 10 primeros *frames* capturados. Se han empleado matrices para acumular las sumas de los bordes horizontales y verticales, capturando así los bordes detectados en cada dirección. Posteriormente, se han identificado los límites de interés tanto verticales como horizontales dentro del vídeo, que se han determinado mediante la búsqueda de máximos locales en las sumas de los bordes acumuladas.

Una vez entrenado el modelo y detectada la trucha, se propone localizar en que sección se halla, y para ello se proponen dos métodos diferentes: utilizando el centroide de la *bounding box* o utilizando la posición de la cabeza de la trucha.

4.4.1 Seguimiento con centroide

La primera propuesta para indicar en qué sección se encuentra la trucha se basa en el uso del centroide de la *bounding box*. Este método implica analizar cada *frame* del vídeo para detectar la trucha y obtener sus coordenadas. Una vez detectada, se calcula el centroide de la *bounding box*, es decir, el punto central de la caja delimitadora que encierra la trucha. Para determinar este punto, primero se extraen las coordenadas del cuadro delimitador (x_min, y_min, x_max, y_max). Luego, se aplica la ecuación (4):

$$centroide = \left(\frac{x_min + x_max}{2}, \frac{y_min + y_max}{2}\right) \quad (4)$$

Este centroide se compara con los límites predefinidos de las distintas secciones. De esta manera, se puede determinar en qué sección específica se encuentra la trucha en cada *frame*, como se puede observar en la Figura 16.

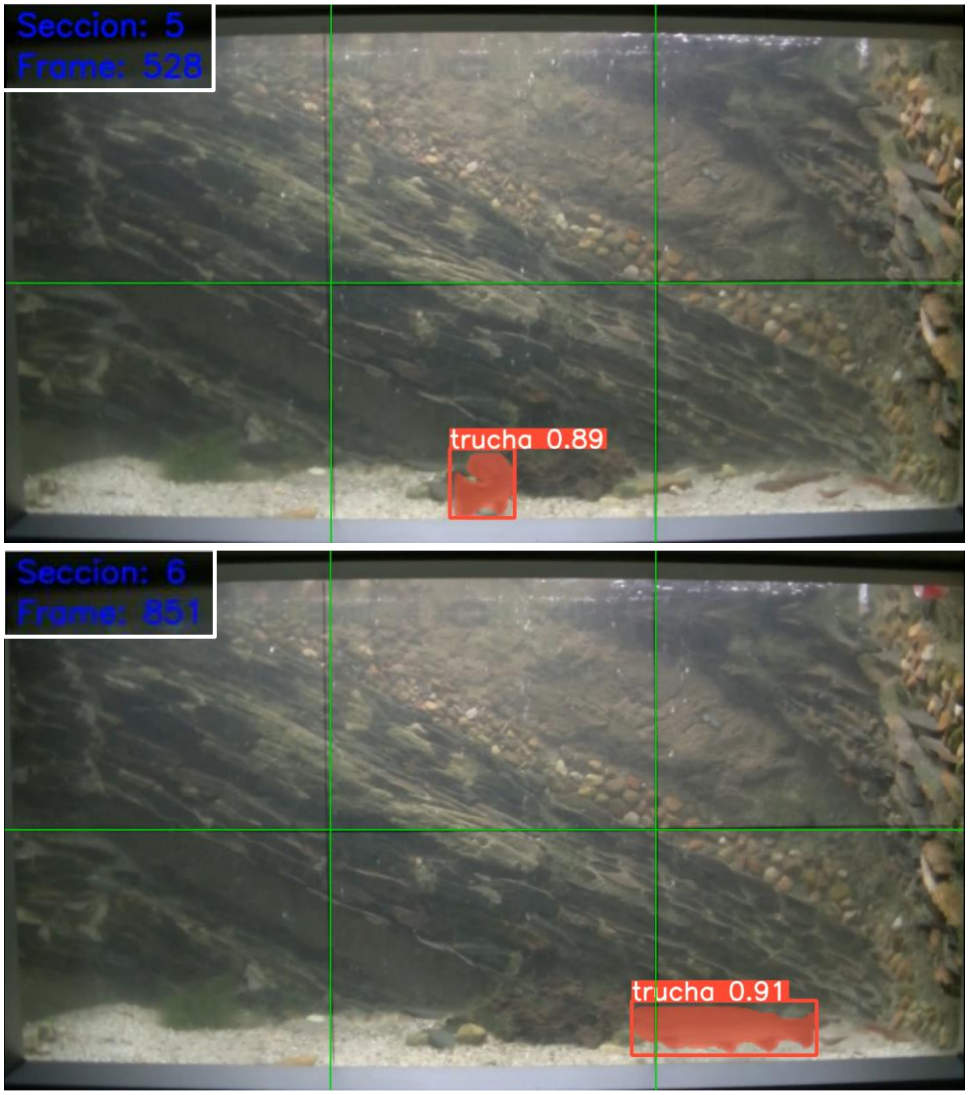


Figura 16. Ejemplo de seguimiento con centroide

El proceso se repite para cada *frame* del vídeo, proporcionando una actualización continua de la ubicación de la trucha. En el caso de que ocurra un error en la detección de la trucha en algún *frame*, el sistema conserva la sección determinada en la última detección exitosa. Esta estrategia de mantener la última posición conocida asegura la estabilidad y continuidad de la información, evitando variaciones indeseadas debido a detecciones fallidas o ausentes. La sección se actualizará únicamente cuando se realice una nueva detección precisa de la trucha, garantizando así que la información de ubicación sea lo más confiable y precisa posible a lo largo del tiempo.

4.4.2 Seguimiento con la cabeza como referencia

La segunda opción planteada para determinar en qué sección se encuentra la trucha se basa en utilizar la posición de su cabeza como referencia.

Para implementar este método, inicialmente se llevó a cabo un entrenamiento utilizando imágenes etiquetadas con la ubicación de la cabeza de la trucha. Durante las pruebas realizadas con imágenes estáticas, los resultados fueron satisfactorios, indicando una detección precisa de la cabeza, como se muestra en la Figura 17.

Figura 17. Detección de la cabeza por parte del modelo

No obstante, al aplicar este método en un vídeo y analizar *frame* a *frame*, los resultados no fueron tan positivos. En la mayoría de los casos, el sistema solo detectaba la trucha sin identificar la posición de la cabeza.

Ante esta limitación, se propuso una nueva metodología para localizar la cabeza de la trucha. Esta nueva aproximación se basa en ciertas suposiciones: primero, que la trucha siempre avanza hacia adelante, lo que implica que la cabeza se encuentra en la dirección del movimiento. Segundo, se supone que la trucha se desplaza principalmente en sentido horizontal. Basándose en estas hipótesis, se desarrolla una posible solución que consiste en comparar la posición de la trucha en el *frame* actual con su posición en los *frames* anteriores para determinar la dirección del movimiento.

Después de realizar múltiples pruebas, se concluye que el mejor resultado se obtiene al comparar la dirección de movimiento con los 7 *frames* anteriores. La implementación de esta solución muestra cierta efectividad, proporcionando una detección más fiable de la posición de la cabeza en los vídeos analizados. Las soluciones obtenidas con este nuevo método se ilustran en la Figura 18, mostrando cómo la comparación con los *frames* anteriores permite una identificación más precisa de la cabeza de la trucha en movimiento.

Figura 18. Detección de la cabeza basándose en el movimiento

A diferencia de la Figura 16, en la Figura 18, a pesar de ser el mismo *frame*, el resultado varía según la referencia utilizada: cuando se utiliza el centroide de la *bounding box* como referencia, la trucha se localiza en la sección 6, mientras que, al usar la posición de la cabeza como referencia, se localiza en la sección 5.

4.5 Parámetros de detección [30]

A la hora de realizar la detección de la trucha utilizando YOLOv8, se pueden ajustar varios parámetros para mejorar la precisión y eficiencia del modelo. Algunos de los parámetros más importantes que se han ajustado en este proyecto son:

- **Confianza (*conf*)**: Este parámetro determina el umbral mínimo de confianza para que una detección sea considerada válida. En este proyecto, se ha considerado aceptable un valor de 0.65. Esto significa que solo las detecciones con una probabilidad de al menos 65% de ser correctas se aceptan como válidas, lo que ayuda a reducir las falsas detecciones y asegurar una mayor precisión.
- **Intersección sobre unión (*IoU*)**: El umbral de IoU se utiliza para evaluar la calidad de las predicciones durante el entrenamiento y la evaluación del modelo. IoU mide la superposición entre la *bounding box* predicha y la *bounding box* real (*ground truth*). Para que una predicción se considere correcta, la superposición debe alcanzar el umbral definido. En este proyecto, se ha optado por un valor alto de 0.7. Este valor asegura que las *bounding boxes* predichas deben coincidir en al menos un 70% con las cajas reales, lo que es particularmente útil si en el futuro se necesita localizar más de una trucha por imagen, proporcionando una detección más precisa y reduciendo las falsas positivas.
- **Tamaño de la imagen de entrada (*Imgsz*)**: Este parámetro define el tamaño de la imagen que se usa durante la detección, independientemente del tamaño original de la imagen. En este proyecto, las imágenes originales son de 1920x1080 píxeles, pero para el entrenamiento, se ha utilizado un valor de 640 píxeles. Este dimensionamiento

de la imagen ayuda a equilibrar la carga computacional y la precisión del modelo. Para las detecciones, se han realizado pruebas tanto con un tamaño de 640 como con 320 píxeles, ya que un valor menor puede reducir significativamente el tiempo de ejecución, aunque a costa de una posible pérdida de precisión.

- **detección máxima (*max_det*)**: Este parámetro indica el número máximo de detecciones permitidas por imagen. Dado que en este proyecto se trabaja exclusivamente con vídeos que contienen una sola trucha, este valor se ha fijado en 1. Establecer este parámetro en 1 ayuda a optimizar el rendimiento del modelo, enfocando todos los recursos en detectar una única trucha por *frame* y evitando así el procesamiento innecesario de múltiples detecciones que no son relevantes.

4.6 Arquitectura del sistema

En la Figura 19 se presenta un diagrama que ilustra el funcionamiento del sistema propuesto.

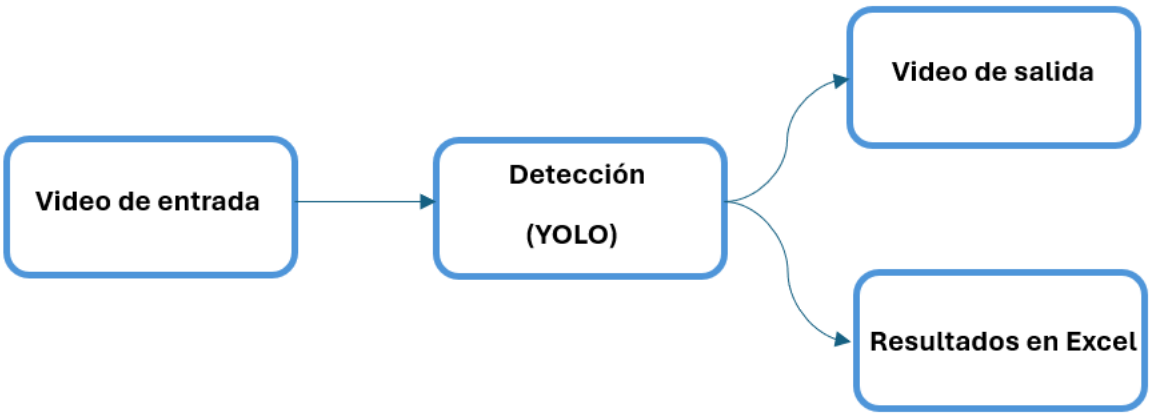


Figura 19. Diagrama del sistema

Inicialmente, el proceso comienza con la introducción de un vídeo que se analiza *frame* a *frame* utilizando un detector YOLO para identificar la ubicación precisa de una trucha. Este análisis se traduce en un vídeo procesado que delimita las secciones pertinentes y presenta en la esquina superior izquierda cada *frame* con la sección actualmente observada. Además de la visualización en tiempo real, el sistema genera un archivo Excel detallando los momentos exactos en que la trucha cambia de una sección a otra, junto con el número de *frame* correspondiente para cada transición. Este enfoque no solo facilita la observación y seguimiento de la trucha en el vídeo, sino que también proporciona datos estructurados que pueden ser utilizados para un análisis más profundo y sistemático de su comportamiento y movimientos. Además, permite comparar con los datos de referencia.

5. Resultados

5.1 Resultados del entrenamiento

A lo largo de todo el proyecto se han realizado numerosos entrenamientos con el objetivo de encontrar el sistema más eficaz posible. Para evaluar el rendimiento de los modelos entrenados, se han analizado diferentes parámetros de salida:[31]

- Matriz de confusión: utilizada para visualizar las predicciones correctas e incorrectas de manera desglosada, mostrando los TP (true positive), FP (false positive), FN (false negative) y TN (true negative). La precisión del sistema se calcula como la proporción de predicciones positivas correctas sobre el total de predicciones realizadas por el modelo:[32]

$$Precisión = \frac{TP}{TP + FP} \quad (5)$$

  También se puede evaluar la exactitud del sistema lo que se traduce en la proporción de predicciones correctas, tanto positivas como negativas, sobre el total de predicciones realizadas por el modelo:

$$Exactitud = \frac{TP + TN}{TP + FP + FN + TN} \quad (6)$$

- dls loss (pérdidas de regresión): mide la precisión de las predicciones de las coordenadas de las *bounding boxes*, penalizando las diferencias entre las coordenadas predichas y las verdaderas. Un valor cercano a 0 indica que las coordenadas predichas están muy cerca de las coordenadas verdaderas.
- mAP (precisión media):[33] métrica utilizada para evaluar la precisión general de un modelo de detección de objetos, calculada como el promedio de la precisión promedio (AP) para todas las clases. Un valor cercano a 1 indica un alto rendimiento del modelo. La mAP50-95 es una versión más detallada de mAP que evalúa la precisión en diferentes niveles de superposición entre las predicciones y las anotaciones verdaderas.

5.1.1 Matriz de confusión

Durante el desarrollo del proyecto, se han realizado múltiples entrenamientos para alcanzar una precisión óptima del modelo. La evaluación de su eficacia ha involucrado el análisis de varios parámetros, como la matriz de confusión. Esta herramienta, mostrada en la Figura 20, permite identificar claramente las predicciones correctas e incorrectas, ayudando a mejorar el rendimiento del modelo mediante ajustes precisos.

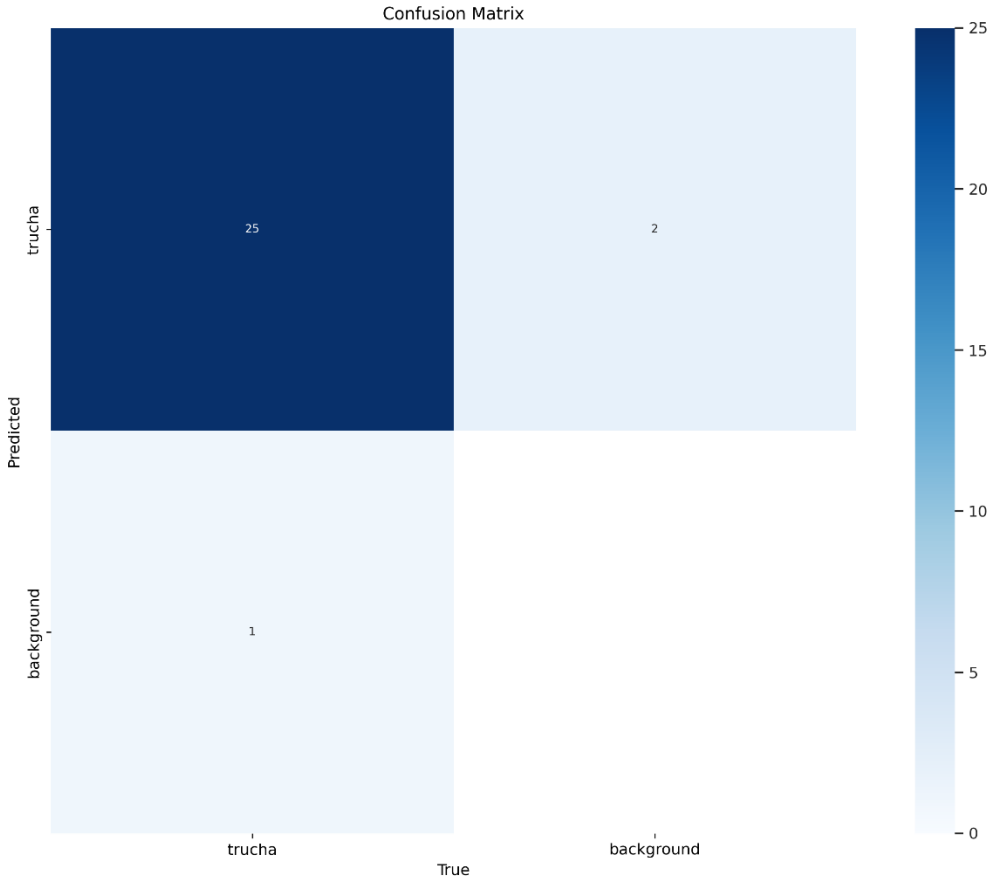


Figura 20. Matriz de confusión

La matriz de confusión ilustra el proceso de validación del sistema. En el eje horizontal se encuentra la respuesta correcta (*ground truth*) y en el eje vertical, la predicción realizada por el modelo. A partir de los resultados obtenidos, se puede observar que, de las 26 imágenes de validación que contienen una trucha en cada una, 25 truchas fueron detectadas correctamente (TP). Sin embargo, el modelo también realizó 2 detecciones incorrectas al identificar truchas donde no las había (FP), y falló en detectar una trucha presente en una imagen (FN).

Esto se traduce en una precisión del 92,59% y una exactitud del 89,29%.

5.1.2 Rendimiento del modelo

A continuación, en la Figura 21, se presentan las gráficas obtenidas durante el entrenamiento del modelo. Estas gráficas proporcionan una representación visual del rendimiento del modelo a lo largo del tiempo, mostrando la evolución de los valores clave a medida que el modelo se ha entrenado durante las 250 épocas.

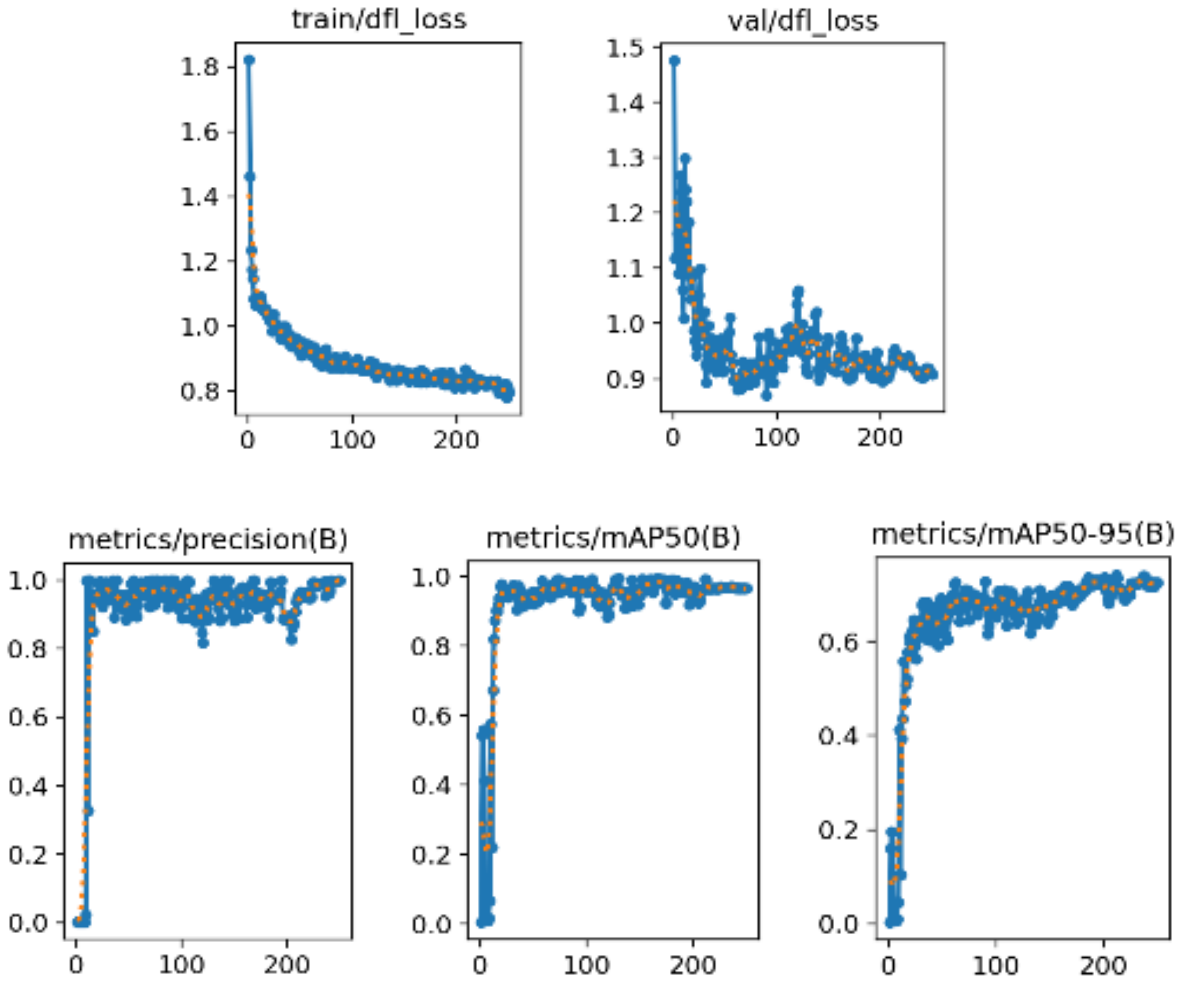


Figura 21. Gráficas obtenidas del entrenamiento del modelo

A lo largo del avance de las épocas, se puede observar una disminución en las pérdidas del modelo. Las dls, descienden en el conjunto del entrenamiento hasta estabilizarse alrededor de 0,8, mientras que en el conjunto de validación se sitúan aproximadamente en 0,9. Contrariamente, la precisión y el mAP50 aumentan con el avance de las épocas. Tanto la precisión como el mAP50 se estabilizan en valores muy cercanos a uno, indicando un alto rendimiento del modelo.

El valor de mAP50-95 también muestra una progresión ascendente a lo largo de las épocas, alcanzando un valor superior a 0,65. Sin embargo, este valor puede ser bajo debido a la imprecisión en el etiquetado manual de los datos, un proceso empleado en este proyecto. Esta tendencia sugiere que, aunque el modelo ha mejorado significativamente, aún hay

margen para optimizar la calidad del etiquetado para alcanzar una mayor exactitud en las predicciones.

5.2 Resultados obtenidos

Para analizar los distintos resultados y corroborar su eficacia se cuenta con el grabado y etiquetado manual que se llevó a cabo por los técnicos Juan Enrique Barrios Sánchez (Barrios Sánchez, 2023) y Almudena Gallego Fernández, bajo la supervisión de los investigadores Álvaro De la Llave-Propín y Morris Villarroel Robinson, [28].

Para ello, el sistema genera un archivo Excel como parte de la salida. Este archivo proporciona información similar al etiquetado manual disponible, detallando el *frame* en el que la trucha cambia de sección y especificando la nueva sección a la que se traslada. De este modo, cada vez que la trucha realiza un cambio de sección, dicha información queda registrada en el archivo Excel, facilitando un seguimiento detallado y preciso de los movimientos de la trucha a lo largo del vídeo.

Posteriormente, se ha implementado un sistema que compara ambos archivos y determina en cuántos *frames* coinciden. Esta comparación se realiza dentro del rango cubierto por el archivo con etiquetado manual, ya que dicho archivo no analiza todo el vídeo de 5 minutos y medio de duración, sino únicamente 5 minutos. Este enfoque permite evaluar la precisión del sistema automatizado en relación con el etiquetado manual, proporcionando una medida clara de su eficacia en el período analizado. Para ello se emplea la ecuación 7, obteniendo así el porcentaje de coincidencia entre ambos sistemas.

$$Eficacia\ [\%] = \frac{nº\ frames\ coincidentes}{nº\ frames\ totales} \cdot 100 \tag{7}$$

En el anexo se incluye una tabla que muestra todos los porcentajes obtenidos para cada uno de los 30 clips analizados, utilizando cada uno de los sistemas propuestos.

5.2.1 Seguimiento con centroide

Con esta propuesta se han obtenido buenos resultados, ya que en 18 de los 30 clips se alcanzó un valor de eficacia superior al 90%, comparando los valores del etiquetado manual y los obtenidos con el sistema automático. En dos de estos clips, el valor de eficacia fue del 100%. Esto se debe a que, en esos clips, la trucha no realizó ningún cambio de sección, eliminando la posibilidad de error en la predicción del *frame* exacto donde ocurren dichos cambios. Como resultado, la trucha fue detectada correctamente en la misma sección a lo largo de todo el vídeo, asegurando una detección precisa y constante.

Por otro lado, solo en dos vídeos se obtuvo un valor de eficacia inferior al 80%. El vídeo con el valor más bajo registró un 74,85%. En la Figura 22, se presenta una gráfica que muestra los valores obtenidos por el sistema automático en comparación con los valores del sistema manual, proporcionando una visualización clara de las diferencias y semejanzas.

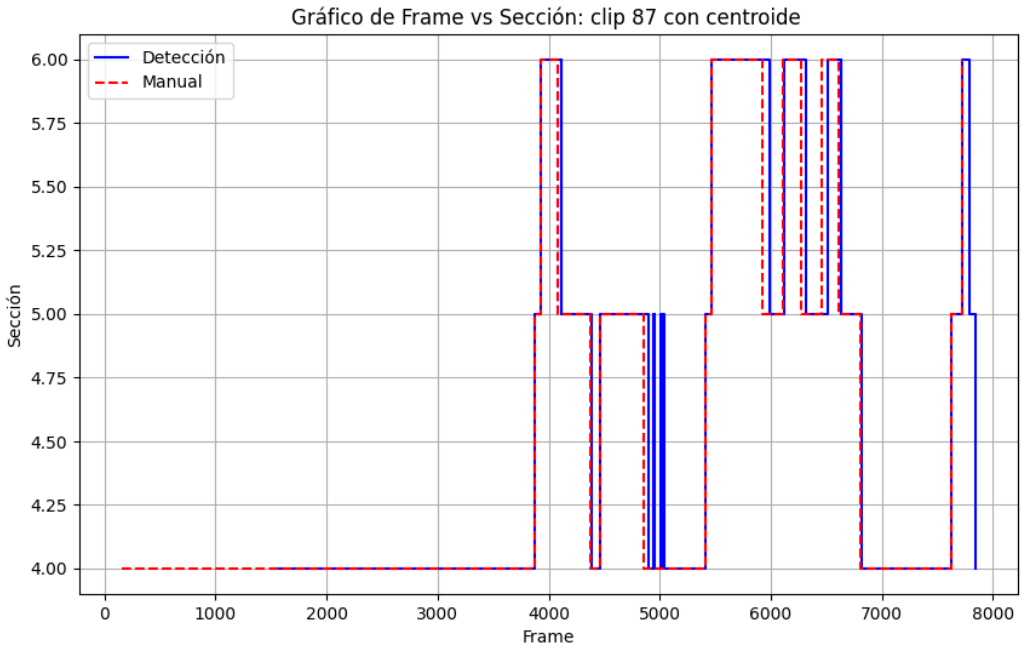


Figura 22. Comparativa del clip 87 con centroide

En la gráfica, se puede observar que el mayor error ocurre alrededor del *frame* 5000. Para investigar la causa de este error, se ha visualizado el vídeo obtenido en la solución. Se observa que la trucha se encuentra sobre la línea divisoria entre las secciones 4 y 5. Debido a esto, el etiquetado manual consideró que la trucha se encontraba en la sección 4, mientras que la detección automática fluctúa entre las secciones 4 y 5. Esta discrepancia explica el error observado en esa parte del vídeo.

Por otro lado, la comparativa del segundo dato con menor porcentaje de acierto se puede observar en la Figura 23. Pertenece al clip 78.

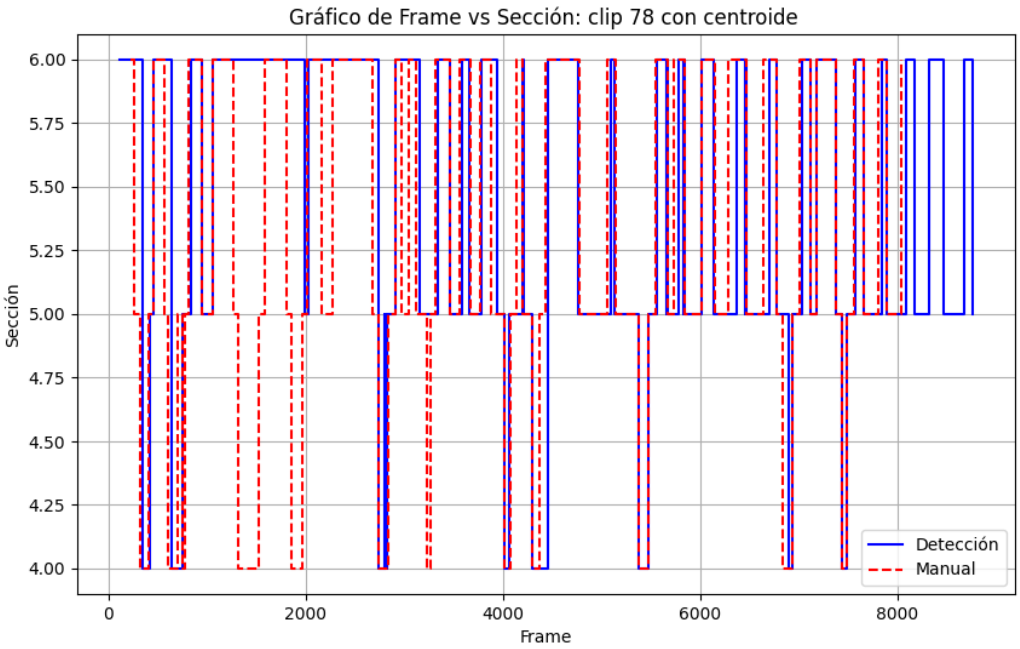


Figura 23. Comparativa del clip 78 con centroide

Lo que se puede ver en la gráfica es que, en algunas ocasiones, el sistema no detecta correctamente el cambio de sección de la trucha. Analizando el vídeo de salida, se observa que, en este caso, el agua de la pecera está muy turbia. Esta turbidez, junto con el fondo de color oscuro, dificulta que el sistema detecte la trucha de manera precisa, especialmente en la sección 4. Esta situación explica el menor porcentaje de acierto en ese clip. A pesar de estos errores el acierto del sistema para este clip es de 75,43%.

Sin embargo, como ya se ha mencionado, de todos los clips, 18 han obtenido un porcentaje de acierto muy alto. Por ejemplo, en la Figura 24 se observa cómo la detección automática y el etiquetado manual coinciden en la mayor parte de los *frames*, cometiendo solo errores mínimos en los cambios de sección. Esta alta coincidencia demuestra la eficacia del sistema automático en la mayoría de los casos.

Figura 24. Comparativa del clip 71 con centroide

Teniendo en cuenta todos los resultados comentados y analizando los datos de los 30 clips, se obtiene un promedio de acierto del 90,38%. Esto indica que la solución proporcionada por el sistema es bastante fiable en general, reflejando una capacidad consistente para detectar y seguir correctamente las secciones de las truchas en los vídeos analizados.

5.2.2 Seguimiento con la cabeza como referencia

En esta propuesta, se ha decidido determinar la sección donde se encuentra la trucha basándose en la posición de su cabeza, la cual se sitúa dependiendo de la dirección de movimiento que sigue.

En este caso, se ha observado que 5 de los 30 clips han alcanzado un porcentaje de acierto superior al 90%. De estos, dos clips han obtenido un valor del 100%. Al igual que en el caso anterior, estos dos clips corresponden a situaciones donde la trucha no ha cambiado de sección a lo largo del vídeo.

Sin embargo, en 8 de los 30 clips se ha obtenido un valor de acierto inferior al 80%, siendo el valor más bajo de 71,28% correspondiente al clip 87. En la Figura 25 se muestra una comparativa entre los resultados de detección automática y el etiquetado manual para este clip en particular.

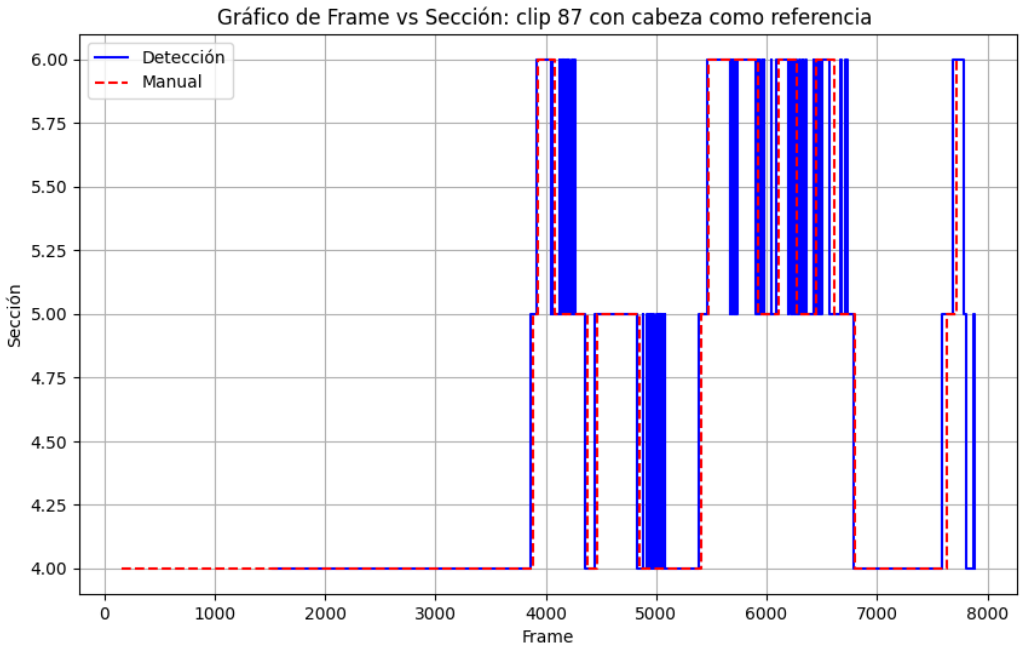


Figura 25. Comparativa del clip 87 con la cabeza como referencia

Lo más destacable de esta gráfica es el cambio continuo entre secciones en el sistema de detección automático. Para investigar la causa, se visualiza el vídeo de salida correspondiente. Se observa que el parpadeo ocurre cuando la trucha está quieta. La detección de la cabeza se realiza comparando su posición con la de los 7 *frames* anteriores, en estos momentos de quietud, la trucha muestra un ligero movimiento lateral, haciendo que la posición de la cabeza varíe entre la cabeza y la cola. Tras analizar este comportamiento, se identifica que el mayor error de esta propuesta radica en la detección de la cabeza cuando la trucha no está en movimiento, lo cual puede llevar a errores de localización de la propia cabeza. En el caso específico de este clip, la trucha se detiene justo en la línea entre dos secciones, lo que provoca el parpadeo y la detección alternativa entre ambas secciones.

A pesar de este tipo de errores, se pueden observar buenos resultados, como se muestra en la Figura 26, donde a pesar de pequeñas discrepancias se observa una gran coincidencia entre la detección automática y el etiquetado manual. Tras analizar los 30 clips, se obtiene un promedio del 84,16% en la detección automática basada en la posición de la cabeza de la trucha.

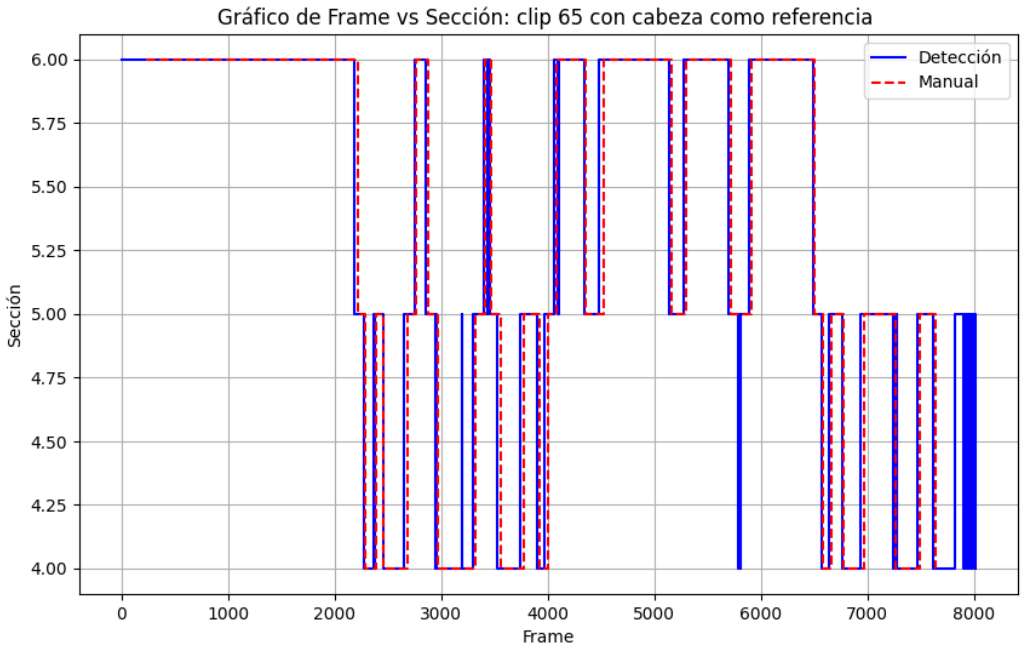


Figura 26. Comparativa del clip 65 con la cabeza como referencia

5.2.3 Cambio de parámetros en la detección

Con el objetivo de aumentar la velocidad de procesamiento del vídeo, se modifica el parámetro imgsz, que controla el tamaño de la imagen utilizado en la detección. Al reducir el tamaño de la imagen, se logra acelerar tanto la velocidad de procesamiento de la detección como del sistema completo. Por ejemplo, un vídeo con imgsz=640 tiene un tiempo medio de procesamiento completo de unos 37 minutos. Sin embargo, al reducir el tamaño de la imagen a la mitad, es decir imgsz=320, el tiempo de procesamiento total se reduce significativamente a una media de 25 minutos para un vídeo de la misma duración, unos 5 minutos y medio. Este cambio de parámetro se ha aplicado en el sistema de detección con centroide.

Con este cambio en los parámetros de detección, se ha obtenido un porcentaje de acierto superior al 90% en 17 de los 30 clips, y dos de esos clips han alcanzado el 100% de acierto, debido a que no hubo cambios de sección a lo largo del vídeo. Por otro lado, solo 5 clips presentaron un valor de acierto inferior al 80%.

Para analizar los posibles errores, se observará la Figura 27, la cual muestra una comparación entre la detección automática y el etiquetado manual con una coincidencia del 74,86%.

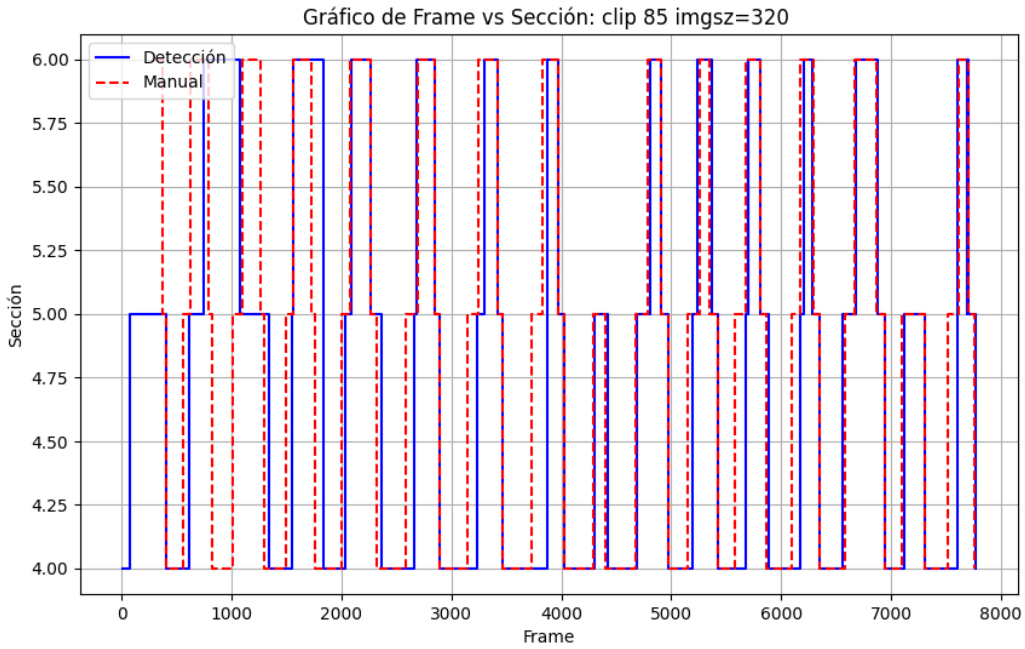


Figura 27. Comparativa del clip 85 con imgsz=320

En la comparación, se puede observar que la detección automática en varias ocasiones detecta el cambio de sección con retraso. Esto se debe a que al reducir el imgsz de la detección, también se reduce la eficacia de esta. Como resultado, la trucha puede cambiar de sección, pero no ser detectada hasta unos *frames* más tarde.

Aun con este problema de detección, el porcentaje de acierto medio de los 30 clips es del 88,29%. Esto indica que, a pesar de los desafíos asociados con la reducción del tamaño de imagen, el sistema sigue mostrando un rendimiento general sólido y fiable

5.3 Comparativa de resultados

En este punto, se procederá a realizar una comparativa exhaustiva de las tres propuestas desarrolladas para la detección y seguimiento de la trucha: utilizando como punto de referencia el centroide, utilizando la cabeza, y utilizando el centroide con la modificación del imgsz a 320 para la detección. Para llevar a cabo esta comparación, se analizarán tanto los porcentajes de acierto como la efectividad general de las detecciones en cada método.

En primer lugar, se presenta en la Figura 28 una comparación de los porcentajes de efectividad obtenidos por cada uno de los tres métodos a lo largo de los 30 clips proporcionados. Esta figura permite visualizar cómo se comporta cada método en términos de precisión y fiabilidad, y ayuda a determinar cuál de las propuestas ofrece el mejor rendimiento global.

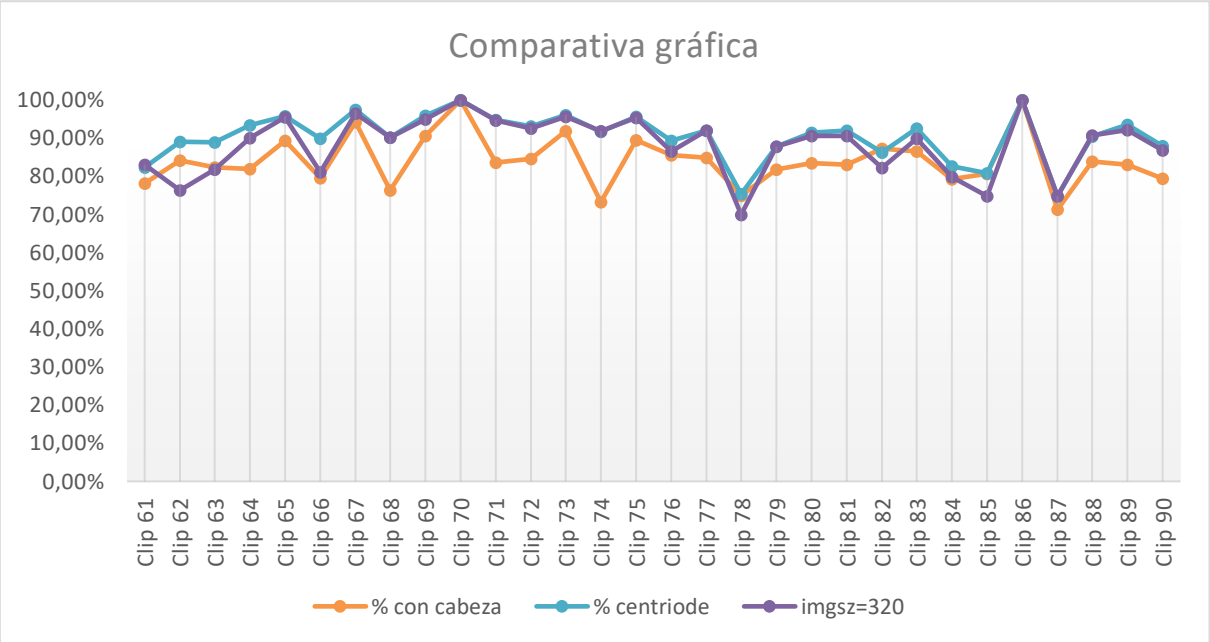


Figura 28. Comparativa gráfica

En la figura, se observa que, en términos generales, tanto el uso del centroide como el caso de imgsz=320 obtienen visiblemente mejores resultados en comparación con el uso de la cabeza como punto de referencia. Además, se puede apreciar que los resultados obtenidos con el centroide y un imgsz mayor son ligeramente superiores en todos los clips.

En la Tabla 1 se puede observar un promedio de los porcentajes de acierto de cada uno de los métodos evaluados. Esta tabla proporciona una visión general de la efectividad de cada enfoque, permitiendo una comparación clara entre ellos. Para un análisis más detallado, la tabla desglosada con los valores individuales de cada clip se encuentra en el anexo.

Tabla 1. Eficacia promedio de las propuestas

	Centroide	**Cabeza**	**Imgsz=320**
Promedio	90,38%	84,16%	88,29%

Observando los valores promedio de la Tabla 1, se puede concluir que el método con el que se obtienen los peores resultados es el que utiliza la cabeza como referencia para indicar en qué sección se encuentra la trucha. Por otro lado, el método que ofrece los mejores resultados es el que utiliza el centroide, con un 90,38% de aciertos. La segunda mejor opción, con una diferencia de solo 2,09%, es la de reducir el imgsz a 320. Esta opción también ofrece resultados sólidos, aunque ligeramente inferiores, destacando su eficiencia en términos de velocidad de procesamiento.

Con el objetivo de entender mejor las diferencias entre las distintas opciones, se realiza una comparación cualitativa entre ellas. Para observar la diferencia entre el uso del centroide y el uso de la cabeza como referencia, se puede visualizar la Figura 29. Esta figura proporcionará

una representación clara de cómo cada método afecta la precisión en la identificación de la sección en la que se encuentra la trucha.

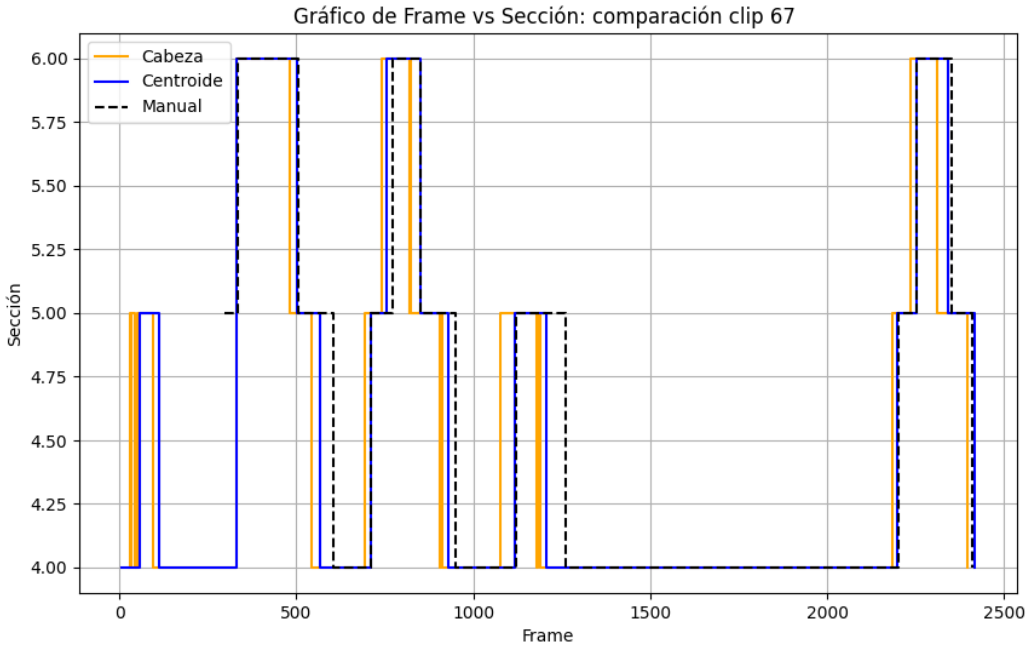


Figura 29. Comparación clip 67

Para ello, la figura presenta el contenido de los archivos Excel correspondientes a los distintos métodos para un vídeo. Los datos se representan en naranja para el uso de la cabeza, en azul para el uso del centroide y en negro para los resultados del etiquetado manual, que contiene la solución correcta.

Previo a los 1500 *frames*, se puede observar cómo ambas detecciones localizan el cambio de sección con cierta anticipación, siendo más destacado el adelantamiento en la detección basada en la cabeza. Sin embargo, en los últimos cambios de sección, se ve cómo la detección utilizando el centroide es prácticamente simultánea al etiquetado manual.

El adelanto observado se debe a la diferencia en el punto de referencia utilizado para identificar el cambio de sección. La detección que emplea la cabeza como referencia detecta el cambio primero, debido a la posición adelantada de la cabeza en relación con el cuerpo de la trucha. Posteriormente, el centroide de la *bounding box* detecta el cambio de sección. Por ello, el método que utiliza el centroide como referencia ofrece un resultado más preciso en comparación con el etiquetado manual.

Por último, para realizar una comparación cualitativa en el uso del centroide como referencia, pero variando el imgsz entre 640 y 320, se puede observar la Figura 30.

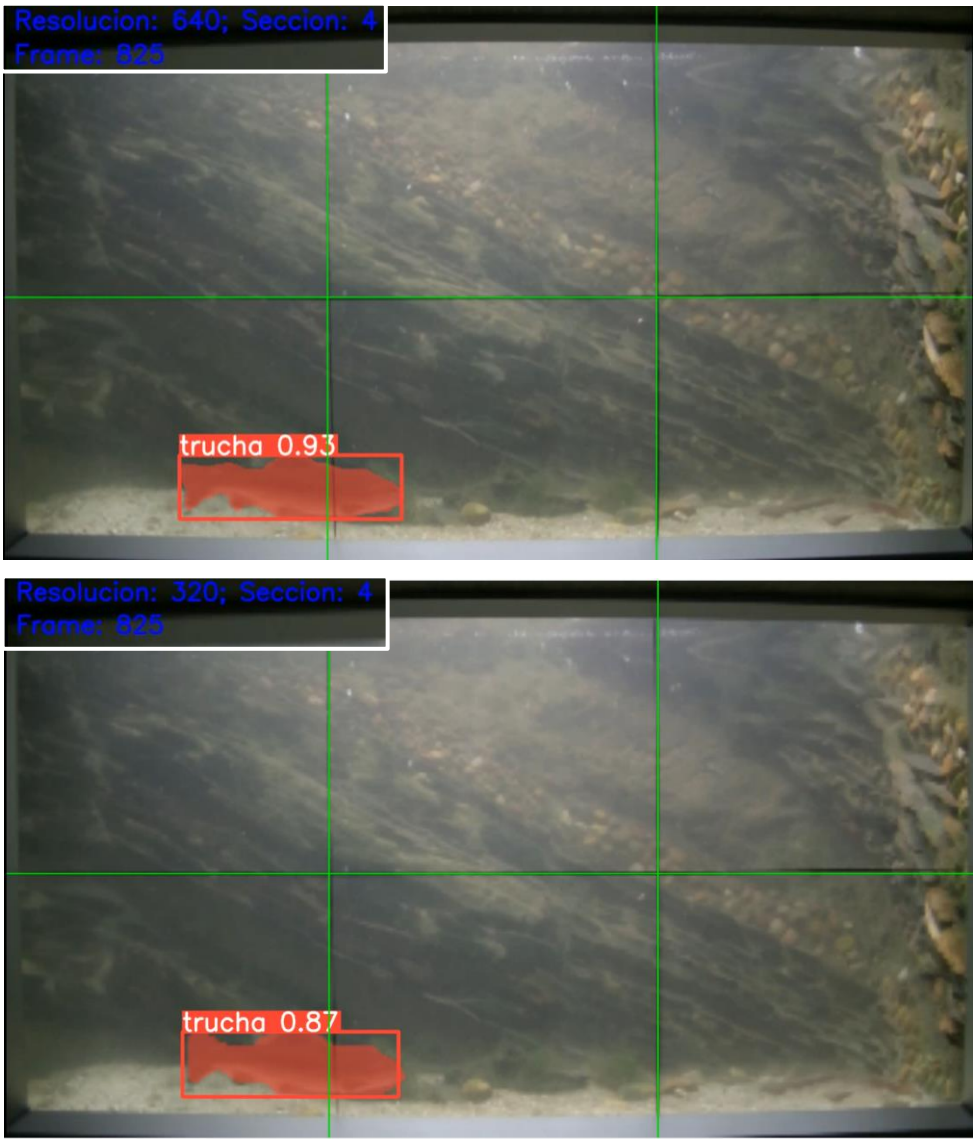


Figura 30. Comparación imgsz=640 e imgsz=320

En la figura, se observa que la imagen superior muestra la detección utilizando un valor de imgsz=640, mientras que la imagen inferior representa la detección con imgsz=320. Ambas imágenes corresponden al *frame* 825 del clip 76.

Un dato notable a simple vista es que la confianza obtenida es mayor en la detección con imgsz=640, con un valor de 0,93, mientras que en la detección con imgsz=320 es de 0,87. Esta diferencia en la confianza significa que en ciertas ocasiones la detección con imgsz=640 supera el umbral de confianza establecido (0,65), mientras que la detección con imgsz=320 no alcanza dicho umbral y, por lo tanto, no muestra esa detección.

El umbral de detección se ha fijado en 0,65 por cuestiones de eficacia, lo que indica que las detecciones con una confianza igual o superior a este valor son consideradas como válidas y mostradas en los resultados.

Otro hecho notable es la pérdida de calidad en la imagen, especialmente evidente al observar las burbujas en la superficie de la pecera, aunque esto se nota más en vídeos donde el agua está más limpia. Sin embargo, este aspecto no afecta al estudio del comportamiento de la trucha.

Después de analizar todos los resultados obtenidos, se concluye que el mejor método para lograr precisión es aquel que utiliza el centroide con un valor de imgsz de 640. En cambio, si se prioriza la velocidad, reduciendo el procesamiento del vídeo 12 minutos a costa de un 2,09% en eficacia, la mejor opción es emplear el centroide con un imgsz de 320. Por lo tanto, el sistema final ofrece la flexibilidad de ajustar este valor según las necesidades específicas de cada situación.

6. Presupuesto

En este apartado se detalla el presupuesto orientativo requerido para la realización del desarrollo del proyecto. En el presupuesto se incluyen tanto recursos materiales como la mano de obra.

El proyecto completo se ha dividido en dos fases: investigación y desarrollo. La fase de investigación ha constado de 70 horas y la fase de desarrollo ha requerido 240 horas.

Tabla 2. Presupuesto general

Activo	Precio unitario (€)	Unidades	Precio total (€)
Personal de investigación	12,05/h	70	843,50
Personal de desarrollo	12,05/h	240	2892,00
Ordenador	829,00	1	829,00
Paquete Office	0,00	1	0,00
Entorno de trabajo Software	0,00	1	0,00
Presupuesto total			**4564,50**

Personal de investigación y desarrollo: 3735,50€ (310 horas totales a 12,05€/h, salario medio de un ingeniero de telecomunicaciones recién graduado en España) [34]

Ordenador personal: Lenovo Ideapad Flex 5: 829,00€ [35]

Paquete office: licencia estudiante UPM: 0€

Entorno de trabajo Software: editores de código (*Google Colab, Visual Studio Code*) y librerías empleadas: 0€ (Gratuito)

El presupuesto total del proyecto ha sido CUATRO MIL QUINIENTOS SESENTA Y CUATRO CON CINCUENTA euros.

7. Impacto del proyecto[36]

7.1 Implicaciones del proyecto

El desarrollo del proyecto tiene repercusión en diferentes áreas, incluyendo la economía, salud y seguridad, tecnología e industria.

Con los datos obtenidos en este proyecto se pueden realizar estudios sobre el comportamiento de las truchas, lo que permitirá mejorar las condiciones de cría. Esto, a su vez, incrementará la productividad mejorando la rentabilidad.

El estudio del comportamiento de las truchas permite identificar problemas de salud y de calidad del agua en cautiverio, por lo que se puede mejorar la vida de los peces garantizando la seguridad para el consumo humano.

Por otro lado, la implementación de un sistema de seguimiento de truchas tiene un impacto significativo en la investigación de sistemas acuáticos en cautividad, así como en la industria acuícola, debido al incremento de producción y reducción de desperdicios.

7.2 Contribución a los Objetivos de Desarrollo Sostenible (ODS)[37]

El proyecto contribuye al cumplimiento de algunos de los ODS planteados por las Naciones Unidas en la agenda 2023:

- **ODS 2 – Hambre cero**: Se mejora la productividad y la sostenibilidad de la acuicultura, contribuyendo a la seguridad alimentaria.
- **ODS 8 – Trabajo decente y crecimiento económico**: Este proyecto puede generar diversas oportunidades laborales, tanto en desarrollo y mejora del sistema como su mantenimiento. Además, al incrementar la eficacia en la producción de truchas se favorece el crecimiento económico.
- **ODS 9 – Industria, innovación e infraestructuras**: Se promueve la innovación tecnológica en la acuicultura y se desarrollan infraestructuras sostenibles para mejorar la eficiencia en la producción.
- **ODS 12 – Producción y consumo responsables**: El sistema ayuda a reducir los desperdicios en la producción de truchas, promoviendo un consumo y una producción más responsables.
- **ODS 14 – Vida submarina**: La investigación sobre el comportamiento de las truchas impulsa una gestión responsable en la cría de especies acuáticas en cautiverio. También fomenta la acuicultura como alternativa a la pesca excesiva en océanos.

8. Conclusiones

Se ha desarrollado un sistema automático de segmentación y seguimiento de truchas en vídeos, con el objetivo de facilitar estudios sobre el comportamiento de estos peces en acuicultura. El problema propuesto consistía en crear un sistema capaz de detectar y seguir truchas en un entorno controlado, superando desafíos relacionados con la segmentación precisa y la localización de la trucha en diferentes secciones del entorno.

La solución planteada implica el uso de inteligencia artificial, concretamente redes neuronales convolucionales (CNN), para segmentar imágenes y localizar la trucha. El sistema ha sido codificado en Python y se ha diseñado para procesar vídeos, generando como salida un vídeo que resalta la trucha con un recuadro, además de un archivo que detalla en qué sección del entorno se encuentra la trucha en cada momento.

Los resultados obtenidos han sido satisfactorios. El método que utiliza el centroide con un tamaño de imagen (imgsz) de 640 ha demostrado ser el más preciso, con una coincidencia con los resultados etiquetados a mano del 90,38%. Sin embargo, para aplicaciones que requieren un procesamiento más rápido, el uso de un imgsz de 320 también ha mostrado ser viable, aunque con una ligera reducción en la precisión (2,09% menos). Estos resultados validan la efectividad del sistema para el seguimiento de truchas, proporcionando una herramienta robusta para estudios de comportamiento.

A partir de toda la información expuesta en el informe, se puede derivar que el sistema es válido para el propósito propuesto. Sin embargo, se han identificado nuevos aspectos del problema, como la sensibilidad de la precisión del sistema a la calidad del vídeo y el tamaño de la imagen procesada. Estos aspectos abren la puerta a mejoras futuras y optimizaciones adicionales.

8.1 Trabajos futuros

A pesar de los avances logrados, el proyecto no deja totalmente resuelto el problema tecnológico abordado. Existen varias líneas futuras de trabajo que podrían mejorar y ampliar las capacidades del sistema:

- **Optimización de la precisión en ambientes diversos**: Adaptar y probar el sistema en diferentes condiciones de iluminación y claridad del agua para mejorar su robustez y aplicabilidad en diversos escenarios de acuicultura.
- **Reducción del tiempo de procesamiento**: Explorar técnicas de compresión y optimización de algoritmos para reducir aún más el tiempo de procesamiento sin sacrificar precisión, permitiendo así un análisis más rápido y eficiente.
- **Integración de nuevas métricas de seguimiento**: Incorporar métricas adicionales que puedan ofrecer una visión más completa del comportamiento de las truchas, como la velocidad de nado y la interacción con otros peces o elementos del entorno.

- **Mejora de la detección de giros y ocultaciones**: Desarrollar métodos avanzados para mejorar la detección de truchas durante giros bruscos u ocultaciones temporales, utilizando técnicas de interpolación y análisis temporal más sofisticadas.
- **Ampliación del conjunto de datos**: Crear un conjunto de datos más extenso y variado para entrenar el modelo, mejorando así su capacidad de generalización y rendimiento en situaciones no vistas previamente.
- **Implementación en tiempo real**: Investigar la posibilidad de implementar el sistema para el seguimiento en tiempo real, lo cual podría ser extremadamente útil para aplicaciones en vivo en entornos de acuicultura.

Estas líneas de trabajo futuras no solo mejorarían la precisión y eficiencia del sistema, sino que también ampliarían su aplicabilidad y utilidad en estudios de comportamiento de truchas y potencialmente otros tipos de peces.

9. Referencias

[1] «Grado en ingeniería de sonido e imagen en Madrid», Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación. Accedido: 30 de junio de 2024. [En línea]. Disponible en: https://www.etsist.upm.es/estudios-ingenieria-sistemas-telecomunicaciones-madrid/grados-ingenieria-Madrid/Sonido_Imagen#outcomes

[2] Á.-J. Rico-Díaz, «Metodología de desarrollo de sistemas de detección y seguimiento de peces mediante tecnología láser y visión artificial con Inteligencia Artificial», 2022, Accedido: 30 de junio de 2024. [En línea]. Disponible en: https://ruc.udc.es/dspace/handle/2183/32805

[3] «Home», OpenCV. Accedido: 30 de junio de 2024. [En línea]. Disponible en: https://opencv.org/

[4] «Librerías de Visión Artificial | SEAS | Blog SEAS». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.seas.es/blog/seas-cursos-online/librerias-de-vision-artificial/

[5] «¿Qué es la inteligencia artificial? - Explicación de la inteligencia artificial (IA) - AWS», Amazon Web Services, Inc. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/artificial-intelligence/

[6] «Aprendizaje automático: Qué es y por qué importa». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.sas.com/es_es/insights/analytics/machine-learning.html

[7] J. Díaz-Ramírez, «Aprendizaje Automático y Aprendizaje Profundo», *Ingeniare Rev. Chil. Ing.*, vol. 29, n.º 2, pp. 180-181, jun. 2021, doi: 10.4067/S0718-33052021000200180.

[8] «¿Qué es el machine learning? - Explicación sobre el machine learning empresarial - AWS», Amazon Web Services, Inc. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/machine-learning/

[9] «¿Qué es el aprendizaje profundo? - Explicación del aprendizaje profundo - AWS», Amazon Web Services, Inc. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/deep-learning/

[10] «Redes neuronales: definición e importancia». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.sas.com/es_es/insights/analytics/neural-networks.html

[11] «¿Qué es una red neuronal? | IBM». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/neural-networks

[12] Datademia, «¿Qué es Deep Learning y qué es una red neuronal?», Datademia. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://datademia.es/blog/que-es-deep-learning-y-que-es-una-red-neuronal

[13] «Activation Functions in Neural Networks [12 Types & Use Cases]». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.v7labs.com/blog/neural-networks-activation-functions

[14] «Redes neuronales - Ciencia sin seso... locura doble». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.cienciasinseso.com/redes-neuronales/

[15] «What are Convolutional Neural Networks? | IBM». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.ibm.com/topics/convolutional-neural-networks

[16] D. Calvo, «Red Neuronal Convolucional CNN», Diego Calvo. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.diegocalvo.es/red-neuronal-convolucional/

[17] R. Gandhi, «R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms», Medium. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[18] 330ohms, «¿Cómo detectar objetos con YOLO?», 330ohms. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://blog.330ohms.com/2020/11/17/deteccion-de-objetos-con-yolo/

[19] «keras_mask_rcnn_couch.jpg (424×600)». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://pyimagesearch.com/wp-content/uploads/2019/06/keras_mask_rcnn_couch.jpg

[20] «Albumentations Documentation - Bounding boxes augmentation for object detection». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/

[21] «Albumentations Documentation - Mask augmentation for segmentation». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://albumentations.ai/docs/getting_started/mask_augmentation/

[22] «NumPy -». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://numpy.org/

[23] Ultralytics, «YOLOv8». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://docs.ultralytics.com/models/yolov8

[24] «pandas - Python Data Analysis Library». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://pandas.pydata.org/

[25] «tkinter — Python interface to Tcl/Tk», Python documentation. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://docs.python.org/3/library/tkinter.html

[26] «Google Colab». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://colab.research.google.com/

[27] «Visual Studio Code - Code Editing. Redefined». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://code.visualstudio.com/

[28] J. E. Barrios Sánchez, «La prueba de la red: evaluando su repetibilidad y el efecto del ayuno en la trucha arcoiris (Oncorhynchus mykiss)». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://oa.upm.es/77142/

[29] K. Wada, «Labelme: Image Polygonal Annotation with Python». 1 de julio de 2024. doi: 10.5281/zenodo.5711226.

[30] Ultralytics, «Configuración». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://docs.ultralytics.com/es/usage/cfg

[31] J. M. Sánchez Muñoz, «Análisis de Calidad Cartográfica mediante el estudio de la Matriz de Confusión», *Pensam. Matemático*, vol. 6, n.º 2, pp. 9-26, 2016.

[32] R. Borja-Robalino, A. Monleon-Getino, y J. Benedé, «Estandarización de Métricas de Rendimiento para Clasificadores Machine y Deep Learning», feb. 2020.

[33] Q. M. Rahman, N. Sünderhauf, y F. Dayoub, «Per-frame mAP Prediction for Continuous Performance Monitoring of Object Detection During Deployment». arXiv, 16 de noviembre de 2020. doi: 10.48550/arXiv.2009.08650.

[34] «Salario para Ingeniero Telecomunicaciones en España - Salario Medio», Talent.com. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://es.talent.com/salary

[35] «Lenovo IdeaPad Flex 5 14IAU7 Intel Core i7-1255U/16GB/512GB SSD/14" Táctil | PcComponentes.com». Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.pccomponentes.com/lenovo-ideapad-flex-5-14iau7-intel-core-i7-1255u-16gb-512gb-ssd-14-tactil?s_kwcid=AL!14405!3!!!!x!!&gad_source=1&gclid=CjwKCAjwmYCzBhA6EiwAxFwfgGM9w_gRSszctNXaIVGNFmmy-RyU8pR0gNivYQlDx5Kz0jXeH6485RoCyzoQAvD_BwE

[36] Fernández Aller, Celia; Miñano, Rafael. “Guía para trabajar la Responsabilidad Social y Ambiental (GRSA)". ETSI Sistemas Informáticos, UPM. (Versión 2.0 Mayo 2015). [En línea]. Disponible en: https://oa.upm.es/35542/1/Guia_Responsabilidad_Social_y_Ambiental-V2-1.pdf

[37] M. J. Gamez, «Objetivos y metas de desarrollo sostenible», Desarrollo Sostenible. Accedido: 1 de julio de 2024. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

Anexo

A.1 Resultados obtenidos

En la Tabla 3 se muestran los porcentajes obtenidos con cada uno de los sistemas propuestos para los 30 clips.

Tabla 3. Porcentajes de coincidencia para cada clip

	Centroide	Cabeza	Imgsz=320
Clip 61	82,35%	78,20%	83,04%
Clip 62	89,08%	84,16%	76,33%
Clip 63	88,94%	82,30%	81,85%
Clip 64	93,37%	81,89%	90,10%
Clip 65	95,73%	89,37%	95,46%
Clip 66	89,87%	79,58%	81,11%
Clip 67	97,43%	94,29%	96,47%
Clip 68	90,14%	76,38%	90,17%
Clip 69	95,97%	90,58%	94,95%
Clip 70	100,00%	100,00%	100,00%
Clip 71	94,86%	83,64%	94,69%
Clip 72	93,08%	84,66%	92,53%
Clip 73	96,12%	91,90%	95,60%
Clip 74	91,67%	73,24%	91,90%
Clip 75	95,67%	89,48%	95,32%
Clip 76	89,33%	85,57%	86,53%
Clip 77	92,03%	84,90%	91,95%
Clip 78	75,43%	74,89%	69,98%
Clip 79	87,81%	81,78%	87,81%
Clip 80	91,49%	83,48%	90,67%
Clip 81	91,95%	83,02%	90,68%
Clip 82	86,28%	87,27%	82,26%
Clip 83	92,52%	86,60%	89,92%
Clip 84	82,58%	79,32%	79,83%
Clip 85	80,82%	80,71%	74,86%
Clip 86	100,00%	100,00%	100,00%
Clip 87	74,85%	71,28%	74,80%
Clip 88	90,41%	83,88%	90,77%
Clip 89	93,55%	83,01%	92,15%
Clip 90	87,92%	79,37%	86,86%
Media	**90,38%**	**84,16%**	**88,29%**

Manual de usuario

En este apartado se realiza una explicación sobre el funcionamiento del sistema y los posibles errores que pueden aparecer.

A.2 Funcionamiento del sistema

En primer lugar, se debe comprobar que el modelo entrenado de YOLO se encuentra en la dirección correcta, en caso contrario se debe introducir en el código como se muestra en la Figura 31.

```
cap = cv2.VideoCapture(video_path)
variables = initialize_var(cap)
model = YOLO('C:/Users/ruthb/Desktop/Ruth/Universidad/PFG/Modelos/resultados11/weights/best.pt')
```

Figura 31. Path del modelo de YOLO

En segundo lugar, hay que cerciorarse de que se dispone de un entorno de ejecución de Python 3 y que se contienen todas las librerías necesarias, las cuales se muestran en la Figura 32. Tras esta comprobación ya se puede ejecutar el código como se ve en la figura.

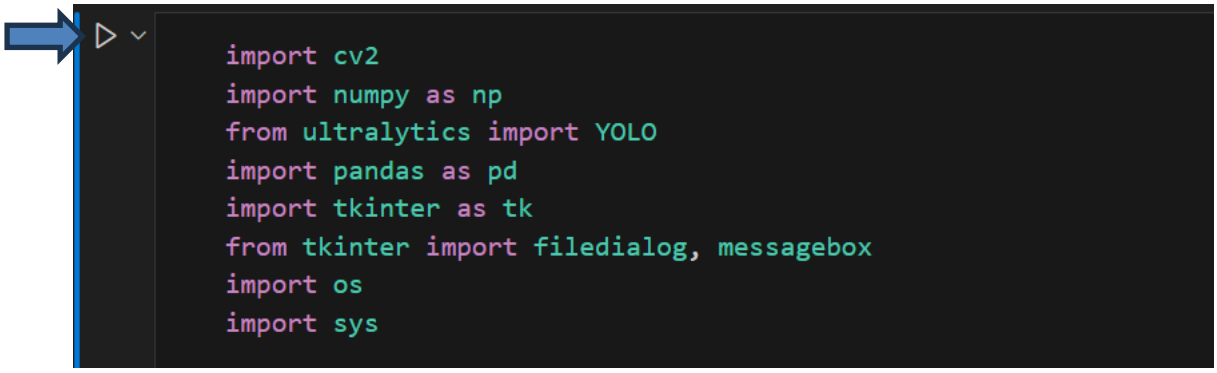


Figura 32. Entorno de ejecución

Una vez comienza la ejecución del código se abre una ventana con la interfaz de la Figura 33.

En la interfaz se observan diferentes opciones:

① Pulsando esta opción se abre una ventana como la que se observa en la imagen de la izquierda de la Figura 34 en la cual se debe seleccionar el vídeo que se desea procesar. Esta ruta no debe contener espacios.

② Pulsando esta opción se abre una ventana como la que se observa en la imagen de la derecha de la Figura 34 en la cual se debe seleccionar la carpeta en la que se almacenan los archivos de salida. Esta ruta no debe contener espacios.

③ En este punto se debe de hacer una selección dependiendo del imgsz que se desee: 640 en el caso de querer asegurar un porcentaje mayor de acierto o 320 si por el contrario se prefiere una ejecución más rápida del sistema.

④ Una vez habiendo seleccionado todos los campos necesarios se debe pulsar este botón para comenzar la ejecución. Al pulsar este botón aparecerá una ventana como la de la Figura 35, en la cual se debe confirmar que los datos introducidos son correctos.

Figura 33. Interfaz del programa

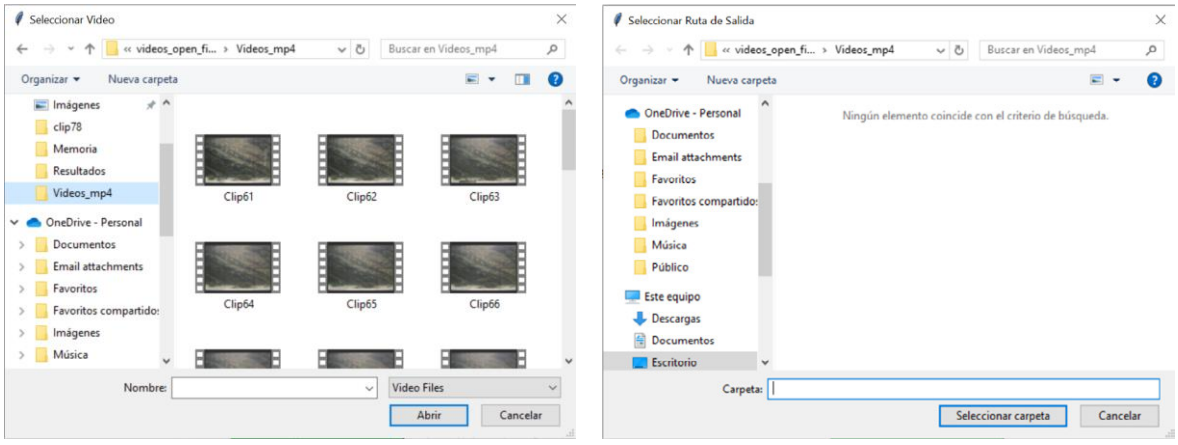


Figura 34. Selección de las rutas

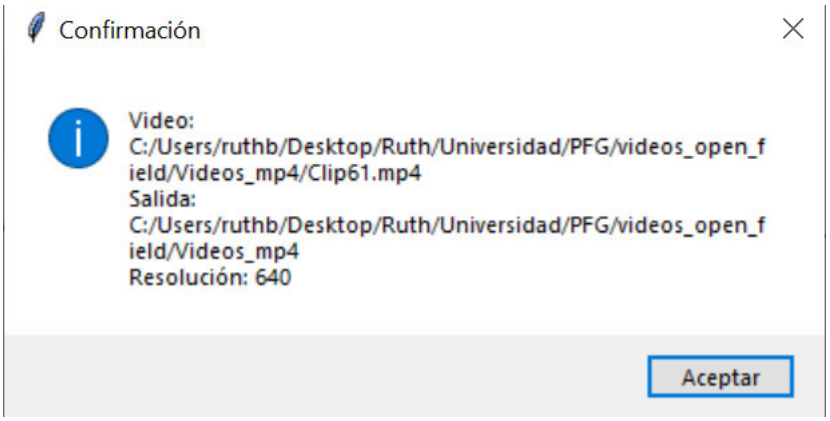
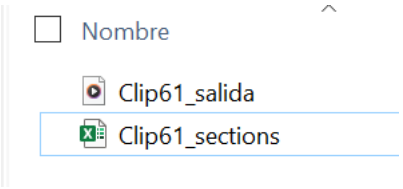


Figura 35. Confirmación de los datos seleccionados

Tras confirmar los datos introducidos comenzará la ejecución del sistema de detección. Durante este proceso se observará la Figura 36, la cual muestra los *frames* en ejecución. En la esquina superior izquierda se puede observar la resolución seleccionada, la sección en la cual se encuentra la trucha y el *frame* que está analizando en cada momento. Si estando en esta pantalla se desea detener la ejecución de debe pulsar la tecla "q".

Figura 36. *Frames* en ejecución

Una vez finalizada la ejecución se podrá consultar los resultados obtenidos en la carpeta seleccionada, como se puede ver en la imagen de la izquierda de la Figura 37. Por un lado, se tiene el vídeo de salida el cual contiene todos los *frames* analizados y con toda la información previamente comentada. Por otro lado, un archivo Excel que contine los cambios de sección que realiza la trucha y el *frame* en el que sucede, se puede ver un ejemplo en la imagen de la derecha de la Figura 37.



	A	B	C
1	Frame	Section	
2	15	4	
3	127	5	
4	182	6	
5	221	5	
6	277	4	
7	497	5	
8	567	4	
9	714	5	
10	796	6	
11	860	5	
12	898	4	
13	1128	6	
14	1326	4	
15	1403	5	
16	1559	4	
17	1670	5	

Figura 37. Salida del sistema

A.3 Posibles errores

Al pulsar en el botón de confirmar pueden aparecer dos errores como los que se observan en la Figura 38. El primero indica que la ruta del vídeo tiene espacios, para solucionarlo se debe comprobar las carpetas afectadas por la ruta del vídeo seleccionado, incluido el propio nombre del vídeo y quitar todos los espacios que haya. El segundo error indica que, o bien no se ha seleccionado el vídeo o no se ha seleccionado la carpeta de salida, para solucionarlo comprobar que se selecciona correctamente estos dos campos.

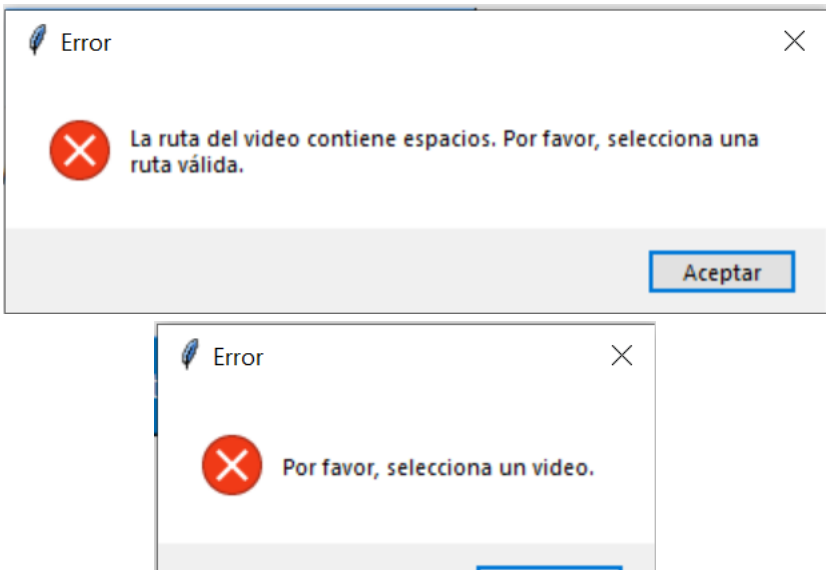


Figura 38. Errores de selección