UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Técnicas avanzadas de Visión por Computador (VC) basadas en *Deep Learning* (DL) aplicadas a la monitorización de especies marinas (*Bluefin Tuna*)

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Joaquín Martínez Peiró

Tutor: Gabriela Andreu García, Pau Muñoz Benavent

Curso 2020-2021

Resumen

Este trabajo fin de grado está orientado a explorar las arquitecturas, técnicas y metodologías de las redes neuronales convolucionales (CNN), aplicadas al procesamiento de imágenes y la caracterización de objetos en ellas. En este proyecto se aplica la técnica conocida como *transfer learning*, técnica que aprovecha redes previamente entrenadas con una gran cantidad de datos para adaptarlas a nuevas tareas con un entrenamiento mucho más rápido. Esto permite acortar de forma considerable el tiempo de desarrollo necesario para una primera aproximación al problema. Se aborda el estudio de los modelos Faster R-CNN, Mask R-CNN, YOLOv5 y PointRend para la detección y segmentación de imágenes subacuáticas de especies marinas, en concreto de la especie *Thunnus thynnus* (atún rojo del Atlántico), con el objetivo de monitorizar automáticamente la biomasa de la población de esta especie.

Palabras clave: aprendizaje por transferencia, aprendizaje profundo, dimensionamiento automático de especies marinas, visión estereoscópica subacuática, visión por computador, redes neuronales convolucionales.

Abstract

This work is aimed at exploring the architectures, techniques and methodologies of convolutional neural networks (CNN), applied to image processing and the characterization of objects in them. In this project, the technique known as transfer learning is applied, a technique that takes advantage of previously trained networks with a large amount of data to adapt them to new tasks with much faster training. This allows to considerably shorten the development time required for a first approach to the problem. The study of the Faster R-CNN, Mask R-CNN, YOLOv5 and PointRend models for the detection and segmentation of underwater images of marine species, specifically of the species *Thunnus thynnus* (Atlantic Bluefin tuna) is addressed, with the aim of automatically monitoring the biomass of the population of this species.

Keywords: automatic fish sizing, computer vision, convolutional neural network, deep learning, transfer learning, underwater stereo vision.

Índice

Índice de figuras

Acrónimos

Ai2	Instituto Universitario de Automática e Informática Industrial
AP	Average Precision
AR	Average Recall
bbox	Bounding Box
CNN	Convolutional Neural Network
COCO	Common Objects in Context
DL	Deep Learning
FLOP	Floating point Operation
FN	False Negative
FP	False Positive
fps	Frames per second
GPU	Graphics Processing Unit
ICCAT	International Commission for the Conservation of Atlantic Tunas
IoU	Intersection over Union
mAP	mean Average Precision
MLP	Multilayer Perceptron
TP	True Positive
VIA	VGG Image Annotator
VxC	Visión por Computador

1. Introducción

El desarrollo sostenible trata de satisfacer las necesidades de la generación presente sin comprometer la capacidad de las generaciones futuras para satisfacer sus propias necesidades. Se trata de lograr un equilibrio entre el desarrollo económico, el desarrollo social y la protección del medio ambiente. La integración de las nuevas tecnologías, *software* y *hardware*, orientadas a buscar soluciones que contribuyan a este logro, incentiva muchos de los trabajos y desarrollos actuales. Recientemente, en febrero de 2020, la Comisión Europea publicó el *LIBRO BLANCO sobre inteligencia artificial* [1], en el que Europa reconoce que la Inteligencia Artificial (IA) puede tener un papel importante en la consecución de los Objetivos de Desarrollo Sostenible (ODS), además de ser una tecnología estratégica que ofrece numerosas ventajas a ciudadanos, empresas y la sociedad en su conjunto [2].

Desarrollar tecnologías y estrategias que repercutan en una mejor conservación de la vida marina y colaboren a hacer sostenible los océanos y sus recursos es vital para todos. En el sector pesquero, tanto en la pesca extractiva como en la acuicultura, es de gran importancia controlar la población y el *stock* de especies. En general, existe una gran necesidad de monitorizar especies marinas tanto para generar nuevo conocimiento de las especies (por ejemplo, modelar fases de engorde y crecimiento, estimar su estado de salud y estudiar su comportamiento) como para desarrollar líneas de sostenibilidad del medio marino (por ejemplo, estudiar tamaño y estructura de poblaciones y establecer cuotas de pesca). El atún rojo del Atlántico (*Thunnus thynnus* o *bluefin tuna*) es una especie altamente migratoria, y para poder evaluar y controlar su población es necesario establecer acuerdos internacionales [3]. Debido a esta dificultad y a su importancia comercial, desde el año 2011, esta especie está en la lista roja de la IUCN (*International Union for Conservation of Nature's Red List of Threatened Species*), ya que se estima que, en tres generaciones, la población del atún rojo del Atlántico disminuyó un 51% [4].

Por suerte, varios planes de recuperación han sido puestos en marcha desde el 2006 [3] [4]; no obstante, la monitorización y el control de la población es una tarea compleja y su automatización es una labor vital para la conservación de la especie [4]. En este trabajo se presentan desarrollos basados en aplicar técnicas de visión por computador y aprendizaje profundo para detectar ejemplares de atún rojo del Atlántico en imágenes subacuáticas, y así poder estimar la biomasa de dichos ejemplares. Se trata de explorar técnicas y metodologías de las Convolutional Neural Networks (CNNs), aplicables al procesamiento de imágenes subacuáticas y la caracterización de individuos en ellas, para la extracción de información orientada a la sostenibilidad del medio marino.

1.1 Motivación

La principal motivación de este proyecto es poder introducir técnicas de *deep learning* (DL) en un proyecto de investigación real que mejore significativamente los resultados obtenidos con técnicas clásicas de reconocimiento de formas. Esto permite explorar las técnicas, arquitecturas y metodologías del estado del arte y usar equipos con una gran potencia computacional.

Por otra parte, una motivación personal de gran peso es poder colaborar en un proyecto de conservación de una especie en peligro de extinción, como es la especie *Thunnus thynnus*, cuya población es difícil de monitorizar y controlar dada su tendencia migratoria.

Este trabajo se contextualiza en el proyecto BIACOP (ref. 2013/410/UE Comisión Europea), proyecto de investigación que se lleva a cabo por el grupo de Visión por Computador (VxC) en el Instituto Universitario de Automática e Informática Industrial (Ai2) de la Universitat Politècnica de València, cuyos resultados hasta el momento presentan un método para poder medir de forma automática la longitud del atún rojo del Atlántico a partir de vídeos grabados con un sistema de visión estereoscópica. No obstante, este método requiere ajustar una serie de parámetros manualmente para adaptar el sistema a las características de la imagen tales como el brillo y el contraste, por lo que es necesaria una modificación del procedimiento actual para poder adecuar el sistema a dichas características de manera automática.

1.2 Objetivos

Las imágenes contienen una gran cantidad de información que es necesario extraer y localizar en ellas de forma automática para configurar sistemas que trabajen de forma autónoma o ayuden a la toma de decisiones. En los últimos años, el volumen de imágenes a procesar ha crecido de forma exponencial. Este crecimiento ha propiciado el desarrollo de sistemas de almacenamiento, tanto a nivel físico (servidores tipo NAS) como de organización (bases de datos tipo MySQL), así como de sistemas de procesamiento potentes (*clusters* de GPUs) y las técnicas y métodos empleados (DL y CNNs).

El objetivo de este trabajo es estudiar las arquitecturas, técnicas y metodologías de las CNNs y DL aplicadas a los problemas de detección y segmentación de objetos en imágenes, trabajando con herramientas con capacidad de almacenar grandes volúmenes de imágenes y confeccionando conjuntos de datos anotados. Aplicando este objetivo al problema a solventar en el proyecto de investigación, se proponen los siguientes objetivos concretos:

- Mejorar el método actual de segmentación del proyecto de investigación, incrementando la precisión de la segmentación y el número de individuos segmentados en cada vídeo adquirido, ya sea detectando peces en *frames* donde el proyecto actual es incapaz de detectarlos o detectando más individuos en cada *frame*.

- Reducir el tiempo de procesamiento de cada *frame*, y por ende de los vídeos, aprovechando la potencia de las CNNs aplicadas sobre una GPU de última generación.

1.3 Estructura del documento

Este documento está estructurado en cinco capítulos. En el capítulo dos, *Estado del arte y punto de partida*, se documentan de forma breve los principales algoritmos de DL para la detección y segmentación de objetos y las propuestas existentes para la aplicación de DL a la medición de peces, además de contextualizar este trabajo dentro del proyecto BIACOP. En el capítulo tres, *Solución propuesta,* se describen la generación del conjunto de datos y la tecnología, las técnicas de DL y las herramientas de validación propuestas para abordar el problema. En el capítulo cuatro, *Desarrollo de la solución propuesta*, se detalla cómo se ha pasado de la propuesta a la solución final, comentando las fases del desarrollo, decisiones tomadas, experimentos y resultados. Finalmente, en el capítulo de *Conclusiones*, se discute la consecución de los objetivos presentados en el primer capítulo y se reflexiona acerca de los conocimientos puestos en práctica y los conocimientos adquiridos durante este trabajo, además de presentar posibles trabajos futuros relacionados con este proyecto.

2. Estado del arte y punto de partida

Si bien es cierto que la estimación de la biomasa en especies marinas antes de su captura es un problema que normalmente requiere de gran cantidad de trabajo por parte de operarios, son pocas las aplicaciones que utilizan técnicas de visión por computador para automatizar totalmente este proceso. En la literatura se documentan aplicaciones que implementan técnicas de *deep learning* (DL) para estimar medidas biométricas de especies marinas a través de imágenes, pero estos procesos requieren que los peces sean capturados.

En este capítulo se exponen, en primer lugar, las técnicas y aplicaciones recientes de DL relativas a los problemas de detección y segmentación de objetos en imágenes y la literatura reciente sobre la monitorización de especies marinas. En el segundo apartado se presenta el punto de partida sobre el cual se sitúa este trabajo, en el cual se comentarán las diferentes fases del proyecto BIACOP.

2.1 Estado del arte

Para facilitar la comprensión del estado del arte, este apartado se ha dividido en tres subapartados. En el subapartado 2.1.1 se exponen los principales algoritmos de detección y segmentación de objetos que emplean técnicas de DL de la actualidad, explicando brevemente el funcionamiento de cada algoritmo y resaltando sus posibles mejoras respecto a su versión predecesora. También se introduce el concepto y las bases de la técnica conocida como *transfer learning*, de gran importancia en el DL y el área de la visión por computador. El subapartado 2.1.2 se dedica a revisar aplicaciones para la monitorización del atún rojo del Atlántico en libertad que existen hoy en día en la literatura y las técnicas de VxC que utilizan. Finalmente, en el subapartado 2.1.3 se comentan las aplicaciones recientes que utilizan técnicas de DL para estimar datos biométricos de especies marinas.

2.1.1 Detección y segmentación de objetos con *deep learning*

El objetivo de los algoritmos de aprendizaje automático aplicados a la clasificación de imágenes consiste en tomar conjuntos de imágenes e identificar en ellas patrones que se pueden utilizar, para discriminar varios tipos de objetos entre sí. Los modelos basados en DL permiten que estructuras computacionales compuestas por múltiples capas de procesamiento aprendan representaciones de datos con múltiples niveles de abstracción. Los parámetros de representación internos de cada capa son calculados a partir de la representación de la capa anterior y ajustados mediante algoritmos de *backpropagation*. Estos modelos consiguen aprender la estructura intrínseca de grandes conjuntos de datos, sin necesidad de utilizar la ingeniería de extracción de características clásica.

Desde el desarrollo de las redes neuronales convolucionales (CNN), las técnicas de DL para la visión por computador han avanzado a gran velocidad, proponiendo estrategias y modelos para resolver el problema de detectar varias instancias de objetos en una misma imagen. Se trata de caracterizar cada instancia por el rectángulo que la contiene en la imagen, o lo que se denomina *bounding box* (bbox). En ese caso, una arquitectura de red neuronal con cuatro nodos de salida, uno para cada una de las características del bbox (las cuales podrían ser coordenadas x e y superior izquierda, ancho y alto) sería incapaz de realizar buenas detecciones.

Técnicas de CNNs para la detección y segmentación de objetos

En 2014 se propuso el algoritmo R-CNN [5], el cual extrae 2000 propuestas de regiones de interés por cada imagen y clasifica posteriormente dichas regiones, solventando el problema de detectar múltiples objetos en una misma imagen. Este algoritmo evolucionó en Fast R-CNN en 2015 [6] y Faster R-CNN en 2016 [7], siendo este último su versión más actualizada en cuanto a detección de objetos se refiere. Por otra parte, para la tarea de segmentación, en 2017 se propuso Mask R-CNN [8], el cual adapta el algoritmo Faster R-CNN para añadirle el componente de segmentación, funcionando en paralelo al de detección e incrementando así muy poco el tiempo de ejecución del algoritmo. Ya en 2020, se propone el módulo PointRend [9], el cual se puede combinar con Mask R-CNN y así devolver segmentaciones de una resolución mucho mayor con un coste computacional mucho menor. Por ejemplo, una detección con máscaras de 224x224 píxeles de resolución tiene un coste de 0,9 GFLOPs en PointRend, frente a 34 GFLOPs en Mask R-CNN.

Otro enfoque al problema de detectar varios objetos en una misma imagen es el propuesto en el algoritmo YOLO (*You Only Look Once*) en 2016 [10]. Esta arquitectura trabaja dividiendo la imagen en celdas mediante una rejilla. De cada celda proporciona dos posibles bboxes con un valor de confianza. Al mismo tiempo, cada celda se clasifica en una clase con otro valor de confianza. Combinando todo lo anterior y eliminando las bboxes con valores de confianza bajos o duplicadas se pueden obtener múltiples detecciones o instancias por cada imagen. YOLO es muy rápido, siendo capaz de ejecutarse a una velocidad de 45 *frames* por segundo (fps), frente a los cinco fps de Faster R-CNN. Tanto este algoritmo como la familia de algoritmos R-CNN se muestran en una línea temporal en la Figura 2.1.

Figura 2.1: cronograma temporal de las técnicas más representativas para la detección y la segmentación de objetos en imágenes.

Aprendizaje por transferencia o *transfer learning*

En el aprendizaje tradicional, cada tarea y cada dominio de datos es considerado de forma aislada. Se trata de un aprendizaje en base a tareas específicas, con conjuntos de datos y entrenamiento de modelos separados para cada caso. Dichos modelos no retienen ningún conocimiento que pueda transferirse a otro modelo para abordar una nueva tarea. El aprendizaje por transferencia se basa en aprovechar el conocimiento (características, pesos, etc.) de modelos previamente entrenados para entrenar modelos nuevos e incluso abordar problemas para los que se dispone de un conjunto de datos pequeño [11].

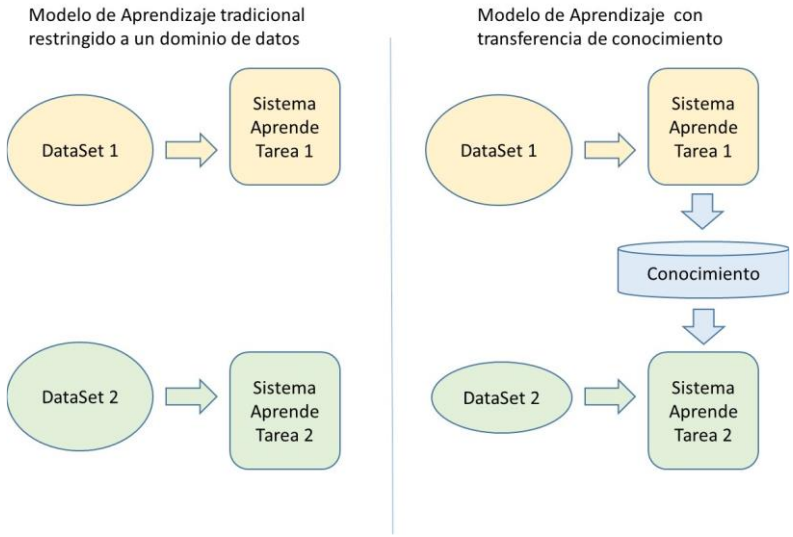


Figura 2.2: aprendizaje tradicional frente a transfer learning.

Los algoritmos de aprendizaje supervisado sin transferencia obtienen resultados muy pobres o fallan cuando no se dispone de suficientes datos de entrenamiento para las tareas. El aprendizaje por transferencia admite el uso del conocimiento de tareas aprendidas previamente para aplicarlo a otras nuevas relacionadas. Por ejemplo, si se dispone significativamente de más datos para la Tarea 1 que para la Tarea 2, se puede utilizar el aprendizaje de la Tarea 1 y generalizar este conocimiento (características, pesos) para la Tarea 2 de la que significativamente se dispone de muchos menos datos. La Figura 2.2 trata de escenificar estos conceptos y diferencias entre dichos modelos de aprendizaje.

En el caso de problemas en el dominio de VxC, ciertas características de bajo nivel, como bordes, formas, esquinas e intensidad, se pueden compartir entre tareas y, por lo tanto, permiten la transferencia de conocimientos entre tareas. Además, como se describe en la Figura 2.2, el conocimiento de una tarea existente actúa como una entrada adicional cuando se aprende una nueva tarea objetivo. En el área del DL, la técnica de *transfer learning* que "transfiere" este conocimiento entre tareas se denomina *feature extraction*. Esta estrategia consiste en entrenar la arquitectura de CNN con un conjunto de datos con una gran cantidad de muestras, para posteriormente eliminar una o varias capas finales de esta arquitectura y sustituirlas por las capas apropiadas para el problema a resolver. Así, finalmente se entrena la red con los datos y anotaciones de la tarea objetivo a partir de estos pesos preentrenados. En la Figura 2.3 se ilustra

el funcionamiento de esta técnica con un ejemplo de red con tres capas convolucionales y dos densamente conectadas (*fully connected*, fc) que terminan con la función de activación *softmax*. La red de la izquierda de la Figura 2.3 Representa una CNN que se entrena con unos datos y anotaciones de un problema similar al objetivo final. Esta red y sus pesos, todas las capas excepto la última densamente conectada (y la *softmax*), son transferidos y se acoplan a un clasificador o detector (en función de la tarea a cumplir) y finalmente se entrenan con los datos y anotaciones del problema objetivo (Figura 2.3, red de la derecha). Por lo general, las arquitecturas de CNNs más populares (Inception, VGG, ResNet, etc.) comparten con la comunidad los pesos entrenados con varios conjuntos de datos relativos al área de la VxC, como suelen ser COCO o ImageNet. De esta manera, entrenar una CNN cuando se dispone de una cantidad de datos reducida es posible.

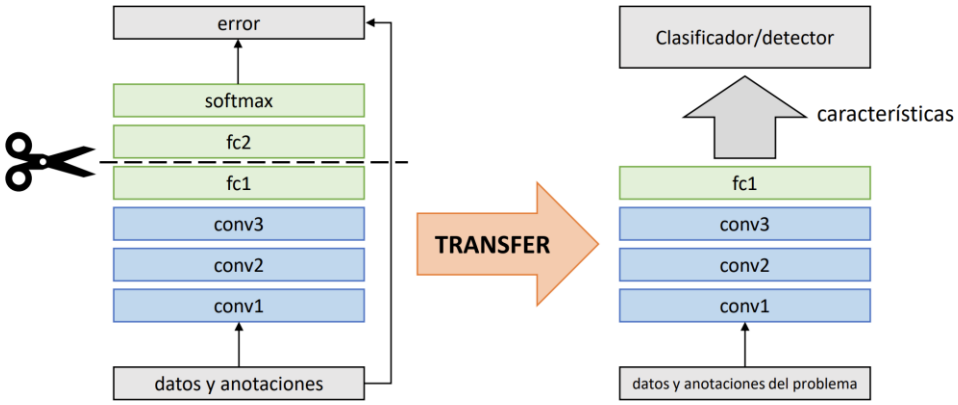


Figura 2.3: técnica feature extraction sobre dos arquitecturas de ejemplo. Las capas de color azul y verde se corresponden con capas convolucionales y capas densamente conectadas (fully connected), respectivamente. El conocimiento aprendido en la red de la izauierda es transferido para abordar un problema similar a la red de la derecha.

Otra técnica complementaria a *feature extraction* es *fine-tuning*. *Fine-tuning* consiste en "congelar" las capas de la CNN con pesos preentrenados mediante *feature extraction*, a excepción de sus últimas capas (en el ejemplo de la Figura 2.3, estas podrían ser desde la fc1 hacia abajo, en función del problema), antes de volver a entrenar el modelo con el conjunto de datos de la tarea objetivo. Esta técnica ajusta las características visuales más abstractas del modelo, haciéndolo más relevante para la tarea objetivo.

2.1.2 Monitorización del atún rojo del Atlántico

Una correcta monitorización de las especies marinas en libertad requiere de un cuidado extremo, sin manipulación de los individuos para evitar dañarlos o estresarlos. Los sistemas encargados de obtener las imágenes han de estar sumergidos, y las condiciones bajo el agua introducen diversas dificultades, como inestabilidad, variaciones de luminosidad o deformaciones [12].

Los métodos más comunes de adquisición de imágenes son sistemas de cámaras estéreo sumergidas, para poder medir así la distancia a los objetos a detectar y las dimensiones del objeto detectado. Este tipo de sistemas es el recomendado por la ICCAT (*International Comission for the Conservation of Atlantic Tunas*), y en ocasiones se combina con técnicas acústicas [13].

La técnica más usada para estimar los tamaños de las especies monitorizadas requiere de la intervención humana, ya que se utiliza un software que necesita del marcado, en cada *frame* del vídeo adquirido, de los extremos del individuo, normalmente el morro y la cola, para poder inferir las medidas. Sin embargo, se han propuesto recientemente métodos de segmentación de objetos para automatizar esta fase. Los métodos más utilizados se basan en métodos convencionales de segmentación, con técnicas como *local thresholding*, la cual segmenta objetos en la imagen en función de la intensidad de los píxeles cercanos a cada píxel de la imagen. Posteriormente se filtran los resultados con la aplicación de un modelo de la especie a monitorizar [12].

2.1.3 *Deep learning* en la estimación de medidas de peces

La estimación de medidas biométricas de la población de peces a controlar permite adoptar medidas para reducir los recursos utilizados y conseguir una producción sostenible. Las aplicaciones recientes muestran resultados prometedores de la mano de la visión por computador en combinación con técnicas de DL [14].

En cuanto a la adquisición de imágenes, los proyectos que trabajan con peces en libertad suelen utilizar sistemas de cámaras estereoscópicas para poder estimar la profundidad, mientras que en las aplicaciones que trabajan con imágenes de peces una vez estos han sido capturados, utilizan una [15] o dos cámaras [16][17]. Por ejemplo, en [18], dado que trabajan con peces en libertad relativamente pequeños, los hacen pasar por una jaula con una fuente de luz indirecta y un sistema estereoscópico de visión, para poder obtener imágenes con la iluminación adecuada y con un fondo contrastado y homogéneo.

En cuanto a las técnicas de DL aplicadas sobre las imágenes, las aplicaciones recientes utilizan algoritmos de CNNs o R-CNNs [14]. Por ejemplo, se utiliza la arquitectura Mask R-CNN [18][15] para segmentar el contorno del pez o detectar la cabeza y la cola e inferir así la longitud de este. Estos algoritmos necesitan de un gran volumen de datos de entrada y, a menudo, incorporan técnicas de *image augmentation* [14] para incrementar artificialmente el número de muestras de entrenamiento. Además, la técnica de *transfer learning feature extraction* es bastante popular [14]. Una vez se ha obtenido la máscara generada mediante DL, se puede refinar la segmentación utilizando técnicas de procesamiento de imagen.

2.1.4 Crítica al estado del arte

En el apartado 2.1.1 se presentan varios modelos de DL del área de la VxC, donde destacan los modelos Mask R-CNN y PointRend. Estos parecen los más adecuados para este trabajo, ya que resuelven directamente el problema de la segmentación de objetos en imágenes. No obstante, se debe estudiar su uso y comprobar tanto si la resolución de las máscaras de Mask R-CNN no es demasiado baja, como si el refinamiento de las máscaras de PointRend es lo suficientemente preciso. Este apartado también introduce las principales técnicas de *transfer learning* en DL, técnicas complementarias al uso de los modelos presentados y que permiten entrenar las CNNs con un conjunto de datos reducido.

En el apartado 2.1.2 se revisan las principales aplicaciones para la monitorización del atún rojo del Atlántico en libertad. Estas aplicaciones proponen el uso de sistemas de cámaras estereoscópicos, junto a otros métodos complementarios como el uso de ecosondas. Los individuos a monitorizar se detectan o segmentan en las imágenes adquiridas por medio del trabajo humano o con métodos de segmentación tradicionales. Con ambos métodos es necesaria la supervisión manual, ya que los parámetros de los métodos tradicionales han de ajustarse para adaptar el sistema a las características de las imágenes adquiridas.

En el apartado 2.1.3 se comentan los trabajos recientes de estimaciones de datos biométricos de especies marinas a partir de imágenes por medio del uso de técnicas de DL. Estos trabajos suelen utilizar métodos comentados en el apartado 2.1.1, como algoritmos de CNNs y R-CNNs, incorporando técnicas de *transfer learning* y refinando las segmentaciones producidas en algunos casos. No obstante, estos proyectos trabajan con imágenes de peces una vez han sido capturados, o con peces relativamente pequeños, los cuales pasan por dentro de una jaula con la iluminación adecuada y un fondo contrastado y homogéneo, facilitando así la segmentación automática de los individuos.

Para este trabajo, parece adecuado el estudio del uso de modelos y técnicas de DL de segmentación o detección de objetos en imágenes. Estas técnicas ya han sido utilizadas en especies marinas en libertad con la ayuda de jaulas con la iluminación y el fondo adecuados. Sin embargo, el uso de este material es inviable en este trabajo, ya que el atún rojo del Atlántico llega a tener una longitud de tres metros [4]. Por otra parte, las técnicas de refinamiento de las segmentaciones producidas no utilizan DL, por lo que el estudio del uso de un modelo como PointRend es interesante para este proyecto.

2.2 Punto de partida

El presente trabajo parte de los desarrollos realizados recientemente por el grupo VxC del Ai2 utilizando visión estereoscópica para la estimación de dimensiones del atún rojo en entornos naturales: entornos subacuáticos con peces vivos nadando libremente. Dichos desarrollos se enmarcan en proyectos como BIACOP, el cual forma parte de uno de los planes de recuperación de la población de atún rojo del Atlántico propuestos por la ICCAT [3]. Los artículos científicos [12] y [13] contienen los avances realizados que se fundamentan en la aplicación de técnicas de visión estereoscópica o visón 3D y técnicas acústicas en entornos subacuáticos para el estudio del atún rojo. La Figura 3.1 trata de resumir los grandes bloques que se abordan y desarrollan en BIACOP:

1. Adquisición de vídeos estereoscópicos
2. Segmentación de individuos en *frames* sincronizados
3. Ajuste del modelo a cada silueta sincronizada
4. Estimación de medidas biométricas

De estos bloques, este trabajo se centra en mejorar el segundo bloque, la segmentación de individuos, con el objetivo de así obtener unos resultados de mayor calidad en los bloques posteriores, y así mejorar las prestaciones del conjunto total de bloques.

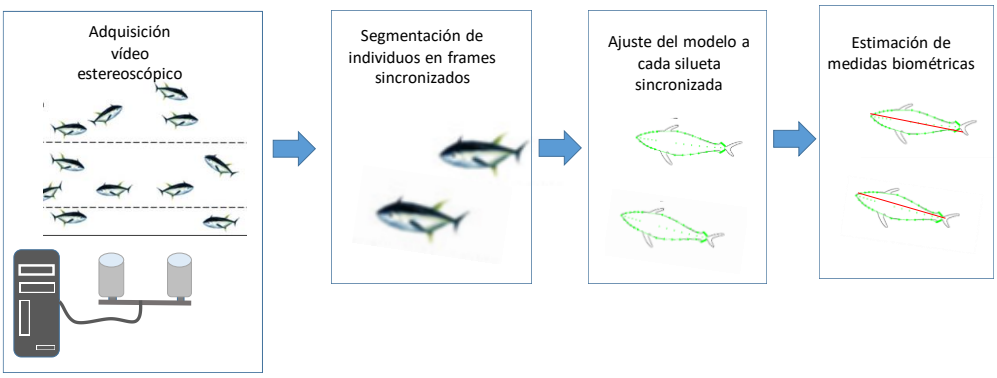


Figura 3.1: bloques o fases sobre los que se desarrolla el proyecto BIACOP. Este trabajo trata de mejorar el bloque de segmentación, para así mejorar las prestaciones del proyecto en conjunto.

Hasta el momento, el grupo de VxC ha utilizado métodos tradicionales de *local thresholding*, junto con algoritmos de detección de bordes para segmentar los individuos. Tener la certeza de que lo extraído de la imagen de forma totalmente automática corresponde al cuerpo de un solo pez es un paso esencial para lograr un proceso de dimensionado de individuos completamente automático. La problemática que conlleva el detectar automáticamente individuos, que debido a la flexión de su cuerpo mientras nadan se perciben con formas, tamaños y orientaciones muy distintas de un fotograma al siguiente en los vídeos adquiridos, condujo al grupo de VxC al diseño de un modelo 2D robusto y deformable que se adapta al tamaño y el gesto de los peces mientras nadan. Se trata de aplicar un modelo deformable de la silueta del pez desde una vista ventral, para verificar que la segmentación se corresponde con un atún completo y finalmente se estiman las medidas. Para el dimensionando de individuos es necesario la adquisición de imágenes con un

sistema de cámaras estereoscópicas y el procesamiento de ambas imágenes que deben ser adquiridas de forma sincronizada. Dicho procedimiento asegura unos valores muy precisos, pero bastante costosos temporalmente de obtener.

Los entornos naturales representan un desafío muy exigente a la hora de desarrollar soluciones totalmente automáticas. La naturaleza del entorno (medio marino) en el que se adquieren las imágenes conlleva una gran dificultad en el proceso de segmentación al trabajar con imágenes con luminosidad variable, brillos, bajo contraste debido a la turbidez del agua, peces superpuestos, etc. Esta dificultad implica que, en algunos casos, se obtengan pocos resultados tras la segmentación, teniendo que reajustar los parámetros del método de análisis de la imagen para poder segmentar los peces en imágenes. Estas técnicas clásicas proporcionan buenos resultados, pero no posibilitan una generalización del problema; el método desarrollado requiere de una actualización en su proceso de análisis de imagen, detección y segmentación para poderse automatizar completamente. Las técnicas de *deep learning* (DL) permiten generalizar problemas desarrollando arquitecturas de redes neuronales específicas para un fin, si se dispone de una gran cantidad de datos para entrenarlas. Las técnicas de DL han supuesto una verdadera revolución, superando el estado del arte en ámbitos como el reconocimiento del habla, el reconocimiento de caras, el reconocimiento de caracteres y, muy particularmente, en el análisis de imágenes médicas y en bioinformática. Por esta razón, los métodos de DL de detección y segmentación de objetos forman una propuesta adecuada para cumplir este cometido.

2.2.1 Adquisición de vídeos estereoscópicos subacuáticos

A continuación, se detalla el proceso de adquisición de las imágenes y los vídeos estereoscópicos utilizados en este trabajo. Las imágenes utilizadas en el proyecto se obtienen con dos sistemas de cámaras estereoscópicas diferentes. El primero es un sistema comercial denominado AQ1 AM100 [19], compuesto por dos cámaras Gigabit Ethernet. El sistema AQ1 genera dos vídeos sincronizados con resolución de 1360x1024 píxeles, a una frecuencia de 12 fps. Como se puede ver en la Figura 3.2, las cámaras se sumergen dentro de las jaulas de engorde de los peces. En los experimentos realizados, las cámaras se han colocado en diversas posiciones y ángulos, obteniendo así vídeos de vista dorsal, ventral y lateral de los peces.

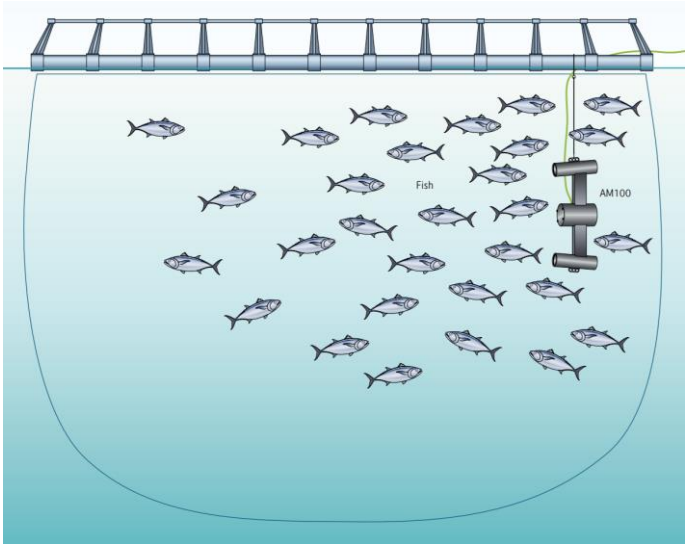


Figura 3.2: ejemplo de configuración del sistema AQ1 AM100 en una jaula con peces. Extraído de [19].

El segundo sistema está compuesto por dos cámaras FLIR. Se trata de un sistema desarrollado por el grupo VxC del Ai2, y utilizado en el proyecto BIACOP. Este sistema genera vídeos sincronizados con 2048x1536 píxeles de resolución, a una frecuencia de 33 fps. La Figura 3.3 muestra el montaje de este sistema ya sumergido en el fondo de una jaula.

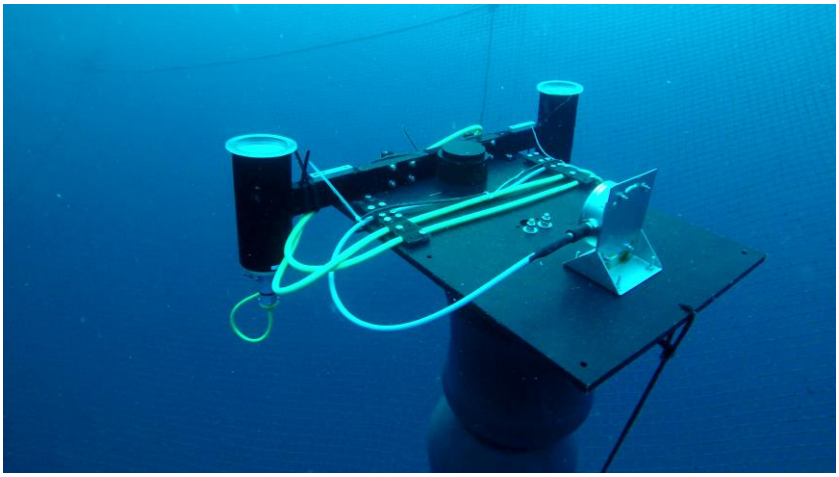


Figura 3.3: sistema estereoscópico con cámaras FLIR.

Dado que ambos sistemas están compuestos por cámaras y lentes diferentes, el color y el aspecto final de las imágenes adquiridas es diferente, lo cual es beneficioso para el entrenamiento de los modelos de DL, ya que estos podrán realizar las detecciones con una mayor capacidad de generalización.

Las grabaciones de los vídeos con peces han sido realizadas en dos lugares diferentes. Unas se han realizado en las aguas del Mediterráneo, en concreto, en las instalaciones del grupo

empresarial Balfegó, a cinco kilómetros del puerto de l'Atmella de Mar, en Tarragona, en jaulas de 50 metros de diámetro en la superficie del agua y de 30 metros de profundidad (Figura 3.4).

Figura 3.4: jaula de 50 m de diámetro y 30 m de profundidad en l'Atmella de Mar, Tarragona.

Los otros vídeos corresponden a grabaciones realizadas las instalaciones ICAR (Infraestructura para la acuicultura del atún rojo del Atlántico), propiedad del Instituto Español de Oceanografía (IEO), en Mazarrón, Murcia. El tanque en el cual se graba está situado en tierra firme y mide 20 metros de diámetro y diez de profundidad (Figura 3.5). En el ICAR, los investigadores están estudiando la reproducción del atún rojo del Atlántico en cautividad.

Figura 3.5: unidad de cultivo de túnidos en las instalaciones del Instituto Español de Oceanografía, en Mazarrón, Murcia. Extraído de [20].

En ambos casos, las grabaciones han sido realizadas desde el año 2018 con cierta regularidad, por lo que se cuenta con una gran cantidad de vídeos sobre los cuales extraer imágenes para el conjunto de datos y con los que entrenar las CNNs.

2.2.2 Estimación de medidas biométricas mediante un modelo de la silueta

Tras segmentar los atunes en los *frames* de los vídeos estereoscópicos adquiridos, se utiliza un modelo deformable de la silueta del atún para estimar de forma automática sus medidas biométricas. Este modelo será utilizado en el apartado 4.3 para presentar resultados de la estimación de medidas con la incorporación de diferentes modelos de CNNs.

El modelo deformable del atún rojo del Atlántico, utilizado en el proyecto BIACOP, está descrito en el artículo [21]. Este modelo está formado por cinco parámetros $\mathbf{M} = [s_x, s_y, l, \alpha, \theta]$, donde los dos primeros son la posición de la punta del morro, el tercero es la longitud de la columna vertebral y los dos últimos son el ángulo de la cabeza con respecto al eje horizontal y el ángulo de flexión de la cola, respectivamente. Este modelo y sus parámetros se pueden visualizar en la esquina superior izquierda de la Figura 3.6. Posteriormente, se actualizó este modelo en [13], con ocho parámetros $\mathbf{M_e} = [s_x, s_y, l, \alpha, \theta, \mathbf{w}, l_p, s_p]$, añadiendo los parámetros $[l_p, s_p]$, longitud e inclinación del segmento perteneciente a la parte trasera del pedúnculo caudal y **w**, parámetro de anchura. Este modelo se caracteriza por 18 puntos, de los cuales 14 se distribuyen a lo largo de la longitud del pez y cuatro para modelar la parte trasera del pedúnculo caudal. En la parte inferior de la Figura 3.6 se pueden ver estos puntos del modelo y en la parte derecha se puede ver el modelo aplicado a la imagen real de un atún.

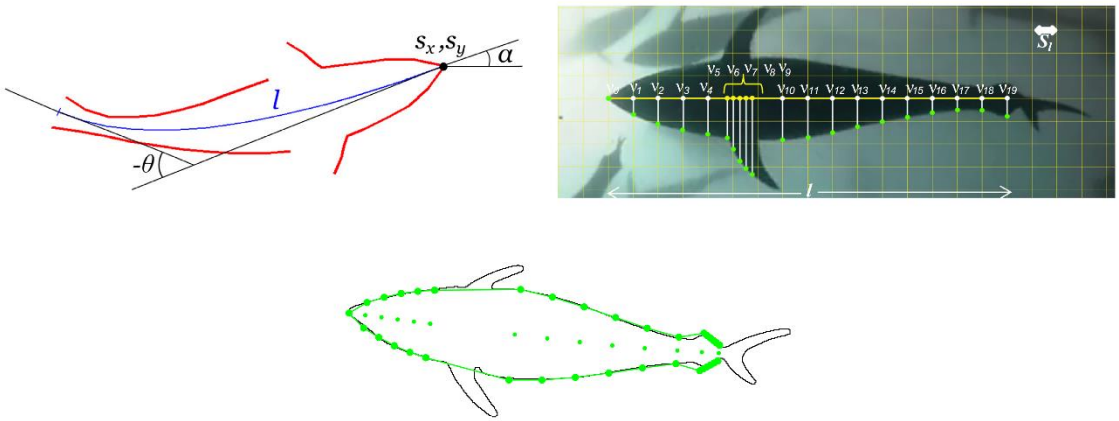


Figura 3.6: modelos deformables del atún rojo del Atlántico. Arriba: modelo propuesto en [21]*, con la localización de los 14 puntos del cuerpo en una imagen real (derecha). Abajo: modelo mejorado propuesto en* [13]*, con un parámetro adicional para el pedúnculo caudal. Extraído de* [13] *y* [21].

Partiendo de las segmentaciones pertenecientes al mismo *frame* de cada cámara del sistema estereoscópico y el modelo comentado anteriormente, se ajusta el modelo a cada segmentación y se obtienen los valores de los parámetros. Además, el ajuste devuelve un índice de error que sirve para descartar las segmentaciones erróneas. Finalmente, las medidas de la longitud total y anchura se transforman en medidas en 3D, utilizando los parámetros de calibración del sistema de cámaras y técnicas de triangulación 3D. Las medidas tomadas del mismo pez en *frames* sucesivos se utilizan para aumentar la precisión de estas, detectando automáticamente que pertenecen al mismo pez con técnicas de *tracking*.

2.2.3 Resumen

En este capítulo se exponen de forma breve, las técnicas y aplicaciones recientes de DL relativas a los problemas de detección y segmentación de objetos en imágenes y la literatura reciente sobre la monitorización de especies marinas. Además, en la sección 2.2 se introduce el contexto en el que se aplicarán los desarrollos realizados. Se presentan los desarrollos del proyecto de investigación BIACOP, que suponen el punto de partida de este trabajo. De dicho proyecto, se exponen las distintas fases del procesamiento automático desarrollado para la estimación del tamaño de ejemplares de atún rojo del Atlántico y se describen más detalladamente las fases de adquisición de vídeos estereoscópicos y la estimación de medidas biométricas a partir de un modelo deformable de la silueta del atún, pues son utilizados también en el presente trabajo como fuentes de información y como método de validación, respectivamente.

3. Solución propuesta

En este trabajo se propone desarrollar un sistema capaz de localizar y segmentar de forma autónoma, los individuos de *Thunnus thynnus* (*bluefin tuna* o atún rojo del Atlántico), que aparecen en imágenes y vídeos subacuáticos. La caracterización y segmentación de peces en imágenes, es un paso esencial tanto para la estimación de biomasa de individuos como la identificación de especies y la estimación de *stock* de poblaciones marinas. El presente trabajo forma parte del contexto de un proyecto más amplio cuyo objetivo es la estimación de biomasa de individuos para continuar con un estudio de su población (modelos de engorde, comportamiento, etc.) útil tanto para científicos, biólogos como acuicultores.

El desarrollo de técnicas no invasivas utilizando información multisensorial (ecosondas y cámaras) para calcular biomasa de peces en condiciones reales de acuicultura (peces nadando libremente) promueve el desarrollo sostenible de la acuicultura, optimizando el uso eficiente de recursos. La estimación de biomasa es una tarea muy importante para la optimizar la alimentación, detectar enfermedades y parásitos, y en general supervisar la producción en el sector de la acuicultura. Además, se trata de una fuente de conocimiento importante y una tarea vital para labores de conservación del medio marino, pudiendo aportar cuotas reales de pesca y evitando la sobreexplotación de especies en peligro. Por otra parte, la automatización del proceso de estimación puede reducir en gran medida tanto el coste en horas de trabajo manual, como el coste económico o el tiempo necesario para recopilar grandes volúmenes de información, lo que conllevará a una mejora de los resultados obtenidos a partir de métodos estadísticos.

La propuesta en este trabajo es emplear técnicas de visión por computador y DL que automaticen completamente el proceso de segmentación y mejoren la calidad y cantidad de los resultados; es decir, que consigan segmentar con más precisión una mayor cantidad de peces. Para ello se explora la incorporación de estas técnicas al procedimiento del proyecto de investigación, comprobando si pueden mejorar el procedimiento de segmentación basado en técnicas clásicas.

3.1 Datos y preprocesado

Para poder llevar a cabo el entrenamiento de una red neuronal, se necesitan grandes cantidades de datos; en este caso, imágenes con las que poder alimentar el modelo. Sin embargo, gracias a las técnicas de *transfer learning*, es posible transferir los pesos resultado de haber entrenado la red con una gran cantidad de imágenes, tal y como se expuso en el apartado 2.1.1. De esta manera, no es necesario disponer de una gran cantidad de imágenes para un problema similar.

Partiendo de esta premisa, se proponen dos líneas de trabajo para disponer de la cantidad adecuada de imágenes. En primer lugar, el conjunto de datos utilizado en este trabajo está compuesto por *frames* de vídeos e imágenes adquiridos por el sistema de cámaras estereoscópico del proyecto BIACOP desde vistas dorsal y ventral, durante diferentes horas del día. Estas imágenes son etiquetadas manualmente con la ayuda del software VGG Image Annotator (VIA), generando archivos JSON con la segmentación anotada. Para obtener las anotaciones en forma de bbox, se toman los valores mínimos y máximos de las coordenadas x e y de cada segmentación.

Por otro lado, con el objetivo de aumentar sintéticamente el número de imágenes de entrenamiento, se propone emplear técnicas de *image augmentation*. Estas técnicas consisten en modificar las imágenes aplicándoles transformaciones tales como rotaciones, cortes y cambios de tono, brillo y contraste para incrementar así de manera artificial el número de imágenes con el que trabajar. De esta manera, la CNN entrenada adquiere una mayor capacidad de generalización y de adaptación a muestras nuevas y diferentes a las vistas en las propias imágenes del conjunto de datos.

3.2 Tecnología propuesta

En este apartado se exponen las herramientas y *frameworks* propuestos para el proyecto y sobre los que se utilizarán los modelos presentados en el apartado 3.2.3, tanto a nivel *software* como *hardware*.

El lenguaje de programación propuesto para el desarrollo de este proyecto es Python. Python es un lenguaje de programación versátil multiplataforma y multiparadigma con licencia para código abierto, lo que admite su uso de manera libre. Grandes compañías como Google o Facebook hacen un uso intensivo de Python y aceptan la ejecución de tareas tanto en entorno cliente como servidor. El lenguaje Python, junto con R, es el más popular en el área de la ciencia de datos. Esta popularidad se debe a que, mientras que hace unos años trabajar en el área del *deep learning* requería amplios conocimientos de C++ y CUDA, ahora es suficiente con conocer los aspectos básicos de Python [22]. Es por esto por lo que las implementaciones de las arquitecturas y algoritmos del estado del arte en el área de la visión por computador están desarrolladas en este lenguaje. Además, Python cuenta con multitud de librerías y paquetes de código abierto para problemas específicos de visión por computador, como son OpenCV y scikit-learn, entre otros.

A continuación, se describen brevemente las herramientas *software* propuestas para este proyecto: Conda, TensorFlow y Keras, PyTorch y VIA. Posteriormente, se comentarán los equipos propuestos para realizar los experimentos.

3.2.1 Conda

En Python se pueden crear entornos virtuales totalmente independientes los unos de los otros. Para ello, es necesario tener instaladas diferentes versiones de Python o diferentes versiones de paquetes sin que entren en conflicto unos con otros. Para poder utilizar varios entornos que manipulen diferentes librerías, paquetes y *frameworks* de Python junto con sus dependencias se ha utilizado Conda. Conda es un gestor de entornos de Python que permite crear y manipular dichos entornos de manera sencilla. Gracias a Conda se ha podido experimentar con los diferentes modelos y arquitectura, aunque estos requieran distintas versiones de las mismas librerías.

3.2.2 TensorFlow y Keras

Dentro de Python existen varias plataformas (o *frameworks*) para la implementación de redes neuronales, de las cuales se han utilizado las dos más populares, TensorFlow y PyTorch, dependiendo de la implementación del modelo a utilizar.

TensorFlow es la plataforma más utilizada actualmente, desarrollada por Google en 2015. Esta plataforma se encarga de la manipulación de tensores (contenedores de datos) y la compilación de modelos de aprendizaje automático. TensorFlow cuenta con varios kits de herramientas de distintos niveles de abstracción. El API de alto nivel y mayor nivel de abstracción es Keras. Esta librería fue desarrollada por François Chollet [22] a principios de 2015 y se convirtió en la API de TensorFlow más utilizada por las nuevas *startups*, estudiantes y nuevos investigadores en el campo del DL. En la Figura 3.7, se puede observar la evolución en la popularidad de TensorFlow y Keras. Las principales características por las que se utilizan TensorFlow y Keras en este trabajo son las siguientes:

- Permiten ejecutar el mismo código en CPU o GPU.
- Incorporan recursos para trabajar con CNNs.
- El nivel de abstracción de Keras posibilita la construcción de una arquitectura completa por bloques conectados entre ellos automáticamente, como si se tratara de bloques de LEGO [22].
- Keras fue pensado para experimentar con modelos de forma rápida, sin entrar en los detalles de la implementación en TensorFlow.

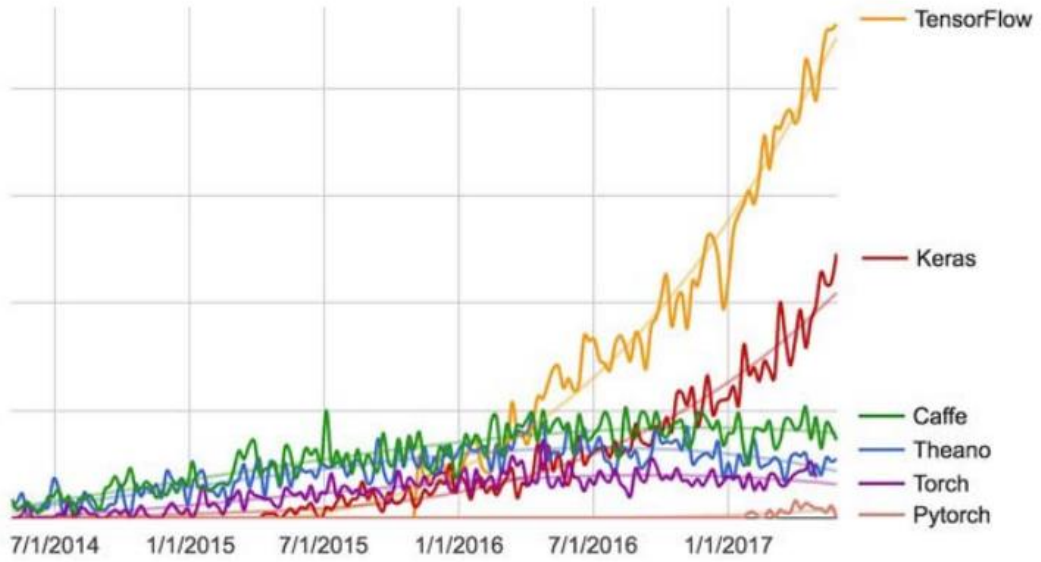


Figura 3.7: evolución del interés de las búsquedas de Google por varios de los frameworks de DL en el tiempo. Extraído de [22].

3.2.3 PyTorch

PyTorch es la plataforma de manipulación de tensores desarrollada por Facebook en 2018. Pese a ser más reciente que TensorFlow, muchas de las CNNs desarrolladas actualmente en tareas de detección y segmentación de objetos utilizan este *framework*, y su popularidad ha aumentado hasta equipararse con TensorFlow (Figura 3.8). De hecho, todas las arquitecturas que se utilizan en este proyecto han sido desarrolladas inicialmente en PyTorch. En el caso de los modelos de la familia R-CNN, esto se debe a que han sido desarrolladas por FAIR, propiedad de Facebook y desarrolladora de PyTorch.

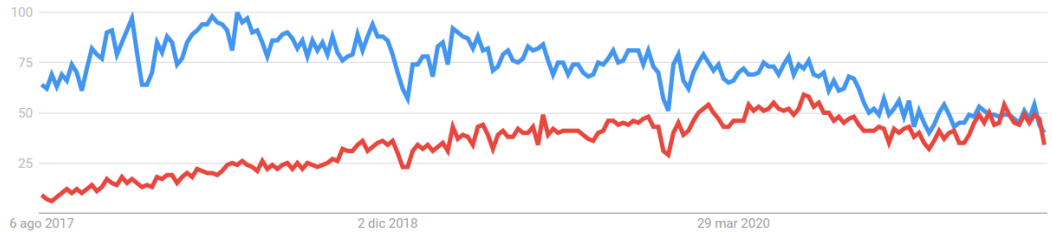


Figura 3.8: evolución del interés de las búsquedas de Google sobre TensorFlow (azul) y PyTorch (rojo) a lo largo del tiempo. Fuente: Google Trends.

3.2.4 VGG Image Annotator (VIA)

Para la creación manual del conjunto de datos utilizado se ha usado el software VGG Image Annotator (VIA). Este programa, incorporado en un archivo HTML y ejecutable en la mayoría de los buscadores web, permite crear anotaciones a partir de imágenes y vídeos. VIA es compatible con múltiples tipos de proyectos del área de VxC, tales como los de detección de objetos, clasificación y segmentación. Como salida genera un archivo JSON con las anotaciones realizadas.

3.2.5 Equipos

Los entrenamientos e inferencias de los modelos han sido realizados en dos equipos con características diferentes. El primero de ellos, equipo remoto del instituto Ai2, cuenta con una tarjeta gráfica NVIDIA RTX 2080 Ti con 12 GB de memoria de vídeo (VRAM). Por otro lado, se utilizó un equipo local con una tarjeta gráfica NVIDIA RTX 3090 de 24 GB de VRAM. Ambos equipos utilizan Linux como sistema operativo.

Para poder comparar los tiempos de entrenamiento e inferencia, los diferentes modelos fueron entrenados y se procesaron los vídeos del apartado 4.3.1 con el equipo local. El equipo remoto se utilizó para realizar los primeros entrenamientos (sin tomar las medidas de tiempos) y posteriormente como apoyo para pruebas.

3.3 Modelos propuestos

La detección de objetos es la técnica de VxC encargada de identificar instancias de un objeto (o clase) o varios en una imagen. Como resultado, esta técnica proporciona las coordenadas en las que se puede encuadrar dicho objeto o el bbox del objeto. La segmentación de imágenes, en cambio, devuelve el contorno de los objetos de una misma clase (*semantic segmentation*) o el contorno de cada una de las instancias de dicha clase (*instance segmentation*).

Para este trabajo, se proponen la utilización de varios modelos de DL del estado del arte. Entre los modelos utilizados para la detección de objetos en imágenes se incluyen Faster R-CNN, Mask R-CNN y YOLOv5, mientras que Mask R-CNN y PointRend son los modelos utilizados para la segmentación de objetos en imágenes. A continuación, se detalla cada uno de los modelos propuestos, dividiéndolos en función de su objetivo, ya sea la detección de objetos o la segmentación.

3.3.1 Faster R-CNN

Este modelo [7], desarrollado por FAIR (*Facebook Artificial Intelligence Research*) en 2016, es la evolución de un modelo anterior, Fast R-CNN [6], el cual ya mejoraba su predecesor, R-CNN [5]. Estos modelos se basan en la detección de objetos mediante la propuesta de regiones de interés en la imagen y su posterior procesamiento.

Faster R-CNN está compuesto por dos módulos. El primer módulo es una CNN que recibe como entrada una imagen de cualquier tamaño y devuelve una serie de propuestas de regiones rectangulares de interés junto a un valor de confianza o puntuación para cada una. En primer lugar, la imagen se procesa y se generan sus *feature maps*, sobre los cuales se desliza una pequeña CNN a modo de ventana deslizante (o *sliding window*), generando así varias propuestas por cada localización por la que se desliza la CNN. El segundo módulo es el módulo detector de Fast R-CNN, el cual redimensiona las *bounding boxes* y predice la clase con una capa *softmax*.

Como CNN para generar la *convolution feature* map (llamada *backbone*), se pueden utilizar multitud de CNNs compatibles, como es el caso de las redes ResNet 101 o ResNet 51, utilizadas normalmente. De esta manera, el coste computacional y temporal del algoritmo varía en función del *backbone* escogido, así como su resultado final y la precisión de este. En cuanto a la precisión, con el modelo VGG-16 como *backbone*, el valor de la métrica bbox mAP (mean Average Precision) sobre el conjunto de test de COCO (*Common Objects in Context*) es de 21,9, tal y como se puede ver en la Figura 3.9. Sin embargo, en las comparativas del artículo del modelo posterior Mask R-CNN, el bbox AP con el *backbone* Inception-ResNet-v2-TDM es de 36,8 (ver Figura 3.10).

method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

Figura 3.9: mean Average Precision (mAP) del modelo Faster R-CNN. Extraído de [7].

3.3.2 Mask R-CNN

Este modelo, también desarrollado por FAIR, tiene como principal objetivo resolver el problema de *instance segmentation*; no obstante, junto con la segmentación, también se devuelve la bbox asociada al objeto segmentado. Mask R-CNN es la evolución de Faster R-CNN, adaptando el mismo funcionamiento en dos fases. La primera parte, encargada de generar el *convolution feature map* y las propuestas de regiones de interés, es idéntica a la de Faster R-CNN; en cambio, en la segunda fase se añade como salida adicional la máscara binaria de cada región de interés. Esta salida se genera en paralelo a las *bounding boxes* y su clasificación, añadiendo un incremento en el coste temporal de las detecciones de alrededor del 20% respecto al modelo Faster R-CNN. La máscara binaria, al ser generada para cada región de interés, a diferencia de otros modelos de segmentación recientes que generan diferentes máscaras por clase (*semantic segmentation*), permite resolver el problema de *instance segmentation*.

En cuanto a la fase de detección y generación de bboxes, se introducen pequeñas mejoras que posibilitan la obtención de una precisión (bbox AP) sobre el conjunto de datos de COCO de 39,8, mejorando los resultados de Faster R-CNN, como se puede ver en la Figura 3.10. Respecto a la precisión de las máscaras generadas, Mask R-CNN tiene un mask AP de 37,1, frente al 33,6 de FCIS+++ +OHEIM, la arquitectura ganadora del reto de segmentación COCO 2016. Los resultados completos se pueden ver en la Figura 3.11.

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

Figura 3.10: bbox AP (Average Precision) del modelo Mask R-CNN. Extraído de [8].

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Figura 3.11: mask AP del modelo Mask-RCNN. Extraído de [8].

3.3.3 YOLOv5

Este modelo, desarrollado en 2021, cuya primera versión es del año 2016, destaca por su rapidez de procesamiento de imágenes, siendo capaz de procesar vídeos en tiempo real. A diferencia de los modelos anteriores, el funcionamiento de YOLO (You Only Look Once) está compuesto por una única fase y la imagen de entrada tan solo se evalúa una única vez, mientras que en modelos de la familia R-CNN la imagen se evalúa cientos o miles de veces durante el proceso de detección completo. Gracias a esto se puede conseguir una alta velocidad de procesamiento.

El proceso de detección del algoritmo YOLO es el siguiente. En primer lugar, la imagen se divide en una rejilla. Si el centro de un objeto está situado en una celda de la rejilla, dicha celda se encarga de detectar ese objeto. Cada celda genera un número determinado de *bounding boxes* y sus valores de confianza. Estos valores de confianza indican cómo de segura está el modelo de que la bbox en cuestión contiene un objeto y cómo de precisa es dicha bbox. De manera paralela, cada celda predice la probabilidad condicionada para cada clase. Este valor indica la probabilidad de que haya un objeto de dicha clase, en el caso de que haya un objeto en esa celda. Finalmente, se combinan las *bounding boxes* y sus confianzas junto a las probabilidades anteriores y se descartan resultados duplicados o cuyo producto de valores de confianza y probabilidad sea menor a un umbral, obteniendo las detecciones finales. En la Figura 3.12 se ilustra un ejemplo de detección.

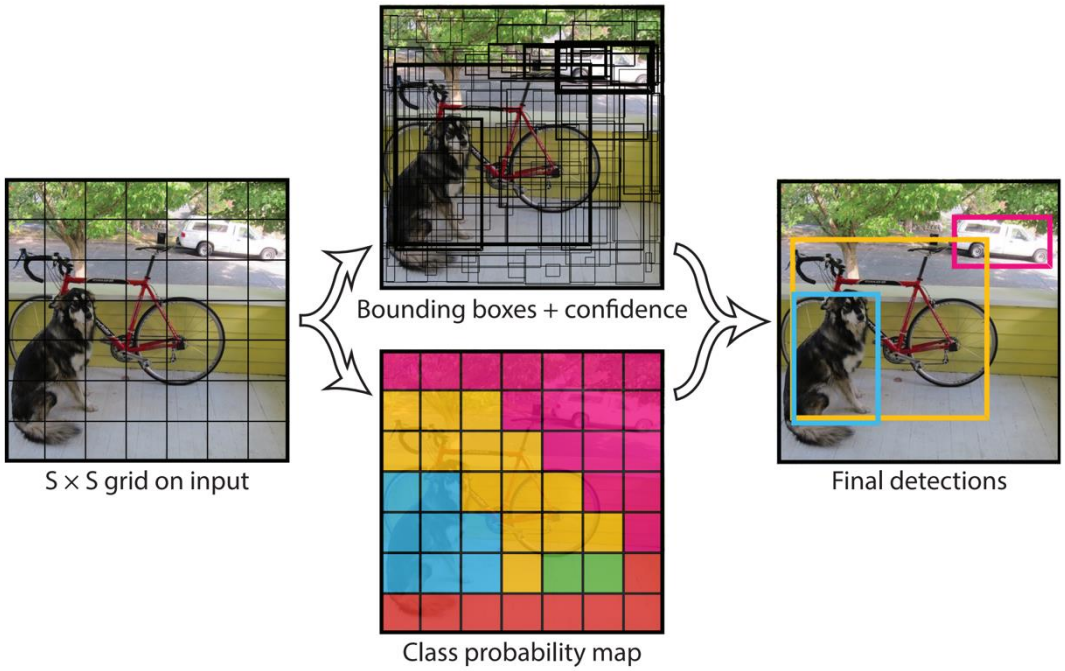


Figura 3.12: proceso de detección de YOLO. Extraído de [10].

En cuanto a la precisión de YOLO, como se puede ver en la Figura 3.13, el modelo original tiene un bbox mAP sobre el conjunto de test del conjunto de datos PASCAL VOC 2007 de 63,4[1], siendo menos preciso que el modelo Faster R-CNN. No obstante, YOLOv5 tiene una bbox AP sobre el conjunto de test de COCO de 54,4 con el modelo YOLOv5x6 TTA [23].

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Figura 3.13: bbox mAP del modelo YOLO. Extraído de [10].

[1] En el artículo de YOLO no se documentan datos acerca de la precisión sobre el conjunto de datos COCO

3.3.4 PointRend

PointRend es un módulo desarrollado por FAIR en 2020 y compatible con modelos de segmentación tales como Mask R-CNN, el cual detecta los puntos "inciertos" en los que se requiere una mayor definición (bordes tales como los dedos de una mano, o las aletas de un pez) y refina la segmentación en dichos puntos mediante interpolación bilineal y un perceptrón multicapa. Esto permite obtener segmentaciones con una resolución de 224x224 píxeles, mucho mayor a la obtenida con Mask-RCNN, pues esta genera máscaras de 28x28 o 56x56 píxeles, suponiendo un gran incremento en el coste computacional aumentar la resolución de la máscara. En la Figura 3.14 se comparan las máscaras generadas por estos dos modelos en una imagen del conjunto de datos COCO.

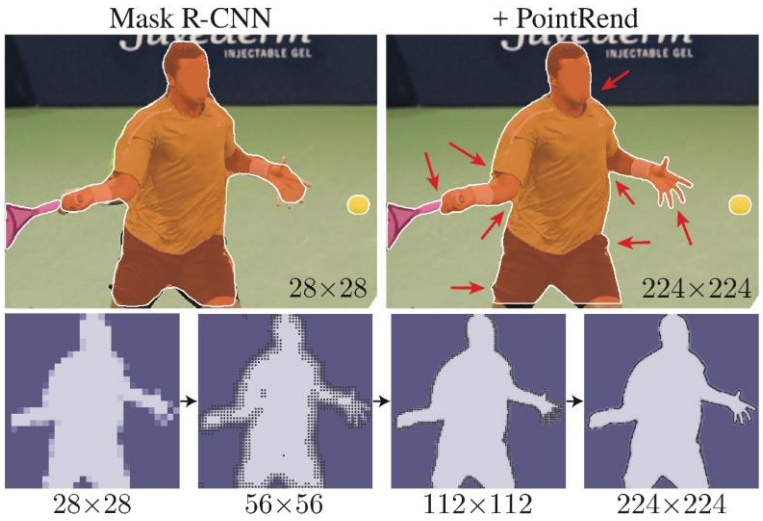


Figura 3.14: comparativa de las máscaras generadas por los modelos s Mask R-CNN y PointRend. Extraído de [9].

El proceso de inferencia de PointRend está compuesto por tres módulos, ilustrados en la Figura 3.15. Partiendo de una rejilla donde se sitúa la imagen, el primer módulo selecciona una serie de puntos donde realizar predicciones, sobre los cuales el siguiente módulo extrae las características (*point-wise feature representations*) y se computan mediante interpolación bilineal con los cuatro vecinos más cercanos de la rejilla, pudiendo predecir una segmentación de mayor resolución. Finalmente, para cada punto se pasa un perceptrón multicapa (MLP) que hace la predicción final.

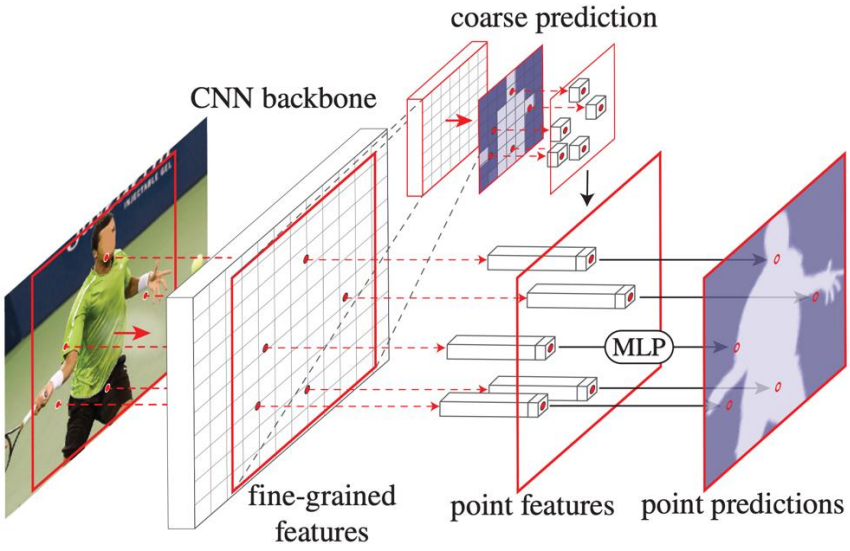


Figura 3.15: partiendo de una rejilla sobre la imagen, el modelo PointRed realiza el proceso de segmentación. Extraído de [9].

En cuanto a la precisión del modelo, utilizando la métrica *mask* AP sobre el conjunto de test de COCO con anotaciones LVIS (de mayor resolución), Mask R-CNN con el modelo ResNet-50-FPN como *backbone*, utilizando una resolución de máscaras de 28x28 píxeles, obtiene un valor de 37,6. Añadiéndole el módulo PointRend y generando máscaras de 224x224 píxeles de resolución, se alcanza un valor de *mask* AP de 39,7 (Figura 3.16).

mask head	output resolution	COCO AP	COCO AP*	Cityscapes AP
4× conv	28×28	35.2	37.6	33.0
PointRend	28×28	36.1 (+0.9)	39.2 (+1.6)	35.5 (+2.5)
PointRend	224×224	**36.3** (+1.1)	**39.7** (+2.1)	**35.8** (+2.8)

Figura 3.16: mask AP del modelo PointRend. Extraído de [9].

3.4 Herramientas de validación

Para las tareas de detección de objetos y segmentación, es necesario el uso de métricas que se adapten al problema e indiquen cómo de precisas son las predicciones. Como solución, existen varias métricas frecuentemente usadas, como *mean* IoU (*Intersection over Union*) o la precisión media (AP, *Average Precisión*) y el *recall* medio (AR, *Average Recall*), utilizados en los retos PASCAL VOC y COCO. En este trabajo se utilizarán como métricas estas últimas por dos motivos. En primer lugar, AP es la métrica más común en las publicaciones actuales de detección y segmentación de objetos. Por otra parte, el código de los modelos utilizados tiene métodos implementados para computar estas métricas automáticamente. Para poder detallar el cálculo del AP, es necesario entender los siguientes conceptos:

1. *Intersection over union* (IoU), o índice de Jaccard: este índice indica el área de solapamiento entre la bbox (o segmentación) de la predicción bb_p y la bbox del *ground truth* bb_{gt}. Se calcula dividiendo la intersección de ambas áreas entre la unión de estas.

$$IoU = \frac{\text{area}\ (bb_p \cap bb_{gt})}{area\ (bb_p\ \cup bb_{gt})}$$

   De esta manera, un solapamiento perfecto tendrá una IoU de uno (100% de solape o coincidencia), mientras que una predicción sin área que se solape con el *ground truth* tendrá una IoU de cero (0% de solape). La Figura 3.17 trata de aclarar este concepto con un ejemplo visual, donde el rectángulo verde representa la etiqueta o *ground truth* bb_{th} y el rectángulo azul corresponde a la bbox de la detección o predicción bb_p.

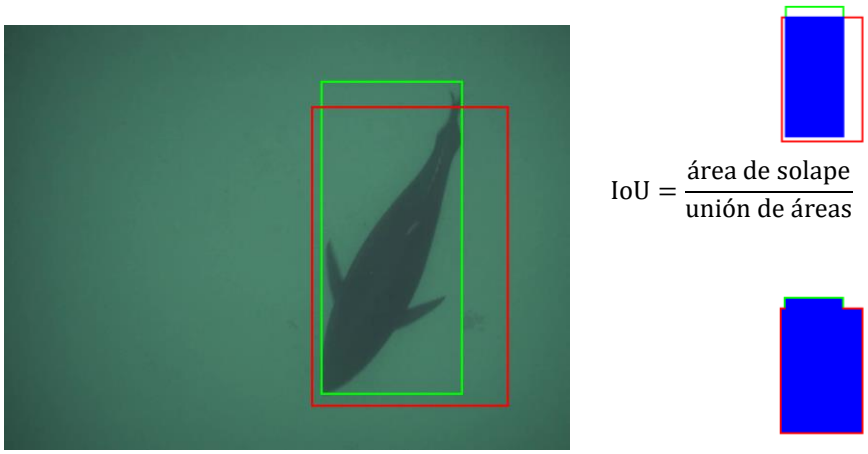


Figura 3.17: ejemplo de IoU, el rectángulo verde corresponde al bbox del ground truth bb_{th} y el rectángulo azul a la bbox de la predicción bb_p.

2. Verdaderos positivos (TP, *True Positive*), falsos positivos (FP, *False Positive*) y falsos negativos (FN, *False Negative*): partiendo de la métrica anterior, en un problema de detección o segmentación podría ser adecuado definir un umbral de IoU por encima del cual la predicción se corresponde con un verdadero positivo y por debajo del cual la predicción se considera un falso positivo. De esta manera, dado un conjunto de predicciones con sus correspondientes etiquetas, se pueden definir los verdaderos positivos y falsos positivos para cualquier umbral de IoU. Los falsos negativos se dan cuando no hay predicción, pero sí hay *ground truth*.

3. Precisión: esta métrica indica la proporción de verdaderos positivos de todas las etiquetas, y se calcula dividiendo los verdaderos positivos entre la suma de verdaderos positivos y falsos positivos.

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

4. Exhaustividad o recuperación (*recall*): esta métrica indica la proporción de verdaderos positivos de las predicciones, calculado dividiendo los verdaderos positivos entre la suma de verdaderos positivos y falsos negativos.

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

Así, el AP se calcula de la siguiente manera:

1. Para un umbral de IoU, se computa la curva de precisión-recuperación (*PR curve*) con la precisión decreciendo monótonamente, determinando la precisión para un *recall r* como la mayor precisión obtenida para cualquier *recall r'* ≥ *r*. La Figura 3.18 muestra diferentes curvas de precisión-*recall* de ejemplo para una serie de modelos, cada uno con un poder predictivo distinto. El color morado representa un modelo de clasificador perfecto. Las curvas más alejadas de esa esquina superior derecha representan a modelo con peores predicciones. Un modelo aleatorio sin entrenar estaría representado como una línea horizontal a media altura (0.5).

2. Se computa el AP como el área bajo esta curva, mediante integración numérica.

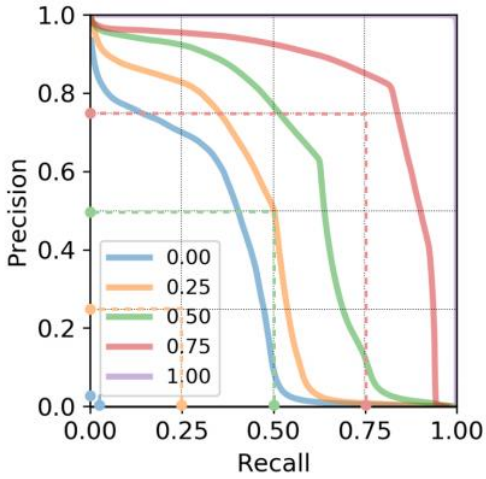


Figura 3.18: ejemplo de curva de precisión-recuperación. La imagen muestra la curva para una serie de modelos, cada uno con un poder predictivo distinto. El color morado representa un modelo de clasificador perfecto. Extraído de [24].

En las APIs utilizadas se computan varios valores de AP para diferentes umbrales de IoU. En el caso de utilizar varios, como es en el caso de la métrica primaria de COCO, se computa el AP para umbrales de 0,50, 0,55, …, 0,95 y se calcula la media aritmética.

En cuanto al AR, este se calcula para el intervalo de IoU entre 0,5 y uno de la siguiente manera, partiendo del solapamiento o entre una anotación gt_i y la segmentación o detección más cercana:

$$\text{AR} = 2 \int_{0,5}^{1} \text{recall}(o) = \frac{2}{\text{n}} \sum_{\text{i}=1}^{\text{n}} \max\left(\text{IoU}(gt_i) - 0{,}5, 0\right)$$

Esta métrica, de acuerdo con [25], premia las detecciones y segmentaciones con un *recall* alto y una buena localización.

En este trabajo, para poder realizar la evaluación de los modelos utilizados, se medirá el AP para varios umbrales de IoU y el AR en un conjunto de 110 imágenes etiquetadas manualmente (con bbox para CNNs de detección y contorno para CNNs de segmentación). Además, para poder comprobar si realmente esos resultados mejoran la calidad o el número de resultados obtenidos en todo el procedimiento llevado a cabo en el proyecto, se incorporará el uso de cada modelo a este procedimiento y se procesarán varios vídeos de diez minutos, midiendo el número de peces (tanto únicos como repetidos a lo largo del vídeo) para cada vídeo y cada modelo.

De esta manera, la solución propuesta consiste en experimentar con diferentes modelos de DL, automatizando completamente el proceso de segmentación del proyecto BIACOP y mejorando sus prestaciones. Se propone etiquetar manualmente las imágenes del conjunto de datos con el *software* VIA y explorar el uso de diferentes modelos de detección y segmentación de objetos en imágenes. Estos modelos están implementados en *frameworks* de Python como TensorFlow, Keras o PyTorch, y se propone evaluarlos tanto con las métricas AP y AR como con el número de peces detectados y el tiempo de procesamiento (segmentación, ajuste de modelo y estimación de medidas) de varios vídeos.

3.5 Resumen de la solución propuesta

Este trabajo fin de grado se centra en las técnicas de VxC y DL que automaticen completamente el proceso de segmentación, consiguiendo segmentar el mayor número de peces en cada *frame* y, por ende, en cada vídeo. Por ello, este capítulo se dedica a exponer los modelos propuestos para la detección de objetos en imágenes (Faster R-CNN, Mask R-CNN y YOLOv5), y para la segmentación de objetos en imágenes (Mask R-CNN y PointRend). También se expone el procedimiento propuesto para la confección del conjunto de datos con el que entrenar cada modelo, además de las métricas que se adapten al problema para que indiquen cómo de precisas son las predicciones realizadas por dichos modelos en las tareas de segmentación y detección de objetos.

4. Desarrollo de la solución propuesta

En este capítulo se detallan los procesos y los pasos seguidos durante el desarrollo de la solución propuesta. En primer lugar, se comenta la elaboración del conjunto de datos utilizado, el cual parte del proceso de adquisición de imágenes documentado en el capítulo anterior. Finalmente, se puntualizarán las pruebas realizadas con cada uno de los modelos utilizados, mostrando además los resultados obtenidos con cada modelo.

4.1 Creación del conjunto de datos

A partir de los vídeos grabados en condiciones reales con peces nadando libremente, detallado en la sección 2.2.1 de este documento, se confecionó un pequeño *ground truth* con un conjunto inicial de 120 imágenes de vista dorsal y ventral de Mazarrón. Estas imágenes fueron etiquetadas manualmente con la ayuda del *software* VGG Image Annotator (VIA) y se aumentó posteriormente con 60 imágenes de vista lateral, tanto de Balfegó como de Mazarrón. Este pequeño *ground truth* que denominaremos GT-180, tenía por objetivo el disponer de imágenes etiquetas para realizar una primera valoración de los modelos propuestos.

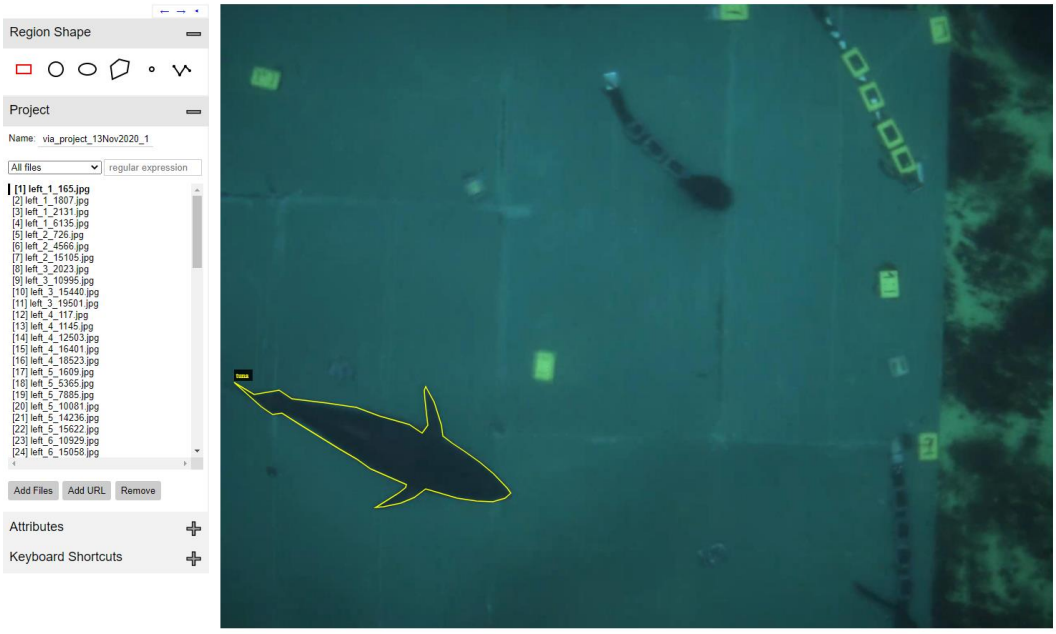


Figura 4.1: interfaz del software VGG Image Annotator.La línea amarilla corresponde al etiquetado manual realizado por el autor de este trabajo, para disponer de un conjunto de imágenes inicial (GT-180) etiquetado con el que comenzar a valorar modelos. La parte de la izquierda, corresponde a una ventana que permite etiquetar las imágenes, especificar la clase de cada objeto anotado y poder navegar por las diferentes imágenes del conjunto.

En concreto, GT-180 se utilizó para que entrenar Mask R-CNN en su versión de TensorFlow, siguiendo la entrada en el blog de Matterport Engineering Techblog [26]. La Figura 4.1 muestra el funcionamiento de VIA. La línea amarilla que se muestra en la silueta del atún se corresponde con el etiquetado manual realizado por el autor de este trabajo. En el lado izquierdo de la Figura 4.1 se muestra una ventana de dicha aplicación que corresponde con una ventana con cuatro apartados. El primero son las herramientas para etiquetar, según sea la forma de los objetos a anotar (la línea de la imagen se ha dibujado con la cuarta herramienta, para polígonos); el segundo es una lista con las imágenes del conjunto a etiquetar; la tercera, los atributos de las etiquetas (aquí se especifica la clase de los objetos etiquetados); por último, el cuarto apartado informa de los atajos de teclado de VIA. VIA es un software intuitivo, que genera como resultado las anotaciones en un archivo con formato JSON. Estas anotaciones contienen atributos como el nombre de la imagen, la segmentación en coordenadas *x* e *y* de cada punto del contorno y la clase a la que pertenece dicha segmentación. Un ejemplo del archivo generado al anotar la imagen de la Figura 4.1 se puede observar en la Figura 4.2:

```
{
    "left_1_165.jpg85852": {
        "filename": "left_1_165.jpg",
        "size": 85852,
        "regions": [
            {
                "shape_attributes": {
                    "name": "polygon",
                    "all_points_x": [
                        33,
                        83,
                        ...,
                        100
                    ],
                    "all_points_y": [
                        931,
                        961,
                        ...,
                        991
                    ]
                },
                "region_attributes": {
                    "name": "tuna"
                }
            }
        ],
        "file_attributes": {}
    }
}
```

Figura 4.2: ejemplo de JSON generado por el software VIA.

Una vez entrenado el modelo Mask R-CNN, realizada una primera valoración y comprobado su potencial, se procedió a la confección de un *ground truth* de mayor tamaño que denominaremos GT-1000. En concreto, para confeccionar el GT-1000, se optó por trabajar con el procedimiento desarrollado en proyecto de investigación BIACOP, para poder generar así un *ground truth* de mayor de manera semiautomática. El procedimiento seguido confeccionar el GT-1000 fue el siguiente:

1. Se extrajeron un total de 1000 imágenes o *frames* manualmente de varios vídeos. Estos vídeos corresponden a grabaciones de vista dorsal y ventral de los peces en las instalaciones de Mazarrón e incluyen grabaciones con cuatro fondos diferentes, como se pueden ver en la Figura 4.3 de izquierda a derecha y de arriba a abajo: fondo del tanque sucio, fondo del tanque con una pantalla artificial, fondo del tanque limpio y techo de las instalaciones de vista ventral. De cada fondo se extrajeron un total 250 imágenes; cada una de ellas con un pez o más nadando libremente.

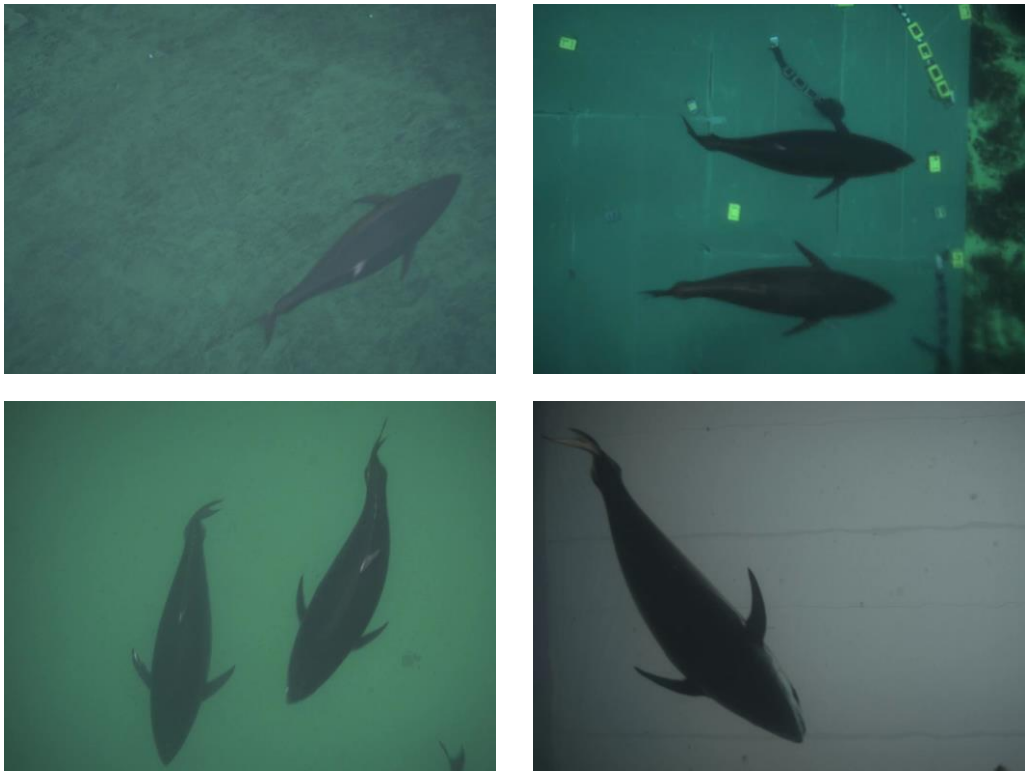


Figura 4.3: ejemplos de imagen extraídas de diferentes vídeos para confeccionar el GT-1000. De izquierda a derecha y de arriba abajo se muestra frame con: fondo del tanque sucio, fondo del tanque con una pantalla artificial, fondo del tanque limpio y techo de las instalaciones de vista ventral.

2. Con el modelo Mask R-CNN entrenado con el conjunto de datos inicial, GT-180, se ejecutó la detección sobre las imágenes del GT-1000, guardando como resultado las *bounding boxes* de las predicciones para cada imagen.

3. Se segmentaron los peces del GT-1000 automáticamente con las técnicas de segmentación convencionales del proyecto de investigación BIACOP. En concreto se

trabajó con la técnica de *clustering* k-means, aplicada sobre los bboxes del GT-1000 proporcionados por Mask_R-CNN del paso anterior, ya que con la ayuda de las bboxes el método de *clustering* resulta mucho más preciso y comete menos fallos. Finalmente se generó un archivo JSON similar al generado por VGG Image Annotator con las segmentaciones exitosas del método tradicional. Todas las segmentaciones fueron revisadas visualmente y las segmentaciones fallidas se anotaron manualmente.

4. Una vez confeccionado el GT-1000 con sus correspondientes anotaciones, se confeccionó para cada fondo un conjunto de entrenamiento y un conjunto de validación. Así, las 250 imágenes de cada fondo fueron dividas en 200 para el conjunto de entrenamiento y 50 para el conjunto de validación.

De esta forma, GT-1000, generado con un conjunto de 1000 imágenes, fue etiquetado de con un procedimiento mucho más rápido que si se hubiese anotado cada imagen de manera manual. Finalmente, se generó un conjunto de test final de 110 imágenes etiquetadas manualmente, para así poder comparar los resultados de cada modelo. La distribución final del número de imágenes por fondo para cada subconjunto se puede consultar en la Figura 4.4.

Tipos de Fondo	**Conjunto de Entrenamiento**	**Conjunto de Validación**	**Conjunto de Test**
Sucio	200	50	25
Pantalla	200	50	25
Limpio	200	50	35
Ventral	200	50	25
	800 imágenes	200 imágenes	110 imágenes

Figura 4.4: número de imágenes del GT-1000 de cada fondo utilizada para los conjuntos de entreanmiento, validadción y test.

4.2 Experimentos y resultados

En este apartado se exponen los detalles de los experimentos realizados con cada modelo, junto con los resultados. La descripción se realiza en el orden cronológico seguido durante el desarrollo de este trabajo, es decir: Mask R-CNN en TensorFlow para segmentación, YOLOv5, Faster R-CNN, Mask R-CNN en PyTorch para segmentación y detección y PointRend. De esta manera, se comenta el procedimiento seguido y se justifican los experimentos realizados con cada modelo.

4.2.1 Mask R-CNN en TensorFlow

Como prueba inicial para estudiar la viabilidad del modelo Mask R-CNN y así resolver directamente el problema de la segmentación, se probó el uso de la implementación de este modelo en TensorFlow y Keras presentada por W. Abdulla [27]. Pese a no ser la implementación oficial, la cual está implementada en PyTorch, se eligió por varios motivos. En primer lugar, TensorFlow y Keras son dos *frameworks* con los que se tiene una mayor experiencia, siendo así más fácil encontrar, entender y resolver las dificultades encontradas a lo largo del experimento. En segundo lugar, esta implementación tiene un mayor apoyo por parte de la comunidad de usuarios en el área del DL, con multitud de tutoriales y guías sobre cómo utilizar el modelo con conjuntos de datos nuevos, además de contar con alrededor de 1600 entradas en el apartado *issues* del repositorio de GitHub. En estas entradas se comentan posibles complicaciones y problemas que han surgido al resto de usuarios, junto con sus respectivas soluciones, si las hay.

La implementación de Abdulla utiliza la CNN ResNet 101 como *backbone*, aunque también permite el uso de la CNN ResNet 50. No obstante, se recomienda el uso de ResNet 101 para obtener unos mayores valores de precisión, a costa de un mayor coste de memoria y tiempo. Además, esta implementación incluye los pesos resultantes al entrenar el modelo con el conjunto de datos de COCO, permitiendo así utilizar la técnica de *transfer learning feature extraction*. Otra técnica de *transfer learning* cuyo uso está implementado de manera sencilla es *fine-tuning*, permitiendo entrenar el modelo con un menor coste de memoria, de cálculo y de tiempo cuando se congelan varias de sus capas [22]. Otro método para incrementar la generalización de los modelos entrenados con soporte en esta implementación es *image augmentation*, con la librería imgaug.

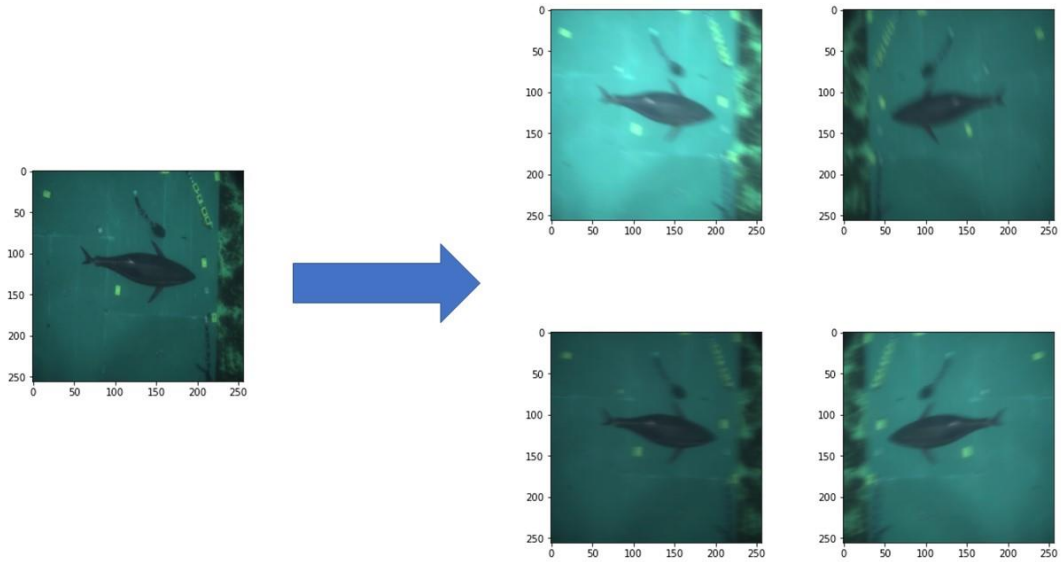


Figura 4.5: ejemplo de transformaciones aplicadas a una imagen del conjunto de datos (image augmentation).

De esta forma, los experimentos realizados con este modelo consistieron en entrenarlo con diversas configuraciones, probando los dos *backbones* soportados ResNet 101 y ResNet 50, utilizando los pesos preentrenados de COCO y probando a utilizar la técnica de *fine-tuning*, además de utilizar la técnica de *image augmentation* para aplicar transformaciones de distorsión de movimiento y cambios en el brillo, el contraste y la orientación de la imagen, tal y como se puede ver en la Figura 4.5.

Los trabajos con este primer modelo de Mask R-CNN con su implementación en TensorFlow formaron parte de una prueba piloto para estudiar la viabilidad de sustituir todo el proceso de segmentación con un único modelo de *deep learning* (DL). La Figura 4.7 muestra el valor de la función de coste *total_loss* sobre el conjunto de entrenamiento y validación a lo largo de un entrenamiento de 100 *epochs*, con 100 *steps_per_epoch*, dos *imgs_per_batch* y un *learning rate* de 0,001. La función *total_loss* es una suma de varias funciones de coste utilizadas durante el entrenamiento de Mask R-CNN. Su cálculo se detalla en [27]. Los hiperparámetros comentados tienen el siguiente significado en esta implementación:

- Una *epoch* se define como una agrupación de *steps*, después de la cual se calcula el valor de la función de coste sobre el conjunto de validación.
- Un *step* consiste en una iteración del algoritmo de *backpropagation* sobre un *batch* de imágenes. El tamaño de este *batch* se calcula como el valor de *imgs_per_batch* * *num_gpus*.

Para clarificar estos conceptos, en la Figura 4.6 se ilustra un ejemplo de entrenamiento del modelo con 30 *epochs* y 100 *steps* por *epoch*. El valor de *imgs_per_batch* depende de la cantidad de memoria VRAM de la GPU donde se ejecuta el entrenamiento y el tamaño de las imágenes con las que se entrena el modelo.

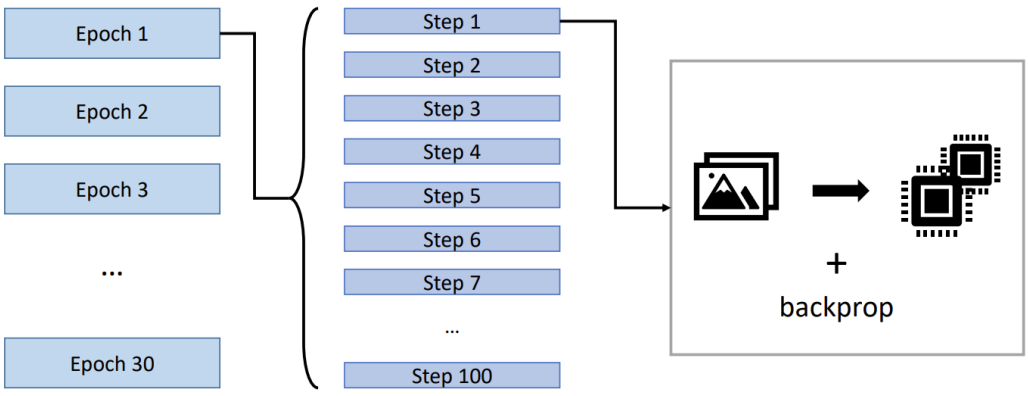


Figura 4.6: ejemplo de entrenamiento del modelo Mask R-CNN en TensorFlow con 30 epochs y 100 steps. Se puede observar que cada epoch está formado por 100 steps, y que cada step computa el algoritmo backprop sobre una serie de imágenes, agrupadas en un batch.

En la Figura 4.7, la línea naranja representa el valor de la función de coste sobre el conjunto de entrenamiento, mientras que la línea azul representa el valor de la función sobre el conjunto de validación. Revisando los valores de la función de coste sobre el conjunto de entrenamiento y validación podemos asegurar que no se produce *overfitting* (sobreentrenamiento). Aunque los valores no terminan de converger, como primera aproximación al problema se considera un entrenamiento válido y se seguirá mejorando el entrenamiento en un posible trabajo futuro.

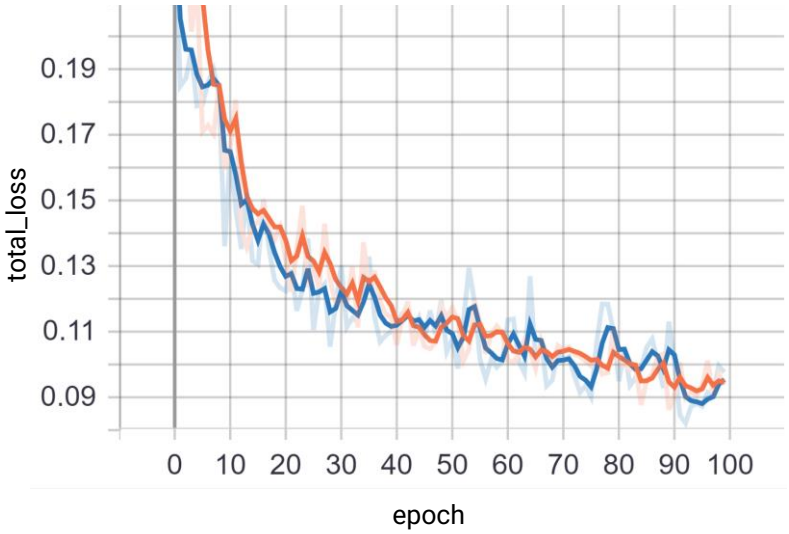


Figura 4.7: valor de la función de coste total_loss durante el entrenamiento de Mask R-CNN en TensorFlow. La línea naranja representa el valor de la función sobre el conjunto de entrenamiento, mientras que la línea azul representa el valor de la función sobre el conjunto de validación.

Los resultados de la inferencia en varias imágenes y vídeos, tras entrenar el modelo durante dos horas, fueron inspeccionados únicamente de forma visual tanto por el autor de este trabajo como por parte del equipo de VxC, Así, en la segmentación de varias imágenes y vídeos con el modelo entrenado a partir del *backbone* ResNet 101, se obtuvieron unos resultados que a simple vista son bastante buenos, como se puede ver en la comparativa de la Figura 4.8.

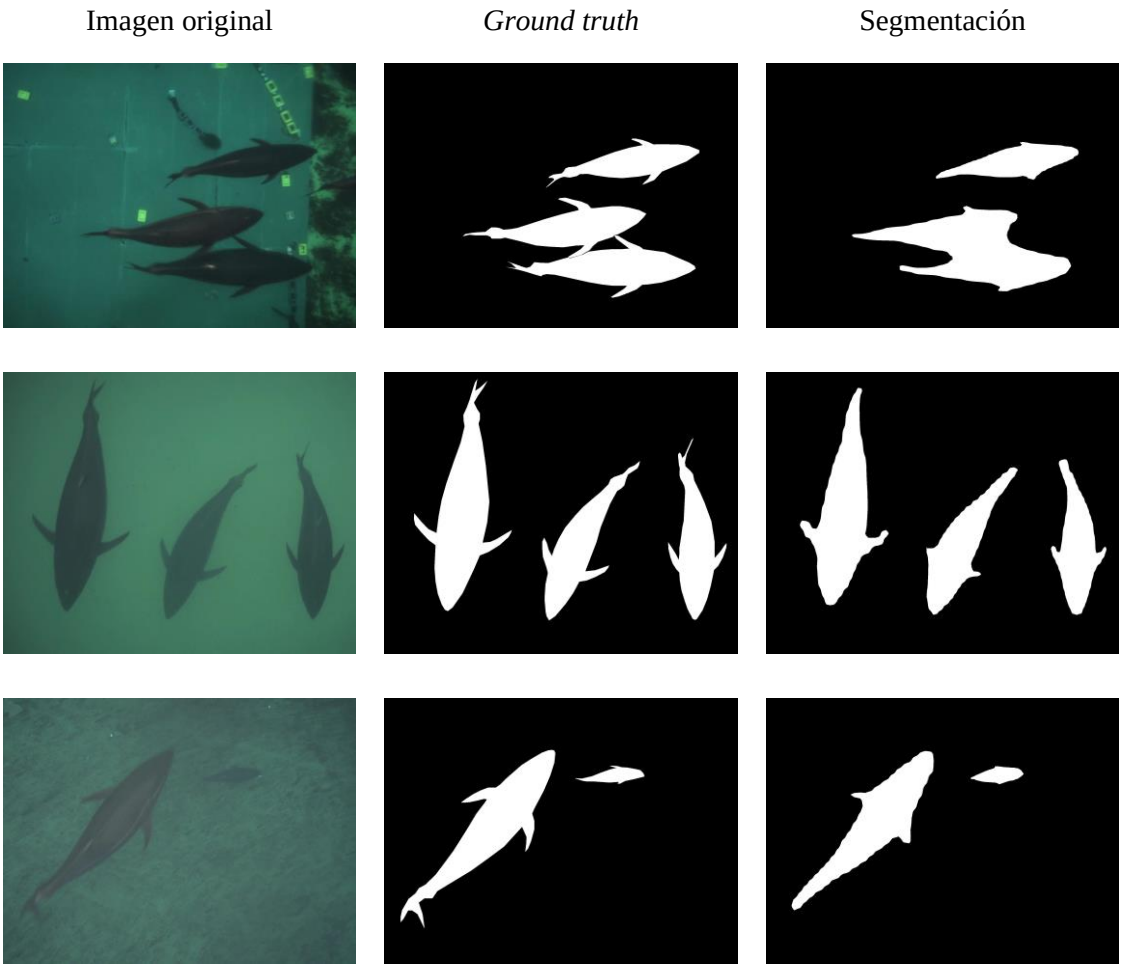


Figura 4.8: segmentaciones de Mask R-CNN en TensorFlow sobre tres imágenes del subconjunto de test (prueba piloto para validar el modelo).

Aun así, la segmentación no es tan precisa como para aplicar directamente el modelo de la silueta del proyecto de investigación y estimar las medidas de longitud y ancho de los peces, puesto que, aparte de no estar en lo suficientemente bien definidos los bordes del pez, la cola y la punta de la cabeza no se suelen segmentar bien. Esto se debe a que las máscaras de segmentación que genera el modelo son de 28x28 píxeles, una resolución demasiado baja para el problema a resolver. Ejecutar el modelo con una máscara mayor incrementa sobremanera el coste computacional del algoritmo, como se comenta en [9], por lo que no es una opción viable.

Por otro lado, Mask R-CNN, junto con la segmentación, también devuelve la bbox del objeto detectado, como puede apreciarse en la imagen de la Figura 4.9.

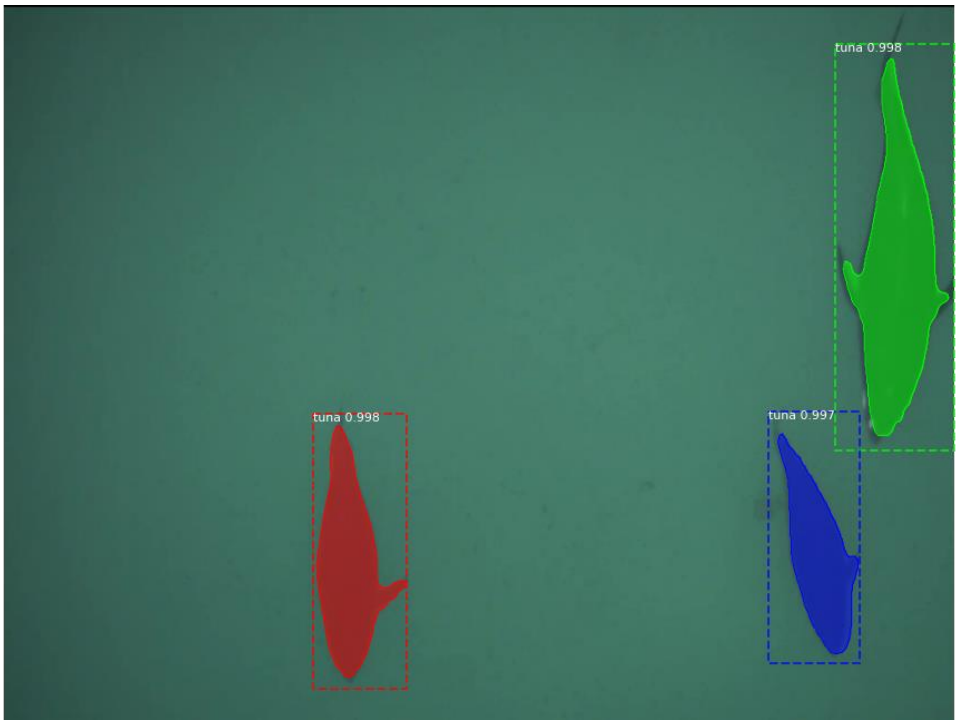


Figura 4.9: resultados de la inferencia del modelo Mask R-CNN. Mask R-CNN devuelve la segmentación, la bbox y el valor de confianza de cada detección.

De esta manera, y a la vista del problema de definición de bordes comentado, surgen dos alternativas:

- Encontrar un método con el que refinar las segmentaciones producidas por Mask R-CNN o un modelo de DL que devuelva segmentaciones de mayor resolución, evitando usar así los métodos convencionales
- Utilizar e incorporar las bboxes como proceso intermedio en el método actual del proyecto BIACOP. De esta manera, se ejecutarían las técnicas convencionales sobre el área de la imagen en las que se sitúan las bboxes proporcionados por el modelo, evitando así problemas de falsos positivos y mejorando el tiempo de ejecución, ya que se trabajaría únicamente en los *frames* de los vídeos con detecciones y sobre un área reducida de la imagen. Por otro lado, la técnica de *local thresholding* utilizada en BIACOP para segmentar dentro de la bboxes, podría ser sustituida por la técnica de *clustering* de k-medias con k=2 *clusters* (pez y fondo). Esta técnica de *clustering* resulta más eficiente a la hora de segmentar en una ventana de cada imagen.

Siguiendo ambas alternativas, se proponen dos modelos adicionales de detección de objetos (YOLO y Faster R-CNN) y un modelo adicional de segmentación, PointRend, el cual refina las segmentaciones producidas por Mask R-CNN.

4.2.2 YOLOv5 en PyTorch

Este modelo [28], cuya implementación está desarrollada usando PyTorch como *framework*, fue propuesto como posible alternativa para utilizar la detección de objetos como paso intermedio en el procedimiento del proyecto, sobre el cual utilizar técnicas tradicionales de segmentación. Al ser un modelo de detección de objetos, las anotaciones del conjunto de datos GT-1000 tuvieron que ser adaptadas al formato de bbox pertinente. En el modelo Mask R-CNN era suficiente con guardar en un directorio para cada conjunto de entrenamiento, validación y test las imágenes junto con el archivo en formato JSON de las anotaciones. Sin embargo, en YOLOv5 se ha de guardar para cada conjunto un directorio con las imágenes y otro directorio para las anotaciones. Cada anotación es un fichero de texto con el mismo nombre que la imagen a la que corresponde y una línea para cada objeto de la imagen, en el formato *clase centro_x centro_y ancho alto*, con un sistema de coordenadas normalizado (de cero a uno desde la esquina superior izquierda). Así, como el conjunto de datos de este trabajo tan solo tiene una clase, los atunes se corresponden con la clase cero.

Por ejemplo, las anotaciones de la Figura 4.10 (imagen con nombre “left_24_6026.jpg”) en el formato requerido por el modelo deberían estar en un archivo llamado “left_24_6026.txt” con el siguiente contenido:

0 0.464111328125 0.3199869791666667 0.43017578125 0.224609375

0 0.234130859375 0.392578125 0.35107421875 0.11328125

0 0.193115234375 0.6028645833333334 0.38623046875 0.26171875

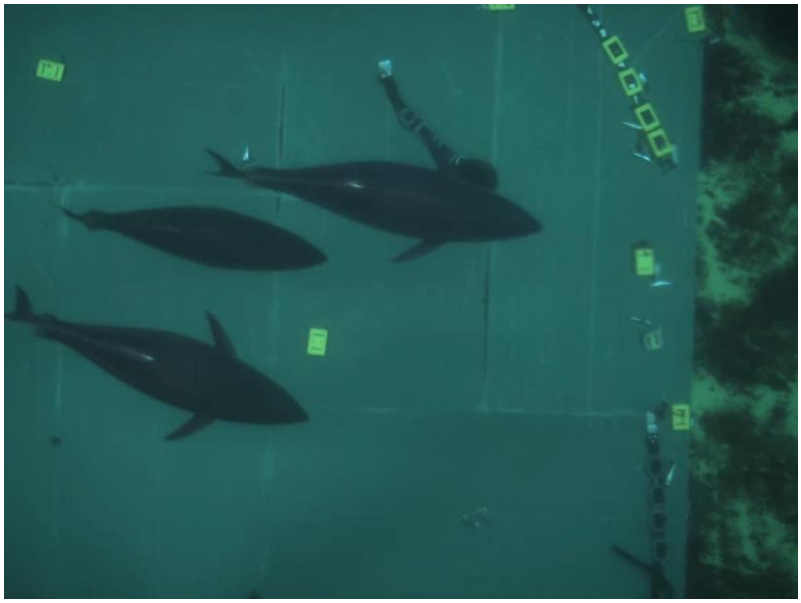


Figura 4.10: imagen "left_24_6026.jpg".

Por otra parte, YOLOv5 permite trabajar con varios modelos dentro de la misma arquitectura, dependiendo del número de parámetros a entrenar. La tabla de la Figura 4.11 muestra dichos modelos y su número de parámetros de entrenamiento [28].

Modelo	Número de parámetros (millones)
YOLOv5s	7,3
YOLOv5m	21,4
YOLOv5l	47,0
YOLOv5x	87,7

Figura 4.11: modelos disponibles en YOLOv5 junto al número de parámetros de cada uno.

Por lo general, un modelo con un mayor número de parámetros de entrenamiento requiere de una mayor cantidad de cálculos, memoria y tiempo de cómputo para entrenar y realizar las detecciones, a cambio de ser capaz de resolver problemas más complicados y devolver resultados más precisos. De esta manera, se decidió entrenar los cuatro modelos de la Figura 4.11 durante 50 *epochs* partiendo de los pesos entrenados del conjunto de datos de COCO, con el resto de hiperparámetros por defecto del modelo. En esta implementación de YOLO, el modelo se entrena durante una serie de *epochs*, los cuales consisten en pasadas completas por el conjunto de entrenamiento. Entre cada *epoch* se computa el valor de varias métricas sobre el conjunto de validación. En la Figura 4.12, se muestran los resultados de la métrica *Objectness* sobre el conjunto de entrenamiento y validación, además del valor de la métrica mAP para el conjunto de validación, durante las diferentes *epochs* del entrenamiento del modelo YOLOv5x. El cálculo de las diferentes métricas utilizadas por YOLOv5 se detalla en el artículo [29]. El conjunto de resultados y métricas de los entrenamientos de los diferentes modelos de YOLOv5 se pueden consultar en el apartado de Anexos.

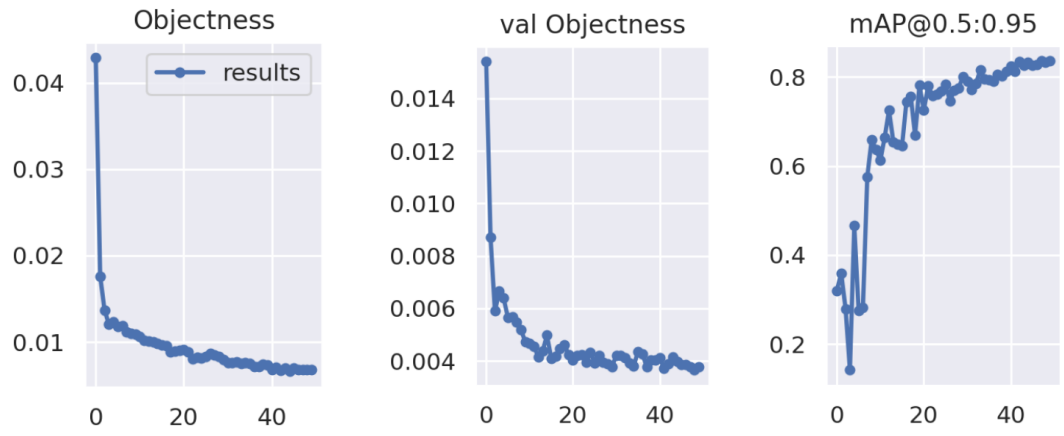


Figura 4.12: valor de diferentes métricas durante el entrenamiento de YOLOv5x. El eje horizontal indica la epoch. Izquierda: Objectness sobre el conjunto de entrenamiento; medio: Objectness sobre el conjunto de validación; derecha: mAP sobre el conjunto de validación.

Finalmente, se midió la precisión media (AP) de cada modelo entrenado sobre el conjunto de test de GT-1000. Así, se pudieron comparar los resultados y valorar si el incremento en el tiempo computacional era realmente una mejor opción teniendo en cuenta el aumento de precisión. Los resultados obtenidos se pueden consultar en la Figura 4.13.

Modelo	Tiempo de entrenamiento (minutos)	Tiempo medio de inferencia (ms/imagen)	AP
YOLOv5s	17	1,3	0,776
YOLOv5m	35	2,6	0,835
YOLOv5l	60	3,9	0,839
YOLOv5x	**113**	**7,6**	**0,850**

Figura 4.13: tabla de resultados de YOLOv5, donde se aprecia el incremento de tiempo computacional, para cada modelo, respecto al aumento de AP, Entrenamiento realizado con 50 epochs y GT-1000.

Tal y como era de esperar, los modelos con más parámetros a entrenar consumen un mayor tiempo en entrenar la misma cantidad de *epochs* y tardan más en ejecutar las detecciones, aunque estas son tan rápidas que pueden ejecutarse en tiempo real. Por ejemplo, con el tiempo de inferencia del modelo más costoso, se podrían procesar vídeos de hasta 130 *frames* por segundo a tiempo real. Además, la precisión aumenta de manera considerable con el número de parámetros, por lo que según muestran los resultados el modelo YOLOv5x es el más adecuado para el problema que nos ocupa.

4.2.3 Faster R-CNN en Pytorch

Este modelo es uno de los propuestos para el paso intermedio de detección de objetos, sobre los cuales segmentar con métodos tradicionales. Faster R-CNN ha sido desarrollado por FAIR en PyTorch, y forma parte del *framework* de detección y segmentación de objetos Detectron2 [30]. Detectron2 permite registrar conjuntos de datos para utilizarlos posteriormente con cualquier modelo del *framework*, guardando en las anotaciones de segmentación de objetos tanto la segmentación como las *bounding boxes* de cada instancia. Además, en tutorial de Google Colab de Detectron2 muestran cómo registrar un conjunto de datos como el de la entrada del blog de Matterport del modelo de Mask R-CNN en TensorFlow y Keras. De esta forma, el traslado de las anotaciones utilizadas en el modelo Mask R-CNN en TensorFlow al formato requerido por Detectron2 y su uso en un problema de detección de objetos, como es el que resuelve el modelo Faster R-CNN, es realmente sencillo.

Así, se entrenó el modelo Faster R-CNN con la CNN ResNet 50 como *backbone* (utilizado en el tutorial) y a partir de los pesos entrenados de COCO durante 5000 iteraciones con un *batch size* de 16 imágenes (alrededor de dos horas) y un *learning rate* de 0,00025. Estos son los principales hiperparámetros utilizados en Detectron2, cuyos entrenamientos consisten en una serie de iteraciones del algoritmo de *backpropagation* sobre un *batch* de *batch_size* * *num_gpus*

imágenes. Como se ve, el concepto de iteración en Detectron2 es semejante a los *steps* de Mask R-CNN en TensorFlow, los cuales se agrupan en *epochs*. No obstante, Detectron2 no utiliza el conjunto de validación durante el entrenamiento, por lo que agrupar las iteraciones no tiene sentido. Por otra parte, al no calcular el valor de la función de coste durante el entrenamiento de forma nativa, no hay manera de poder identificar si se produce *overfitting* de manera sencilla, por lo que en este trabajo no se utiliza el conjunto de validación en los modelos pertenecientes al *framework* Detectron2. De todas formas, se estudiará la implementación de una solución en trabajos futuros. El valor de la función de coste *total_loss* para el conjunto de entrenamiento durante el entrenamiento del modelo se puede consultar en la Figura 4.14. El cálculo de esta función se detalla en el artículo [7].

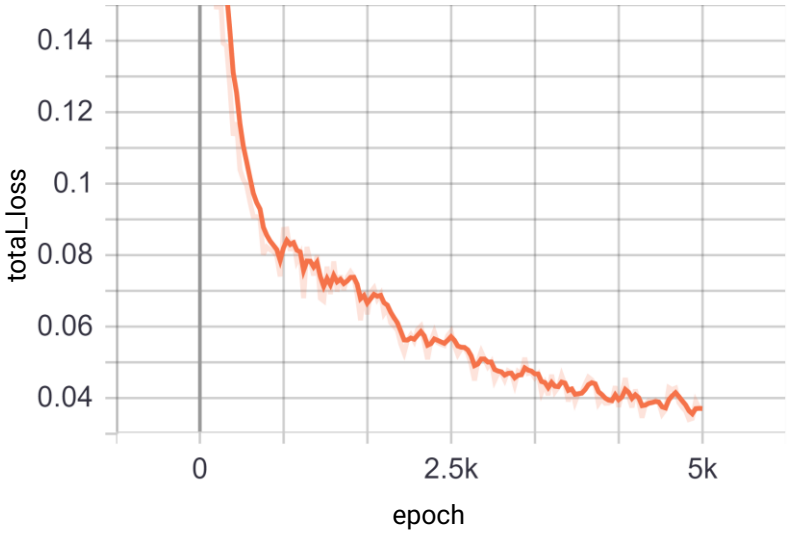


Figura 4.14: valor de la función de coste total_loss durante el entrenamiento de Faster R-CNN. La línea naranja representa el valor de la función sobre el conjunto de entrenamiento.

El cálculo de la métrica AP sobre el conjunto de test obtuvo un valor de AP de 0,806. El tiempo medio de inferencia fue de 36,9 milisegundos.

Modelo	**Tiempo medio de inferencia (ms/imagen)**	**AP**
Faster R-CNN	36,9	0,855

Figura 4.15: tabla de resultados del modelo Faster R-CNN (PyTorch).

4.2.1 Mask R-CNN en PyTorch

Este modelo [8], tal y como se comentó en el apartado 4.3.1 de su implementación en TensorFlow, sirve tanto para resolver problemas de segmentación como de detección de objetos. Por esta razón se utiliza en este trabajo, tanto para obtener las *bounding boxes* como operación previa a la segmentación con métodos convencionales, como para segmentar directamente.

Esta implementación, desarrollada por FAIR, forma parte del *framework* Detectron2, por lo que se utiliza en sus experimentos el conjunto de datos que ya había sido adaptado para realizar los experimentos con el modelo Faster R-CNN,

El modelo Mask R-CNN fue entrenado con la CNN ResNet 50 como *backbone*, partiendo de los pesos de COCO durante 5000 iteraciones con un *batch size* de 16 imágenes y un valor de *learning rate* de 0,00025, al igual que el modelo Faster R-CNN (dos horas fue el tiempo necesario para llevar a cabo dicho entrenamiento). El valor de la función de coste *total_loss*, calculada como se indica en [8], se puede consultar en la Figura 4.16:

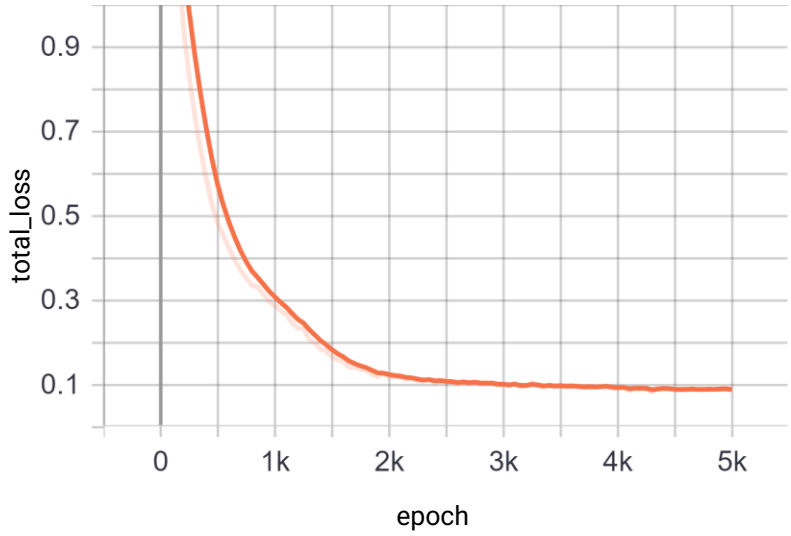


Figura 4.16: valor de la función de coste total_loss durante el entrenamiento de Mask R-CNN en PyTorch. La línea naranja representa el valor de la función sobre el conjunto de entrenamiento.

Como se comenta con anterioridad, de las salidas que produce el modelo nos interesan tanto las *bounding boxes* como la segmentación, por lo que se midió tanto el bbox AP como el mask AP sobre el conjunto de test del ground truth GT-1000. Ambos resultados se muestran en la Figura 4.17.

Modelo	Tiempo medio de inferencia (ms/imagen)	AP^{bb}	AP^{mask}
Mask R-CNN	36,3	0,855	0,856

Figura 4.17: tabla de resultados del modelo Mask R-CNN (PyTorch) obtenidos sobre el conjunto de test del ground truth GT-1000. AP^{bb} corresponde a la AP obtenida con la detección, mientras que AP^{mask} es la de la segmentación.

4.3.4 PointRend en PyTorch

Por último, se experimentó con el modelo PointRend [9], como propuesta alternativa a la segmentación producida con los métodos convencionales. PointRend, desarrollado por FAIR (al igual que Faster R-CNN y Mask R-CNN), está implementado en PyTorch y forma parte del *framework* Detectron2. Por lo tanto, como con Mask R-CNN en PyTorch, el conjunto de datos ya está registrado y listo para entrenar este modelo.

De esta manera, se entrenó el modelo durante 10.000 iteraciones con un *batch size* de ocho imágenes (alrededor de cuatro horas, equivalente a entrenar 5000 iteraciones con un *batch size* de 16, como con Faster R-CNN y Mask R-CNN) a partir de los pesos de COCO. La Figura 4.18 muestra el valor de la función de coste *total_loss*, calculada como se detalla en el artículo [9], sobre el conjunto de entrenamiento durante las diferentes iteraciones:

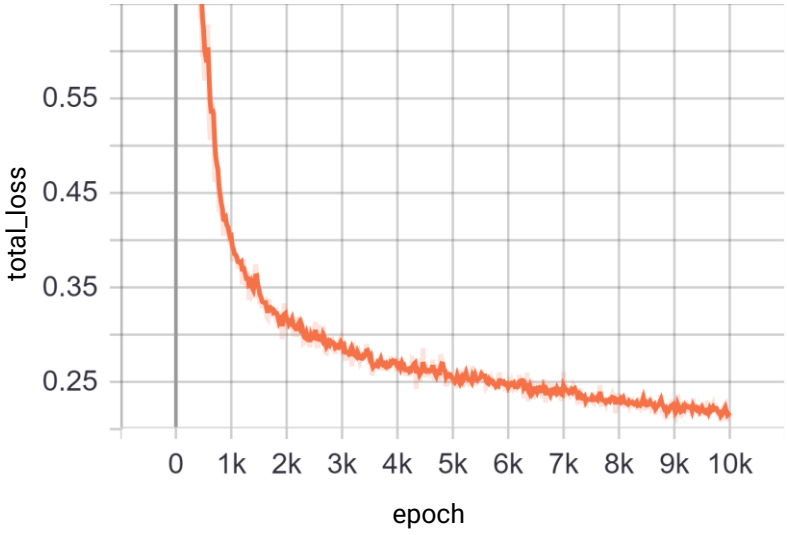


Figura 4.18: valor de la función de coste total_loss durante el entrenamiento de PointRend. La línea naranja representa el valor de la función sobre el conjunto de entrenamiento.

Finalmente, se midió la precisión del modelo; no solo de segmentación, sino también de bbox (este modelo devuelve las mismas salidas que Mask R-CNN). Los resultados se pueden consultar en la Figura 4.19.

Modelo	**Tiempo de inferencia (ms/imagen)**	**AP^{bb}**	**AP^{mask}**
PointRend	41,6	0,762	0,870

Figura 4.19: tabla de resultados del modelo PointRend.

4.3 Análisis de resultados

Una vez entrenados y probados los modelos con el conjunto de test de GT-1000, se pueden comparar los resultados y analizar cuál es el más adecuado para resolver el problema. Se presentan dos tipos de estudios comparativos: desde el punto de vista de la detección y segmentación de objetos y desde el número de peces extraídos.

4.3.1 Resultados para la detección y segmentación de peces

En primer lugar, se analizan los resultados desde el punto de vista de la detección de objetos, y por tanto de la obtención del bbox. Para ello, para cada modelo se obtienen los valores de AP para los umbrales de IoU de 50, 75 y 90 (AP^{bb}_{50}, AP^{bb}_{75}, AP^{bb}_{90}), el valor de AP (AP^{bb}) calculado como la media aritmética del AP de dichos umbrales de IoU (tal y como se expone en el apartado 3.2.4 de este documento). Cuanto mayor es el umbral de IoU, más exigente será la métrica a la hora de considerar un solapamiento de áreas de bbox (bb_p y bb_{gt}) como un verdadero positivo (TP). Además, para cada modelo se obtiene el *average recall* (AR), medido tanto para una detección por imagen (AR^{bb}_1) como para diez detecciones por imagen (AR^{bb}_{10}). Se debe tener en cuenta que 10 detecciones, en este caso, implica todas las detecciones posibles en una imagen o *frame*, ya que en las imágenes del conjunto de test el número máximo de detecciones por imagen es de tres.

Así, con lo valores de dichas métricas con las *bounding boxes*, se deduce que el modelo Mask R-CNN es el más preciso en la tarea de detección; es decir, con el mayor valor en AP^{bb} (Figura 4.20); no obstante, el AR^{bb}_1 y AR^{bb}_{10} de este modelo es inferior al del modelo YOLOv5x. También se observa que, para los AP^{bb}_{50}, AP^{bb}_{75}, obtenidos con los umbrales 50 y 75 de IoU, YOLOv5x es superior a Mask R-CNN; sin embargo, en el AP^{bb}_{90} obtenido con 90 de umbral de IoU, Mask R-CNN supera a YOLOv5x. Otro aspecto a considerar es el tiempo de inferencia, donde YOLOv5x destaca sobre el resto de modelos por su reducido tiempo de inferencia, por lo que la elección entre estas dos opciones más igualadas en prestaciones dependerá del tiempo de ejecución. En cuanto a PointRend y Faster R-CNN, sus métricas tienen valores inferiores a Mask R-CNN y YOLOv5x.

Modelo	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{90}	AR^{bb}_{1}	AR^{bb}_{10}	Tiempo de inferencia (ms/imagen)
Faster R-CNN	0,806	0,988	0,958	0,445	0,603	0,848	36,9
YOLOv5x	**0,850**	**1,000**	**0,979**	**0,529**	**0,631**	**0,888**	**7,4**
Mask R-CNN	**0,855**	**0,988**	**0,959**	**0,663**	**0,624**	**0,881**	**36,4**
PointRend	0,762	0,975	0,941	0,343	0,571	0,811	41,6

Figura 4.20: tabla de resultados de detección de los modelos experimentados. AP^{bb}_{50}, AP^{bb}_{75} y AP^{bb}_{90} representan los valores de AP para los umbrales de IoU de 50, 75 y 90, mientras que AP^{bb} corresponde a la media aritmética de los mismos.

En segundo lugar, se analizan los resultados desde el punto de vista de la segmentación. Para abordar el problema de la segmentación se ha trabajado con dos modelos: Mask R-CNN y PointRend. En este caso se trabaja con las mismas métricas que en los modelos de detección, utilizando las máscaras de manera idéntica a las *bounding boxes*. Como se comentó en el apartado 4.2.1 de Mask R-CNN en su versión de TensorFlow, la dificultad radica en la resolución de las máscaras y en la segmentación de zonas como el morro y la cola del pez. Así, además de tener en cuenta en el APmask y el resto de métricas, se le dará una mayor relevancia al valor de AP^{mask}_{90} obtenido con un umbral de IoU de 90, ya que es la métrica más exigente, tomando como verdaderos positivos únicamente los solapamientos con un IoU mayor o igual al 90% (las segmentaciones más precisas).

Como se puede ver en la Figura 4.21, PointRend es ligeramente superior a Mask R-CNN, salvo en el AP^{mask}_{75} para un umbral de IoU de 75, PointRend es mejor que Mask R-CNN de hecho, en el AP^{mask}_{90} para un umbral de 90, PointRend supera claramente a Mask R-CNN, mejorando su resultado en aproximadamente un 11%, con tan sólo cinco milisegundos más de tiempo de inferencia.

Modelo	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{90}	AR^{mask}_{1}	AR^{mask}_{10}	Tiempo de inferencia (ms/imagen)
Mask R-CNN	0,856	0,988	0,978	0,708	0,611	0,871	36,4
PointRend	**0,87**	**0,997**	**0,956**	**0,792**	**0,643**	**0,896**	**41,6**

Figura 4.21: tabla de resultados de segmentación de los modelos.

La Figura 4.22 muestra una comparativa entre las segmentaciones producidas por ambos modelos sobre las mismas imágenes de la Figura 4.8 (segmentadas por la versión en TensorFlow de Mask R-CNN), que permite observar la mejora en la segmentación de PointRend, sobre todo en los extremos de los peces, como el morro y la cola. En esta comparativa, la mejora de la segmentación frente a Mask R-CNN es mucho más clara que en la tabla anterior. Además, las segmentaciones producidas por el modelo PointRend, de alta resolución, segmentan mucho mejor

las partes de los peces como la cabeza y la cola, zonas muy importantes a la hora de estimar las medidas de longitud y tamaño. De hecho, si se dan condiciones idóneas en la imagen, como un pez aislado con un fondo relativamente limpio, la segmentación con PointRend puede llegar a ser idéntica al *ground truth* a simple vista.

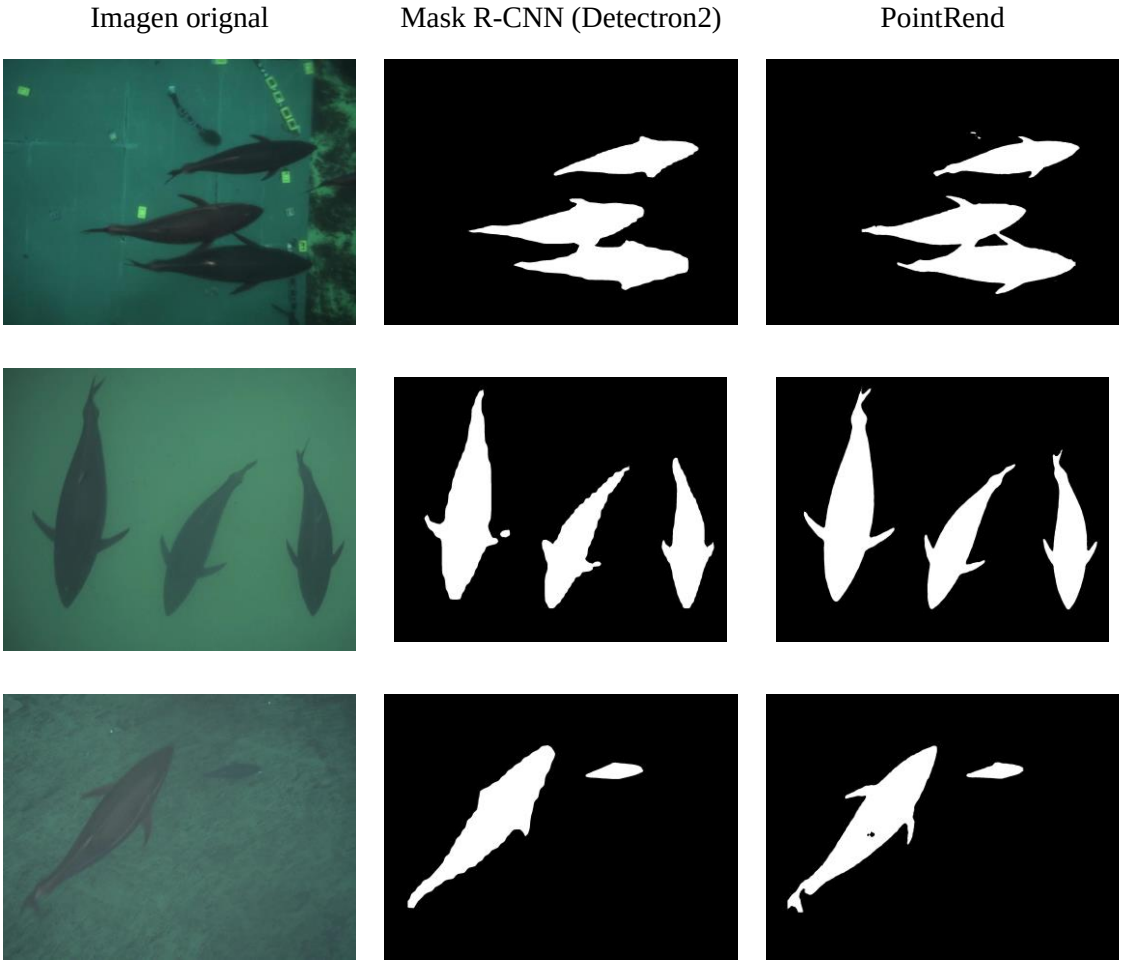


Figura 4.22: comparativa de segmentaciones de los modelos Mask R-CNN y PointRend.

Por otra parte, en la Figura 4.23 se puede observar la mejora en la segmentación con PointRend respecto a el método anterior del proyecto BIACOP basada en la técnica de *local thresholing*. Estas segmentaciones han sido producidas sobre un vídeo de las instalaciones del IEO con el fondo sin limpiar:

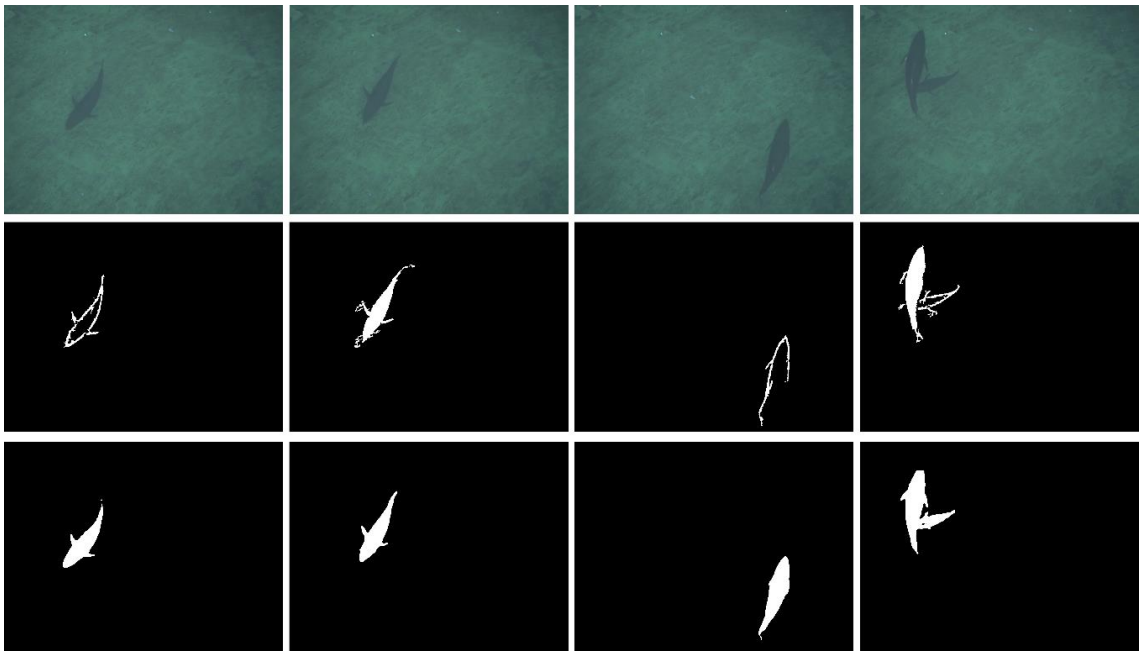


Figura 4.23: comparación de las segmentaciones producidas por la técnica de local thresholding y PointRend. Arriba: frame original; medio: local thresholding; abajo: PointRend.

4.3.2 Resultados para el número de peces extraídos

Dado un conjunto de imágenes o vídeos, es importante extraer la máxima información posible; es nuestro caso, detectar y segmentar el máximo número de peces para poder posteriormente realizar estudios estadísticos válidos. Como se comentó en el capítulo dos, el objetivo más ambicioso en el que se contextualiza este proyecto es en el de estimar medidas de tamaños de peces para hacer estimaciones de stock y biomasa. A un mayor número de medidas de peces, más reales serán las estimaciones y estadísticas que se infieran de ellas. En este apartado se analizan los resultados de los diferentes modelos trabajado desde el punto de vista del número de objetos extraídos (peces) que proporciona para un vídeo dado.

Para realizar este estudio se trabaja con un total de cinco vídeos estereoscópicos (apartado 2.2.1), que realmente equivales a diez vídeos, ya que se trabaja con cámaras estereoscópicas sincronizadas y por tanto son cinco vídeos por cámara. Los vídeos seleccionados intentan ser representativos de todos los escenarios en los que se trabaja. Así, de los cinco vídeos utilizados, dos de ellos corresponde al fondo del tanque sucio, uno al fondo del tanque con una pantalla artificial, uno al fondo del tanque limpio y uno al techo de las instalaciones de vista ventral (ver Figura 4.3).

Los vídeos son procesados con cada uno de los modelos propuestos, y finalmente se utilizan los desarrollos realizados en BIACOP para estimar medidas de los peces detectados (apartado 2.2.2). Con los vídeos procesados se midió el total de peces medidos junto con el tiempo total de procesamiento.

En cuanto al número total de peces medidos, como muestran la Figura 4.24 y la Figura 4.25, los modelos CNN con los que se ha trabajado en este proyecto consiguen un aumento considerable

en el número de mediciones obtenidas respecto a las técnicas tradicionales de VxC utilizadas en BIACOP. EN concreto, el modelo PointRend multiplica prácticamente por diez el número de medidas obtenidas en el segundo vídeo (773), con respecto a aplicar métodos tradicionales (80). Es necesario puntualizar que para el modelo Mask R-CNN, los valores proporcionados corresponden al número total de *bounding boxes* generadas, ya que las segmentaciones producidas podían resultar en medidas erróneas de longitud al no segmentar con precisión los extremos del pez como la cabeza y la cola. En general, se aprecia que los cuatro modelos utilizados obtienen un número de muestras semejante para un mismo vídeo, excepto para las segmentaciones de PointRend en los primeros dos vídeos, cuyo número de muestras es superior al resto de modelos. Estos vídeos corresponden a adquisiciones con fondo sucio, lo que podría justificar esta discrepancia en el número de muestras obtenidas, debido a que la segmentación en los modelos de detección (Mask R-CNN, Faster R-CNN, YOLOv5) se lleva a cabo mediante técnicas tradicionales aplicadas directamente en el área de las detecciones obtenidas (sobre el bbox). El fondo sucio dificulta la segmentación en mayor medida a las técnicas tradicionales que a la técnica de segmentación de PointRend, obteniendo este último un mayor número de segmentaciones. En el resto de los vídeos (vídeos tres, cuatro y cinco), el fondo es mucho más homogéneo y "limpio", con lo que segmentar objetos respecto al fondo no conlleva tanta dificultad para las técnicas tradicionales, y el número de muestras obtenidas por los cuatro modelos es similar. En la Figura 4.24 y la Figura 4.25 también cabe destacar los resultados del vídeo tres, muy similares para las cuatro técnicas aplicadas al procesamiento del vídeo. Ningún modelo destaca significativamente sobre el método tradicional, probablemente porque tanto los métodos convencionales como los modelos estudiados detectaron o segmentaron correctamente la mayoría de los peces del vídeo.

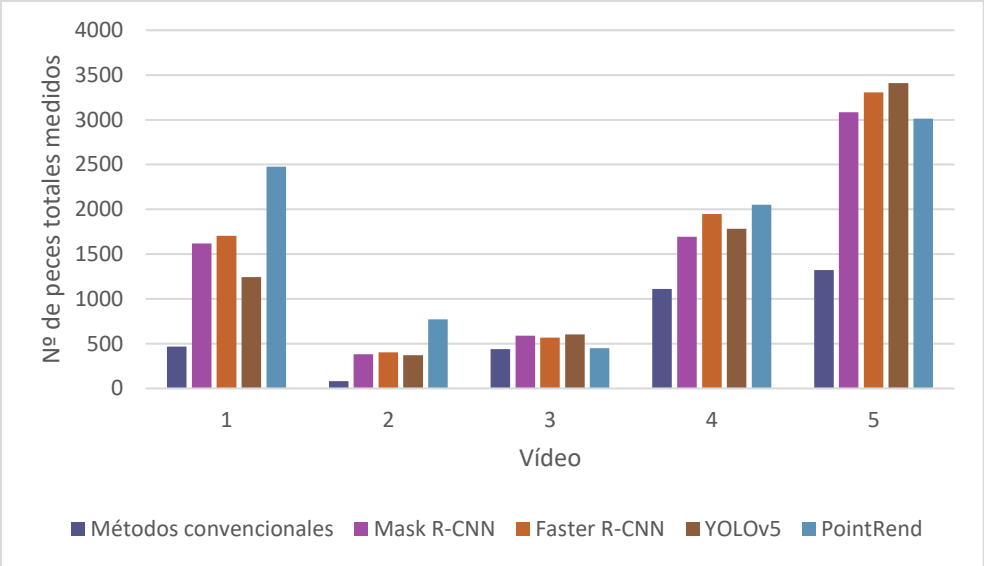


Figura 4.24: gráfica de número total de peces medidos con las diferentes técnicas de segmentación.

VÍDEO	Métodos convencionales	Mask R-CNN	Faster R-CNN	YOLOv5	PointRend
1	469	1619	1705	1242	2477
2	80	383	404	371	773
3	440	588	569	604	451
4	1112	1692	1946	1782	2052
5	1322	3085	3307	3409	3012

Figura 4.25: número total de peces medidos con las diferentes técnicas y modelos de segmentación, para los cinco vídeos estereoscópicos seleccionados.

En cuanto a los tiempos de procesamiento de cada vídeo, disponibles en la Figura 4.26 y la Figura 4.27, se aprecia que PointRend es el modelo más eficiente al compararlo con Mask R-CNN, Faster R-CNN y YOLOv5. La cantidad de horas de procesamiento de PointRend es inferior para cada uno de los vídeos respecto al resto de modelos estudiados. Pese a que PointRend tiene un tiempo de inferencia mayor al resto de los modelos, se ha de tener en cuenta que este modelo proporciona como resultado la segmentación final del pez, sobre el cual tan solo hay que aplicar el modelo y obtener sus medidas de longitud y ancho. En cambio, el resto de los modelos devuelven el área de la imagen (bbox del objeto) donde se ha de segmentar posteriormente el contorno del pez con otros métodos, aumentando con ello el tiempo total de procesamiento. Así, el tiempo de procesamiento de los vídeos con PointRend oscilo entre aproximadamente dos horas y 14 (vídeos dos y cinco, respectivamente), mientras que el procedimiento con YOLOv5 empleó en el procesamiento de los vídeos entre alrededor de cuatro y 33 horas (vídeos dos y cinco).

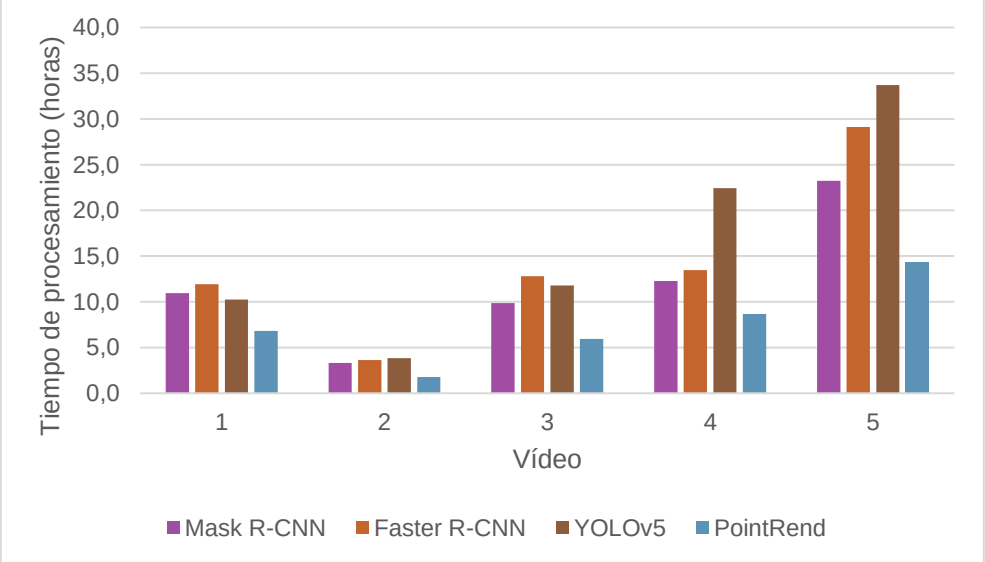


Figura 4.26: tiempo de procesamiento de cada vídeo en horas para cada modelo.

VÍDEO	Mask R-CNN	Faster R-CNN	YOLOv5	PointRend
1	10,9	11,9	10,3	6,8
2	3,3	3,6	3,8	1,8
3	9,9	12,8	11,8	5,9
4	12,3	13,5	22,4	8,7
5	23,2	29,1	33,7	14,4

Figura 4.27: tiempo de procesamiento de cada vídeo con cada modelo en horas.

Como se ha descrito en este apartado, a partir de los resultados de AP y AR de los diferentes modelos (Figura 4.20 y Figura 4.21), su tiempo de inferencia y el número de peces medidos en los vídeos, se puede deducir que los modelos más adecuados para ser utilizados en el proyecto de estimación de biomasa son YOLOv5 o Mask R-CNN para detección y PointRend para segmentación. Sin embargo, el tiempo total necesario para el procesamiento de vídeo (Figura 4.26 y Figura 4.27) muestra la ventaja en cuanto a tiempo de segmentar directamente con DL en vez de utilizarlo para detectar los peces como solución intermedia. Además, ante imágenes con un fondo irregular con diferentes condiciones de iluminación, los parámetros en los que se basan los métodos tradicionales se han de reajustar manualmente, mientras que con las técnicas de DL no es necesario, ya que son aprendidas por el modelo si se le proporciona un conjunto de imágenes suficientemente amplio y no sesgado. De esta manera, de acuerdo con los experimentos realizados y los resultados obtenidos, se puede concluir que el modelo PointRend es el más adecuado para el proyecto que nos ocupa.

5. Conclusiones

Este proyecto ha permitido explorar las arquitecturas, técnicas y metodologías de las CNNs y *deep learning* (DL) aplicadas a los problemas de detección y segmentación de objetos en imágenes El conjunto de datos o *ground truth* utilizado en los experimentos ha sido confeccionado específicamente para este trabajo, abordando de forma manual y supervisada tanto el proceso de anotación o etiquetado como el de segmentación de las imágenes que lo componen. En concreto, se ha experimentado con modelos y *frameworks* considerados, en la literatura especializada, de la vanguardia del DL en el área de la visión por computador. Los desarrollos realizados se contextualizan en un proyecto actual de investigación del grupo de VxC del instituto Ai2 de la UPV y los desarrollos descritos en este documento se encuentran en fase de incorporación a dicho proyecto de investigación dada las mejoras de prestaciones que podrían suponer según los resultados alcanzados este trabajo.

5.1 Conclusiones

Este trabajo fin de grado (TFG) se ha centrado en estudiar técnicas de VxC y DL que automaticen completamente el proceso de identificación y segmentación, consiguiendo segmentar el mayor número de peces y con la suficiente precisión en imágenes subacuáticas.

El primer problema a resolver para poder desarrollar este proyecto fue la necesidad de conseguir el suficiente volumen de imágenes en el formato adecuado. Trabajar con modelos de CNN implica disponer de un gran volumen de datos no sesgados del problema. Afortunadamente el grupo de VxC dispone de gran cantidad de vídeos adquiridos con cámaras estereoscópicas, en las instalaciones de Mazarrón del IEO y del grupo Balfegó en el Mediterráneo, con atunes nadando libremente. Así, se confeccionó un conjunto de datos inicial (GT-180) que se fue ampliando para poder lograr una mayor precisión con los modelos (GT-1000), ambos conjuntos fueron creados de forma manual y supervisada con la herramienta VGG Image Annotator (VIA).

Los modelos sobre los que se ha trabajado y obtenido resultados incluyen tanto modelos para la detección de objetos en imágenes (Faster R-CNN, Mask R-CNN y YOLOv5), y como para la segmentación de objetos en imágenes (Mask R-CNN y PointRend).

En primer lugar, se comenzó trabajando con el modelo Mask R-CNN, ya que se partió de la idea de utilizar los modelos de CNNs para segmentar, pero las segmentaciones que proporcionó dicho modelo eran de una resolución demasiado baja para poder ser utilizadas directamente en la estimación de medidas de longitud de los peces con precisión. La baja resolución de dichos resultados nos impulsó a considerar alternativas al planteamiento original y experimentar con otros modelos, como Faster R-CNN o YOLOv5. Para trabajar con estos nuevos modelos fue necesario y adaptar las anotaciones del conjunto de datos y rediseñar el *ground truth* GT-1000, abordando así el problema de detección de objetos.

Respecto a los conocimientos cursados en el grado, cabe destacar que, si bien es cierto que las implementaciones de los modelos explorados en este trabajo son de una complejidad mucho

mayor que lo visto durante los diferentes cursos del grado, las bases y los fundamentos básicos de su funcionamiento sí han sido estudiados durante el grado. Sistemas Inteligentes, Percepción o Aprendizaje Automático son algunas asignaturas cuyos contenidos están directamente relacionados con este trabajo. A pesar de ello, se ha realizado un gran esfuerzo en el uso y puesta en marcha de estos modelos, ya que ha sido necesario entender el funcionamiento básico de cada uno de ellos, junto con el formato de las entradas requeridas y las salidas que genera cada modelo. En concreto, para el desarrollo del proyecto, ha sido necesario estudiar nuevos conocimientos, tales como los conceptos básicos de la programación en TensorFlow, Keras y PyTorch, y otras librerías de Python como OpenCV, NumPy o scikit-learn; además de conocimientos como el funcionamiento de las CNNs, los modelos utilizados y su aplicabilidad. Asimismo, este trabajo ha puesto a prueba multitud de competencias transversales adquiridas durante los diferentes cursos, como es el caso de la planificación y gestión del tiempo, la comunicación efectiva y el trabajo en equipo a la hora de trabajar junto al resto de miembros del grupo de VxC, o la creatividad y el análisis y resolución de problemas para abordar los obstáculos comentados anteriormente.

Para poder comparar resultados entre modelos (capítulo cuatro), se han procesado un total de cinco vídeos con todos los modelos estudiados. Las conclusiones que se pueden deducir, respecto a la idoneidad de incorporar un modelo u otro a los desarrollos del proyecto de estimación de biomasa (BIACOP) de dichos resultados experimentales son:

- YOLOv5 o Mask R-CNN son los modelos más adecuados para la detección de peces, según se infiere de los resultados de sus AP, AR, tiempo de inferencia y número de peces medidos en los vídeos (Figura 4.20 y Figura 4.21).
- PointRend es el modelo más adecuado para la segmentación, según se infiere del resultado de su AP, AR, tiempo de inferencia y número de peces medidos en los vídeos (Figura 4.20 y Figura 4.21).
- El tiempo total necesario para el procesamiento de vídeo (Figura 4.26 y Figura 4.27) muestra ventajas de utilizar directamente modelos que proporcionen segmentación frente a modelos que detecten peces como un paso previo para continuar trabajando con una técnica de segmentación.
- Al procesar imágenes con un fondo irregular con diferentes condiciones de iluminación, los modelos han conseguido buenos resultados, de lo que se deduce que el conjunto de imágenes utilizando en el entrenamiento es lo suficientemente amplio y no sesgado.
- Finalmente, de acuerdo con los experimentos realizados y los resultados obtenidos, se puede concluir que el modelo PointRend es el más adecuado para incorporar al proyecto BIACOP de estimación de biomasa.

En cuanto a la consecución de los objetivos concretos planteados en el inicio de este trabajo y expuestos en el capítulo uno:

- Se han estudiado las arquitectura, técnicas y metodologías de CNNs y DL aplicadas a los problemas de detección y segmentación de objetos en imágenes más actuales.
- Se ha trabajado tanto con herramientas *software* como *hardware*: se han utilizado los *framework*s de TensorFlow y Pytorch, se ha hecho uso de un sistema NAS para

almacenamiento y se ha trabajado con equipos dotados de tarjetas gráficas con varias GPUs haciendo uso de sus prestaciones

- Se ha confeccionado un *ground trouth* o conjunto de dato anotados adecuado para los desarrollos realizados (GT-1000), el cual se ha ido adaptando y configurando a las necesidades de cada modelo.
- Se ha conseguido mejorar el método de segmentación, respecto al utilizado hasta ahora en el proyecto BIACOP, logrando multiplicar hasta diez veces el número de detecciones y peces medidos en cada vídeo.
- Se han obtenido resultados que demuestran que el tiempo de procesamiento de cada vídeo es más eficiente si la segmentación se lleva a cabo mediante técnicas basadas en DL, en lugar de con técnicas tradicionales.
- Se ha conseguido aumentar la calidad de la segmentación producida respecto al método de segmentación utilizado hasta ahora en el proyecto BIACOP, permitiendo así obtener un mayor número de estimaciones de medidas por vídeo, al mismo tiempo que la precisión de estas ha aumentado.

5.2 Trabajos futuros

Teniendo en cuenta los resultados de este proyecto, a continuación, se proponen una serie de posibles trabajos futuros que podrían ser abordados tanto con los modelos de CNN ya estudiados, como infiriendo nuevos modelos adecuados a cada desarrollo.

- Adaptar los desarrollos realizados a imágenes y vídeos grabados en otros entornos y condiciones como en las jaulas de Balfegó, con una cantidad de peces mucho mayor. Esto dificulta las detecciones, ya que estos pasan ante las cámaras en grupos y aparecen solapados la mayoría de las veces.
- La incorporación de la vista lateral de los peces es también una propuesta interesante. Desde esta posición, las condiciones de visibilidad son peores y es probable que esto conlleve errores de segmentación. En las imágenes de vistas ventral y dorsal con las que se ha trabajado, los fondos de las imágenes son el techo de las instalaciones y el fondo del tanque, respectivamente, donde las siluetas a segmentar destacan sobre los fondos. En la vista lateral, en cambio, los peces han de segmentarse sobre un fondo en el que apenas destacan, mimetizándose a menudo y reflejando sobre sus escamas los colores del fondo. Además, para esta vista no se cuenta con un modelo con el cual estimar las medidas de los peces, por lo que este aspecto es clave para trabajar con la vista lateral.
- Otro posible desarrollo, sobre el cual ya se está investigando en el proyecto BIACOP, consiste en apoyarse de la información acústica de una ecosonda, colocada junto al sistema de cámaras estereoscópico. Estas señales pueden ser tratadas como imágenes, detectándose las señales de los peces con CNNs. La principal ventaja de esta solución es que no son necesarias ningunas condiciones de luminosidad concretas, por lo que es importante explorar esta posible incorporación al sistema de estimación de medidas.
- Otro trabajo futuro es el uso de la segmentación y la estimación de medidas en el momento de la captura de los atunes. Estos son sacados de la jaula con la ayuda de una grúa. La propuesta consiste en utilizar una única cámara y estimar las medidas

3D apoyándose en una esfera de diámetro conocido situado en la grúa. Bajo la grúa, el atún gira sobre sí mismo. Así, si se es capaz de estimar medidas de vista lateral, se podría obtener una estimación más precisa de las medidas del individuo que únicamente con el modelo de vistas dorsal y ventral. Se puede ver un ejemplo de las imágenes obtenidas con este montaje en la Figura 5.1.

Figura 5.1: imagen del montaje en el momento de la captura.

Bibliografía

[1] Comisión Europea and C. y T. Dirección General de Redes de Comunicación, "LIBRO BLANCO sobre la inteligencia artificial-un enfoque europeo orientado a la excelencia y la confianza," 2020. Accessed: Jun. 23, 2021. [Online]. Available: https://ec.europa.eu/info/sites/default/files/commission-white-paper-artificial-intelligence-feb2020_es.pdf

[2] Ametic, "Posicionamiento en Inteligencia Artificial," 2020. https://ametic.es/sites/default/files//posicionamiento_ia.pdf (accessed Jun. 23, 2021).

[3] ICCAT, "Recommendation by ICCAT to establish a multi-annual recovery plan for bluefin tuna in eastern Atlantic and Mediterranean," 2006.

[4] B. Collette *et al.*, "Thunnus thynnus. The IUCN Red List of Threatened Species," 2011, doi: 10.2305/IUCN.UK.2011-2.RLTS.T21860A9331546.en.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Nov. 2013, [Online]. Available: http://arxiv.org/abs/1311.2524

[6] R. Girshick, "Fast R-CNN," Apr. 2015, [Online]. Available: http://arxiv.org/abs/1504.08083

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.01497

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Mar. 2017, [Online]. Available: http://arxiv.org/abs/1703.06870

[9] A. Kirillov, Y. Wu, K. He, and R. Girshick, "PointRend: Image Segmentation as Rendering," Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.08193

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.02640

[11] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10. pp. 1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[12] P. Muñoz-Benavent, G. Andreu-García, J. M. Valiente-González, V. Atienza-Vanacloig, V. Puig-Pons, and V. Espinosa, "Automatic Bluefin Tuna sizing using a stereoscopic vision system," *ICES Journal of Marine Science*, vol. 75, no. 1, pp. 390–401, Jan. 2018, doi: 10.1093/icesjms/fsx151.

[13] P. Muñoz-Benavent, G. Andreu-García, J. M. Valiente-González, V. Atienza-Vanacloig, V. Puig-Pons, and V. Espinosa, "Enhanced fish bending model for automatic tuna sizing

using computer vision," *Computers and Electronics in Agriculture*, vol. 150, pp. 52–61, Jul. 2018, doi: 10.1016/j.compag.2018.04.005.

[14] X. Yang, S. Zhang, J. Liu, Q. Gao, S. Dong, and C. Zhou, "Deep learning for smart fish farming: applications, opportunities and challenges," *Reviews in Aquaculture*, vol. 13, no. 1. Wiley-Blackwell, pp. 66–90, Jan. 01, 2021. doi: 10.1111/raq.12464.

[15] A. Álvarez-Ellacuría, M. Palmer, I. A. Catalán, and J. L. Lisani, "Image-based, unsupervised estimation of fish size from commercial landings using deep learning," *ICES Journal of Marine Science*, vol. 77, no. 4, pp. 1330–1339, Jul. 2020, doi: 10.1093/icesjms/fsz216.

[16] M. Zaher and A. Ulmoula, "BIOMASS ESTIMATION OF FISH USING DEEP NETWORKS AND STEREO VISION," 2020.

[17] C. H. Tseng, C. L. Hsieh, and Y. F. Kuo, "Automatic measurement of the body length of harvested fish using convolutional neural networks," *Biosystems Engineering*, vol. 189, pp. 36–47, Jan. 2020, doi: 10.1016/j.biosystemseng.2019.11.002.

[18] R. Garcia *et al.*, "Automatic segmentation of fish using deep learning with application to fish size measurement," *ICES Journal of Marine Science*, vol. 77, no. 4, pp. 1354–1366, Jul. 2020, doi: 10.1093/icesjms/fsz186.

[19] AQ1 SYSTEMS, "AM100. FISH SIZING AND COUNTING TECHNOLOGY." Accessed: Jun. 08, 2021. [Online]. Available: www.aq1systems.com

[20] Instituto Español de Oceanografía, "Instalaciones - ICAR." http://icar.ieo.es/facilities/ (accessed Jun. 20, 2021).

[21] V. Atienza-Vanacloig, G. Andreu-García, F. López-García, J. M. Valiente-González, and V. Puig-Pons, "Vision-based discrimination of tuna individuals in grow-out cages through a fish bending model," *Computers and Electronics in Agriculture*, vol. 130, pp. 142–150, Nov. 2016, doi: 10.1016/j.compag.2016.10.009.

[22] F. Chollet, *Deep Learning with Python*. Manning Publications Co, 2018.

[23] G. Jocher, A. Stoken, and J. Borovec, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations (Version v5.0). Zenodo.," 2021.

[24] L. Simon, R. Webster, and J. Rabin, "Revisiting Precision and Recall Definition for Generative Model Evaluation." 2019.

[25] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 814–830, Feb. 2015, doi: 10.1109/TPAMI.2015.2465908.

[26] "Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow | by Waleed Abdulla | Matterport Engineering Techblog." https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46 (accessed Jun. 11, 2021).

[27] W. Abdulla, “Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow,” *GitHub repository*, 2017. https://github.com/matterport/Mask_RCNN (accessed Jun. 11, 2021).

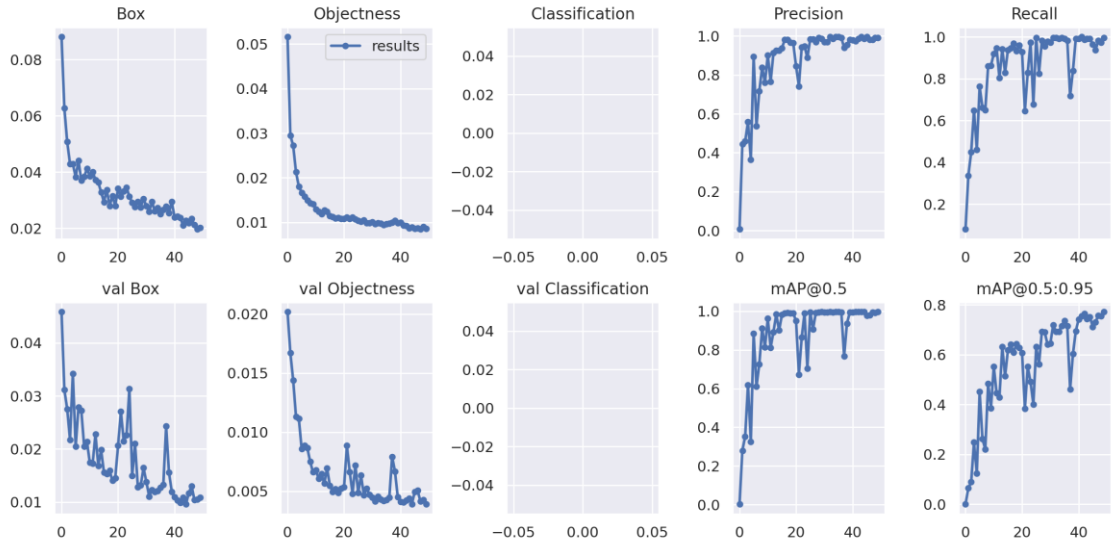
[28] G. Jocher *et al.*, “ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations,” Apr. 2021, doi: 10.5281/ZENODO.4679653.

[29] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” Apr. 2018, Accessed: Jul. 08, 2021. [Online]. Available: https://arxiv.org/abs/1804.02767v1

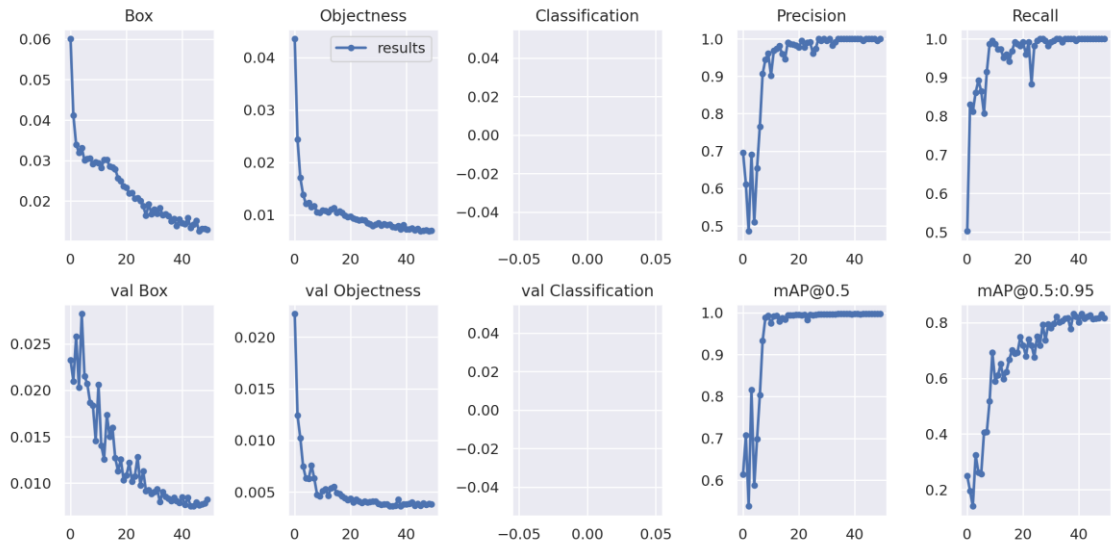
[30] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” 2019. https://github.com/facebookresearch/detectron2 (accessed Jun. 14, 2021).

Anexos

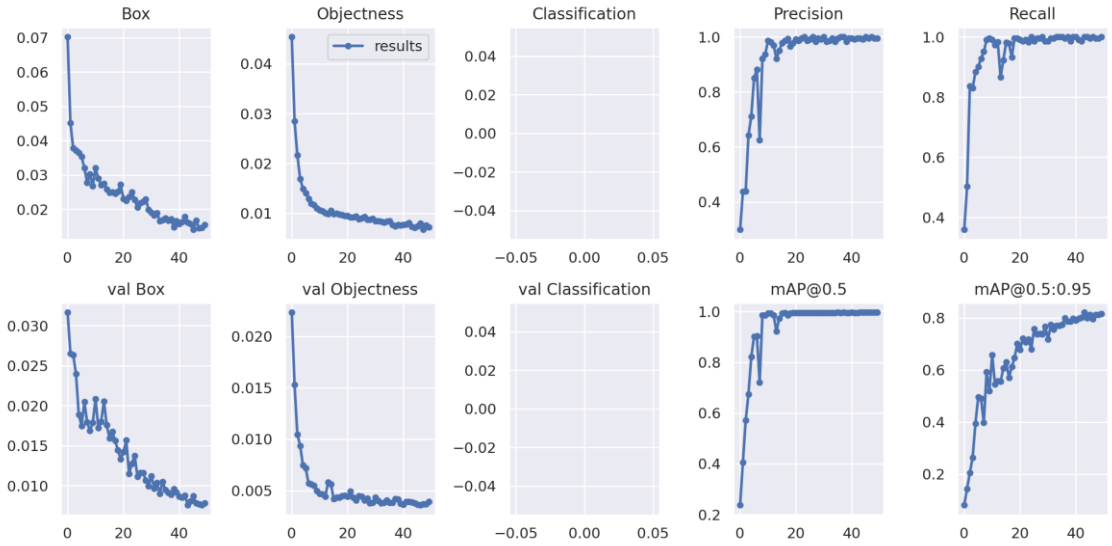
A. Resultados del entrenamiento de YOLOv5s



B. Resultados del entrenamiento de YOLOv5m



C. Resultados del entrenamiento de YOLOv5l



D. Resultados del entrenamiento de YOLOv5x