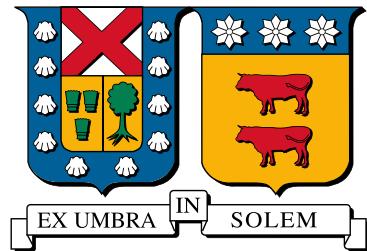


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



**“SEGMENTACIÓN DE INSTANCIAS Y
SEGUIMIENTO BASADO EN DETECCIÓN
PARA CARACTERIZACIÓN DE BIOMASA Y
SALUD EN LA SALMONICULTURA”**

JULIO EDUARDO LÓPEZ BLANCHE

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
ELECTRÓNICO, MENCIÓN COMPUTADORES

PROFESOR GUÍA: MARCOS ZÚÑIGA
PROFESOR CORREFERENTE: ALEJANDRO WEINSTEIN

JULIO 2025

Agradecimientos

Al equipo de WildSense por proveer la base de datos de salmones y una computadora para el entrenamiento y validación de modelos de forma remota.

A Natalia Vega Quinteros por facilitar plantillas para los informes de ELO-307 y el documento presente.

Segmentación de instancias y seguimiento basado en detección para caracterización de biomasa y salud en la salmonicultura

Julio Eduardo López Blanche

Memoria para optar al título de Ingeniero Civil Electrónico, Mención en Computadores y Sub-Mención en Telecomunicaciones.

Universidad Técnica Federico Santa María

Profesor Guía: Dr. Marcos Zúñiga Barraza

Julio 2025

Resumen

En la industria piscícola, el monitoreo constante de la salud de los peces es crucial. Gracias a los avances en visión por computadora es posible realizar esta labor de forma escalable y menos invasiva. WildSense, empresa spin-off de la UTFSM, provee servicios para la estimación de masa en salmones, donde la segmentación de instancias y seguimiento basado en detección son parte fundamental de su “pipeline”, pero aún presenta oportunidades de optimización.

Este proyecto perfecciona una base de datos de segmentación de instancias de salmones, al depurarla para incluir únicamente casos de interés, lo que permite entrenar modelos YOLO con rendimientos superiores a trabajos previos. Se optimizan los hiperparámetros durante el entrenamiento y se exportan los modelos a TensorRT para reducir los tiempos de inferencia.

Los resultados demuestran que una base de datos más precisa mejora la calidad de los modelos, la optimización de hiperparámetros produce mejores resultados y la conversión a TensorRT reduce significativamente los tiempos de inferencia, con mínima pérdida de desempeño.

Palabras Clave: Visión por computadora, segmentación de instancias, seguimiento basado en detección, aprendizaje de máquina, aprendizaje profundo, modelos convolucionales, hiperparámetro, YOLO, cuantización numérica, TensorRT.

Instance segmentation and tracking by detection for biomase and health characterization in aquaculture of salmonids.

Julio Eduardo López Blanche

Final Project Report for the Electronic Civil Engineering degree with Minor in Computers and Secondary Minor in Telecommunications.

Universidad Técnica Federico Santa María

Advisor: Dr. Marcos Zúñiga Barraza

July 2025

Abstract

In the aquaculture industry, the constant monitoring of fish health is crucial. Advances in computer vision now allow this task to be performed in a scalable and less invasive manner. WildSense, a spin-off company of the UTFSM, provides services for weight estimation in salmon, for which instance segmentation and detection-based tracking are integral components of its pipeline, although there remains room for optimization.

This project refines an instance segmentation database for salmon by filtering it to include only relevant cases, thereby enabling the training of YOLO models with performance superior to previous works. Hyperparameters are optimized during training, and the models are exported to TensorRT in order to reduce inference times.

The results demonstrate that a more precise database enhances the quality of the models, hyperparameter optimization yields better outcomes, and the conversion to TensorRT significantly reduces inference times with minimal performance loss.

Keywords: Computer vision, instance segmentation, tracking by detection, machine learning, deep learning, convolutional models, hyperparameter, YOLO, numerical quantization, TensorRT.



DEPARTAMENTO DE
ELECTRONICA
UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA



DEPARTAMENTO DE
ELECTRONICA

Índice de contenidos

Introducción	1
1. Contexto general	1
1.1. Objetivo general	1
1.2. Objetivos específicos	1
2. Marco Teórico	2
2.1. Visión por computadora y su uso en la acuicultura	2
3. Estado del arte	3
3.1. Segmentación en imágenes	3
3.2. Modelos de segmentación de instancias	4
3.2.1. Mask-RCNN	4
3.2.2. MS-RCNN	4
3.2.3. SOLOv2	4
3.2.4. YOLOv8	5
3.2.5. YOLOv9	5
3.2.6. YOLOv11	5
3.3. Hiperparámetros de entrenamiento	8
3.3.1. Optimizador	8
3.3.2. Técnicas de regularización	8
3.3.3. Ajustador de la tasa de aprendizaje	9
3.3.4. Tamaño de lote	9
3.4. Técnicas de optimización	9
3.4.1. Pruning (poda)	10
3.4.2. Cuantización numérica	10
3.4.3. Knowledge Distillation	11
3.5. Seguimiento	11
3.5.1. Algoritmos de seguimiento relevantes	12

Trabajo a realizar	13
4. Trabajo realizado por WildSense y motivación de mejoras	13
5. Metodología	15
6. Herramientas computacionales	16
6.1. Herramientas de desarrollo y despliegue	16
6.1.1. Ultralytics	16
6.2. Herramientas de etiquetado	17
6.2.1. CVAT	17
7. Resultados	18
8. Conclusión	19
8.1. Trabajo Futuro	19
Anexos	29
9. Estado del arte en modelos de segmentación de instancias	29
9.1. Técnicas tradicionales para segmentación de imágenes	29
9.2. Técnicas con redes neuronales	30
9.3. Marcos de detección: dos etapas versus una etapa	30
9.3.1. Categorización taxonómica	31
9.4. Segmentación de instancias en vídeo	32

Índice de tablas

6.1. Plataformas de desarrollo en Python para visión por computadora	16
6.2. Plataformas de etiquetado de bases de datos	17
9.3. Taxonomía de técnicas de segmentación con sus fortalezas y debilidades	31

Índice de figuras

2.1. Flujo de trabajo propuesto por Tonachella para estimación de la talla en peces	2
3.2. Tipos de segmentación en visión por computadora	3
3.3. Arquitectura Mask RCNN	4
3.4. Arquitectura MS RCNN	4
3.5. Arquitectura SOLO	5
3.6. Arquitectura SOLOv2	5
3.7. Arquitectura YOLOv8 de detección	6
3.8. Arquitectura YOLOv8 de segmentación utilizada en FastSAM	6
3.9. PGI de YOLOv9 y otras arquitecturas de red y métodos relacionados	7
3.10. Mapas de características para diferentes arquitecturas de red (PlainNet, ResNet, CSPNet y GELAN)	7
3.11. Módulos arquitectónicos clave de YOLO11	7
3.12. Representación de una red optimizada con poda y cuantización	10
4.13. Flujo de trabajo implementado por WildSense para estimación de masa y volumen en peces	14

Índice de códigos

Introducción

1. Contexto general

WildSense, empresa spin-off de la UTFSM, entre sus muchas actividades provee servicios de estimación de biomasa con análisis de profundidad estéreo [1]. Es en este ámbito que la visión por computadora juega un rol fundamental, donde la segmentación de instancias y el seguimiento funcionan como etapas iniciales en la caracterización biométrica de salmones [2] (Fig. 4.13).

Previo a la estimación de masa se descartan las muestras que presentan deficiencias, las cuales constituyen una mayoría de los casos. Esta situación es una clara inefficiencia en el flujo de procesamiento. En consecuencia, se plantea el desarrollo de una base de datos especializada que permita entrenar modelos de segmentación orientados únicamente a identificar los peces de interés, reduciendo así los tiempos de procesamiento en las etapas de seguimiento y posteriores. Además, se busca ampliar y explorar aspectos poco abordados en trabajos previos. Se llevarán a cabo entrenamientos con búsqueda exhaustiva de hiperparámetros en modelos tales como YOLOv8, YOLOv9 y YOLOv11, además de experimentar con el despliegue en Tensor-RT.

1.1. Objetivo general

Entrenar modelos de segmentación de instancias YOLO para una base de datos de salmones mejorada y validar los modelos resultantes en las tareas de segmentación y seguimiento.

1.2. Objetivos específicos

- Optimizar una base de datos de salmones para adecuarla al caso de uso de WildSense.
- Evaluar y buscar hiperparámetros que potencien el rendimiento de modelos entrenados.
- Llevar a cabo entrenamientos con una base de datos pública para fines de replicación.
- Validar los modelos en segmentación de instancia y seguimiento basado en detección.
- Concluir sobre los efectos de optimizar la base de datos, la búsqueda de hiperparámetros y la exportación con cuantización numérica.

2. Marco Teórico

2.1. Visión por computadora y su uso en la acuicultura

La visión por computadora, también llamada visión artificial, es una disciplina dentro del aprendizaje de máquina y la informática que se centra en desarrollar algoritmos y sistemas capaces de interpretar y analizar imágenes y videos. Esto se logra mediante técnicas de procesamiento digital, modelado matemático, análisis de patrones y métodos estadísticos que permiten identificar objetos, segmentar escenas y comprender el contenido visual. Tiene aplicaciones prácticas en áreas como la robótica, la seguridad o la conducción autónoma.

La acuicultura constituye cerca de dos tercios de la producción de productos marinos en el mundo [3] y la piscicultura forma parte importante de la actividad de acuicultura del país [4]. Es entonces de vital importancia monitorear la integridad y salud de los peces por medio de revisiones regulares para asegurar una alimentación adecuada y la detección temprana de enfermedades. Lamentablemente, las mediciones manuales dependen de las condiciones meteorológicas, exigen mucho tiempo y trabajo y, sobre todo, son invasivas y estresantes para los peces [5].

Como solución alternativa se han propuesto métodos de estimación estadística para la talla de peces mediante sensores, radares y cámaras [5] (Fig. 2.1), y el uso de modelos de aprendizaje automático en detección de enfermedades [6]. Estas soluciones prometen ser más escalables, menos costosas e intervenir menos en el ciclo de vida de los animales.

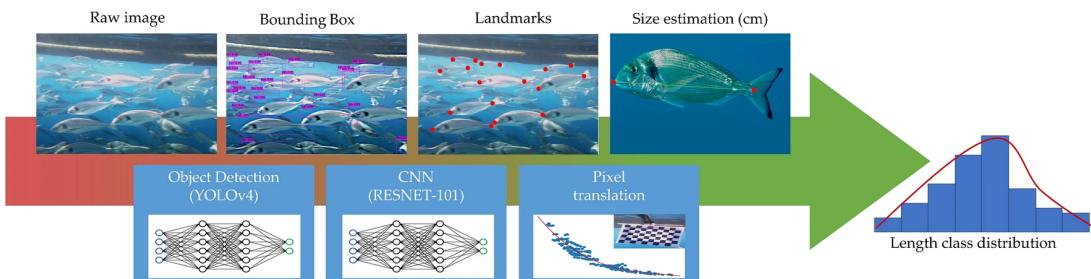


Figura 2.1: Proceso general del método propuesto por Tonachella [5]: reconocimiento automático por visión artificial y estimación de la talla en peces.

WildSense por su parte implementa una novedosa técnica basada en el emparejamiento de mapas de profundidad obtenidos con cámaras estéreo y modelos tridimensionales generados en Blender, para la estimación de volumen y masa en peces.

3. Estado del arte

En la presente sección se exponen los avances recientes y las técnicas predominantes en tareas de visión por computadora, enfocándose en la segmentación de instancias, el seguimiento de objetos y la optimización del entrenamiento de modelos. Se abordan además las técnicas actuales de optimización y las consideraciones en validación y despliegue.

3.1. Segmentación en imágenes

La segmentación de imágenes constituye una técnica común en el procesamiento de imágenes digitales, para dividir una imagen en diversas regiones, a menudo basándose en las características de los píxeles. De interés particular es la segmentación de instancias, que permite identificar cada objeto presente en la imagen asignándole una clase y una máscara específica a nivel píxel (Fig. 3.2c).



(a) Imagen original.



(b) Segmentación semántica.



(c) Segmentación de instancias.



(d) Segmentación panóptica.

Figura 3.2: Tipos de segmentación en visión por computadora.

3.2. Modelos de segmentación de instancias

Diversas arquitecturas han sido desarrolladas para abordar el problema de la segmentación de instancias, con especial relevancia en aplicaciones de tiempo real. Se describen a continuación algunos de los modelos más representativos para este propósito.

3.2.1. Mask-RCNN

Mask R-CNN [7] es una extensión del modelo Faster R-CNN [8] donde se añade una tercera rama en su arquitectura, que integra una capa para predecir la máscara binaria de cada objeto detectado. Esto le permite detección y segmentación simultánea de objetos.

3.2.2. MS-RCNN

MS-RCNN [9] representa una mejora sobre Mask R-CNN mediante la incorporación del mecanismo denominado “mask scoring”, que mide la precisión de las máscaras segmentadas para cada objeto en vez de asumir una confianza fija. Esto se logra introduciendo una capa adicional que evalúa la calidad de cada máscara generada, mejorando así el rendimiento en la segmentación de objetos y reduciendo las falsas detecciones.

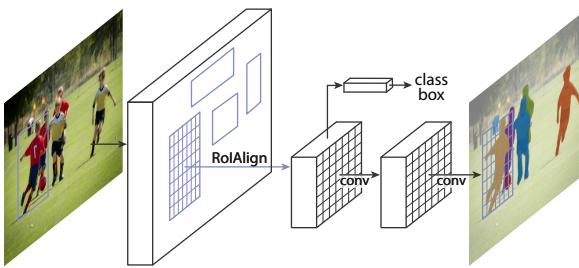


Figura 3.3: Arquitectura Mask RCNN.

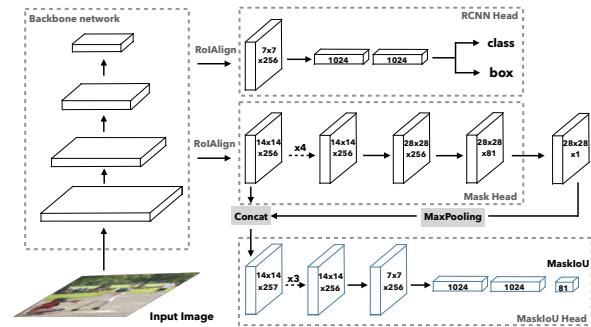


Figura 3.4: Arquitectura MS RCNN.

3.2.3. SOLOv2

El modelo SOLO [10] introduce un enfoque basado en la división de la imagen en una malla de cuadrículas, eliminando la necesidad de propuestas de regiones o anclas. SOLOv2 [11] incorpora mejoras significativas, entre las que destaca la convolución dinámica, con la que se obtiene una mayor precisión en segmentación sin aumentar considerablemente la complejidad computacional.

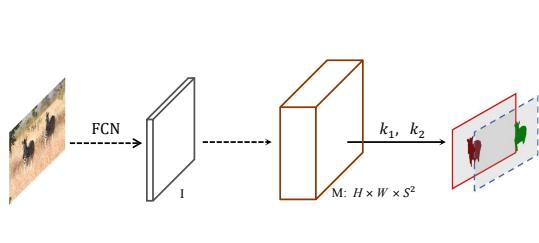


Figura 3.5: Arquitectura SOLO.

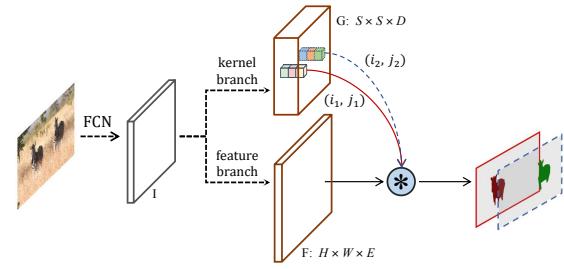


Figura 3.6: Arquitectura SOLOv2.

3.2.4. YOLOv8

La serie YOLO, en su versión YOLOv8 [12], ha sido diseñada para la detección en tiempo real, combinando eficiencia y precisión. Esta versión introduce un diseño modular, una red “backbone” basada en una variante optimizada de CSPDarknet53 (con el módulo de convolución C2f) y mecanismos de detección sin anclas [13]. Con la incorporación de una red Proto es además capaz de realizar tareas de segmentación (Fig. 3.8).

3.2.5. YOLOv9

La versión YOLOv9 [15] introduce innovaciones encaminadas a mejorar la precisión y la eficiencia, haciendo uso de mecanismos como Programmable Gradient Information (PGI), que optimiza la transmisión de gradientes mediante una rama reversible auxiliar. Además, se incorpora GE-LAN (Generalized Efficient Layer Aggregation Network), que facilita una mejor utilización de los parámetros y posibilita configuraciones ajustables según el dispositivo.

3.2.6. YOLO11

YOLO11 [16] se posiciona como la última actualización de la serie YOLO por parte de Ultralytics, optimizando sus capacidades en tareas de segmentación de instancias, estimación de pose y detección de objetos orientados, etc. La introducción del bloque C3k2 y del módulo de convolución con atención espacial paralela (C2PSA) permite mejorar la velocidad y la precisión mediante la reducción de la carga computacional [17].

YOLOv8

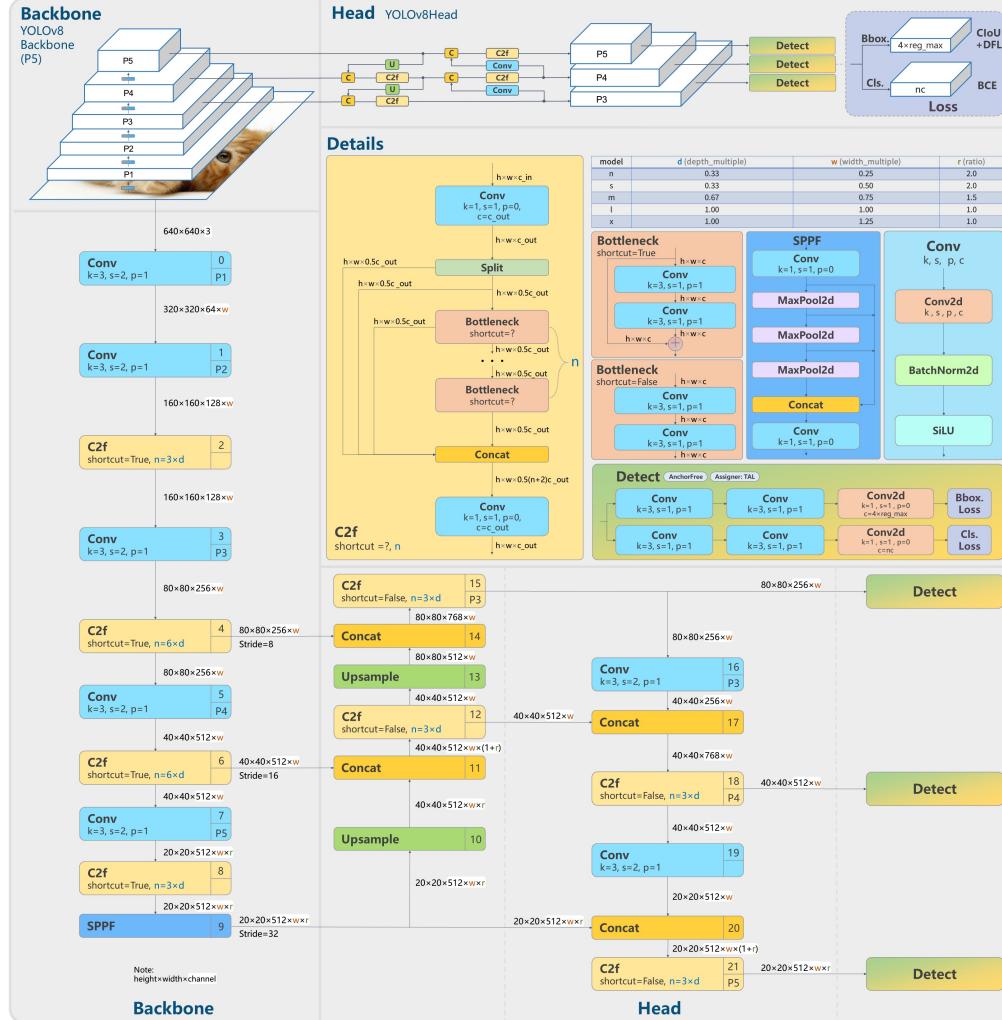


Figura 3.7: Arquitectura YOLOv8 de detección.

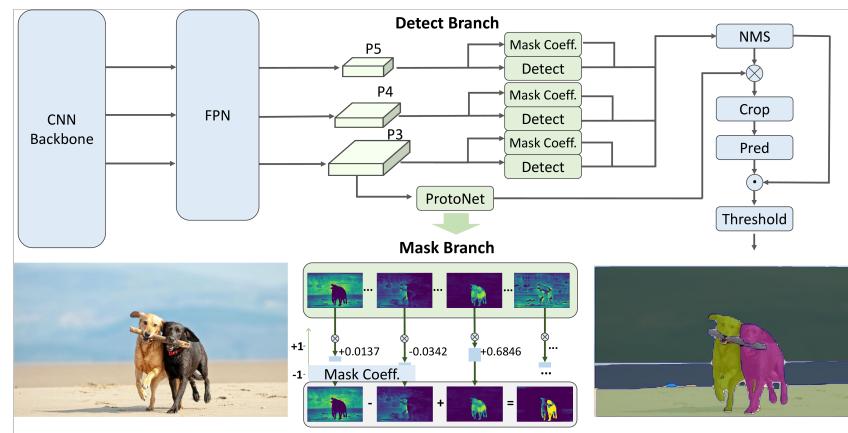


Figura 3.8: Arquitectura YOLOv8 de segmentación utilizada en FastSAM [14].

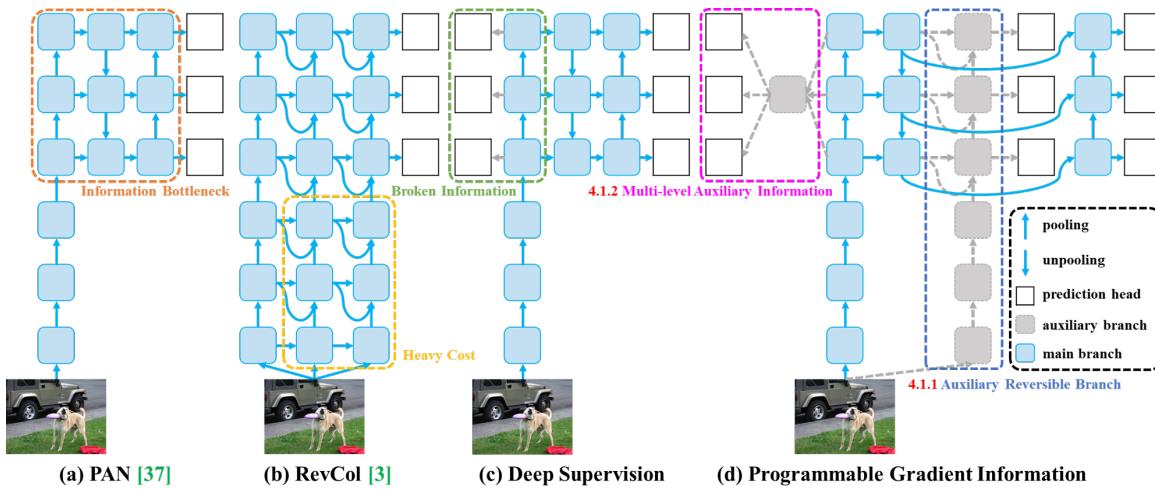


Figura 3.9: PGI y arquitecturas de red y métodos relacionados. (a) Path Aggregation Network (PAN), (b) Reversible Columns (RevCol), (c) supervisión profunda convencional, y (d) método usado en YOLOv9 de Información de Gradiente Programable (PGI).

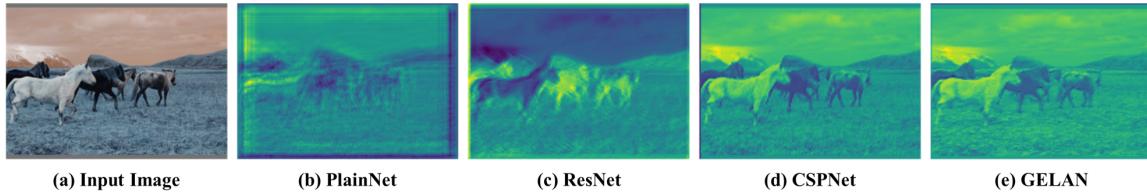


Figura 3.10: Resultados de visualización de mapas de características de salida con pesos iniciales aleatorios para diferentes arquitecturas de red: (a) imagen de entrada, (b) PlainNet, (c) ResNet, (d) CSPNet, y (e) la arquitectura propuesta GELAN.

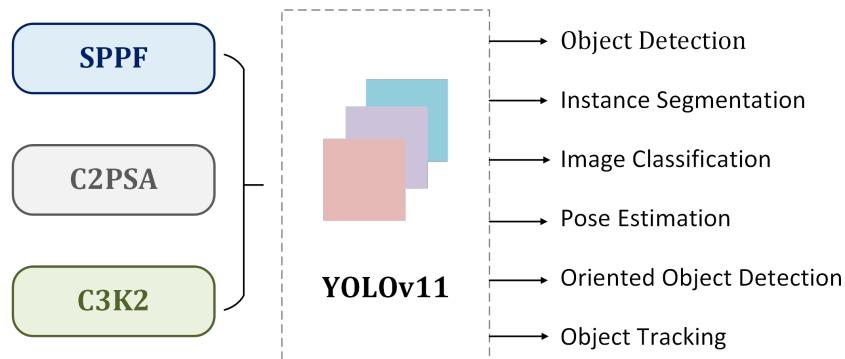


Figura 3.11: Módulos arquitectónicos clave de YOLO11.

3.3. Hiperparámetros de entrenamiento

La configuración adecuada de los hiperparámetros resulta determinante para alcanzar un desempeño óptimo en modelos de aprendizaje automático. Dichos parámetros, definidos antes del inicio del proceso de entrenamiento, influyen notablemente en la velocidad de convergencia, la estabilidad del algoritmo y la capacidad del modelo para generalizar frente a nuevos datos.

3.3.1. Optimizador

Los optimizadores de entrenamiento son algoritmos utilizados en el proceso de entrenamiento de modelos en aprendizaje de máquina para minimizar la función de pérdida ajustando los pesos y sesgos del modelo. Su utilidad principal es mejorar la convergencia del modelo hacia un mínimo local o global en el espacio de la función de costo, lo que se traduce en una mejor predicción [18]. Algunos ejemplos comunes incluyen:

- **Gradiente Descendente (SGD)**: Realiza ajustes basados en la derivada del error respecto a los pesos.
- **RMSProp**: Emplea una media móvil de gradientes pasados para normalizar el valor de la derivada, mejorando la estabilidad.
- **Adam**: Combina las ventajas del momentum y RMSProp, adaptando la actualización de cada parámetro individualmente.

A pesar de que los algoritmos adaptativos, como Adam, suelen acelerar la convergencia, diversas investigaciones han señalado que, en ciertos escenarios, Adam puede generar soluciones con menor capacidad de generalización en comparación con SGD [19-22]. Además existe precedente en trabajos previos de segmentación de instancias con YOLO, donde se ha observado que SGD supera a Adam [23, 24].

3.3.2. Técnicas de regularización

Las técnicas de regularización se aplican durante el entrenamiento para contrarrestar el sobreajuste, al imponer restricciones adicionales en la función de costo [18]. Entre las técnicas más comunes se encuentran:

- **L1 y L2:** Penalizan pesos elevados, utilizando normas basadas en L1 y L2 respectivamente.
- **Dropout:** Inhabilita aleatoriamente una fracción de neuronas durante el entrenamiento, para evitar que el modelo dependa demasiado de algunas características.
- **Early Stopping:** Finaliza el entrenamiento cuando la mejora en la función de pérdida se estanca, evitando sobre-entrenar el modelo.
- **Data Augmentation:** Aumenta la diversidad de muestras de entrenamiento mediante transformaciones (rotación, escalado, ruido, etc.).

3.3.3. Ajustador de la tasa de aprendizaje

Los ajustadores de la tasa de aprendizaje —en inglés “learning rate schedulers”— son técnicas que modifican dinámicamente la tasa de aprendizaje a medida que avanza el entrenamiento. Esta adaptación permite mejorar la convergencia y evitar que el modelo quede atrapado en mínimos locales oscilando alrededor de la solución óptima [25]. Entre los métodos más comunes destacan:

- **Step Decay:** Reduce la tasa de aprendizaje en intervalos predefinidos.
- **Exponential Decay:** Disminuye la tasa de aprendizaje de forma exponencial a lo largo del entrenamiento.
- **Cosine Annealing:** Aplica una disminución de la tasa de aprendizaje siguiendo una función cosenoide para lograr una convergencia más suave.

3.3.4. Tamaño de lote

Estudios recientes han indicado que el tamaño del lote, si bien no afecta significativamente el rendimiento final, debe ajustarse en conjunto con otros hiperparámetros (como la tasa de aprendizaje) para optimizar el entrenamiento. En la búsqueda de hiperparámetros, se ha evidenciado que lotes de tamaño intermedio facilitan un ajuste más rápido y eficiente [26, 27].

3.4. Técnicas de optimización

Diversas técnicas se han desarrollado para optimizar modelos de aprendizaje automático, ya sea mediante la reducción del costo computacional, la disminución del tiempo de inferencia o la reducción en consumo energético. Entre las técnicas más populares se destacan:

3.4.1. Pruning (poda)

La poda comprende la eliminación de conexiones, pesos o neuronas redundantes en un modelo entrenado, con el fin de reducir la complejidad del modelo. Esta técnica requiere un manejo cuidadoso, ya que la eliminación indiscriminada de parámetros puede deteriorar la precisión [28].

3.4.2. Cuantización numérica

La cuantización consiste en representar los pesos y activaciones utilizando menos bits (por ejemplo, pasar de 32 bits flotantes a 8 bits enteros). Esta técnica reduce el tamaño del modelo y disminuye el consumo de memoria, siendo especialmente útil en dispositivos con recursos limitados, como teléfonos móviles o sistemas embebidos [29]. Se distinguen dos enfoques principales:

- **Post-Training Quantization (PTQ)**: se aplica la cuantización después del entrenamiento, lo cual es más rápido pero puede inducir pérdida de precisión.
- **Quantization Aware Training (QAT)**: el modelo es entrenado teniendo en cuenta la cuantización, simulando la precisión reducida durante el entrenamiento. Esto permite que el modelo se ajuste mejor a las restricciones de cuantización, obteniendo mejores resultados.

En grandes modelos de lenguaje se observa que el impacto negativo de la cuantización disminuye conforme aumenta el tamaño del modelo [30]. Si bien aún no se ha evidenciado de forma concluyente un comportamiento similar en tareas de visión por computadora, se piensa que la tendencia podría ser comparable.

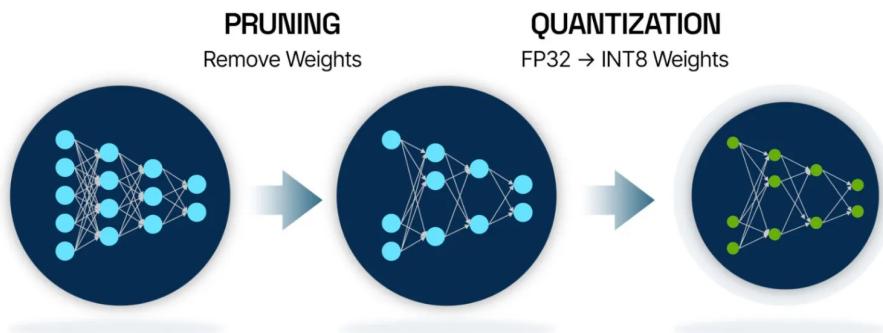


Figura 3.12: Representación de una red optimizada con poda y cuantización.

3.4.3. Knowledge Distillation

Esta técnica implica entrenar un modelo más pequeño y eficiente (llamado modelo estudiante) utilizando las predicciones de un modelo más grande y preciso (modelo maestro). El estudiante aprende a imitar el comportamiento del maestro, lo que permite reducir el tamaño del modelo manteniendo un buen rendimiento [28].

3.5. Seguimiento

En su forma más simple, el seguimiento puede definirse como el problema de estimar la trayectoria de un objeto en el plano de la imagen a medida que se desplaza por una escena. En otras palabras, un rastreador asigna etiquetas coherentes a los objetos rastreados en diferentes fotogramas de un vídeo. Además, dependiendo del dominio de seguimiento, un rastreador también puede proporcionar información centrada en el objeto, como la orientación, el área o la forma de un objeto [31].

Según [31], Yilmaz et al.] es posible clasificar taxonómicamente el seguimiento según el método empleado, donde distinguen basado en el modelo de representación utilizado para el objeto de interés.

- **Seguimiento por punto:** los objetos detectados se representan mediante puntos, y la asociación de los puntos se basa en el estado anterior —o futuro— del objeto, que puede incluir su posición y movimiento. Este método requiere detecciones previas.
- **Seguimiento por plantilla (*kernel*):** se rastrean objetos basándose en modelos de apariencia del objeto de interés. Estos modelos de apariencia pueden ser plantillas, estimaciones de densidad de aparición o regiones de objeto (por ejemplo cajas englobantes). Este método puede utilizar detecciones previas o ser la forma en que se realizan detecciones futuras.
- **Seguimiento por silueta:** aquí el modelo de apariencia corresponde a un contorno o silueta del objeto de interés. Similar al seguimiento por plantilla es posible utilizar detecciones previas (“shape matching”) o realizar nuevas detecciones (“contour tracking”).

Otra forma de categorizar la tarea de seguimiento es según su implementación basada o no en detecciones.

- **Seguimiento basado en detección:** implica detectar objetos en cada fotograma de una secuencia de video mediante un detector y luego usar algoritmos de asociación de datos (como SORT o DeepSORT), para vincular estas detecciones a través de los fotogramas.
- **Seguimiento basado en modelos:** utiliza un modelo predefinido del objeto, el cual puede ser predeterminado o aprendido, simple (forma, color, etc.) o complejo (representación semántica o basada en contexto) y lo ajusta en cada fotograma. En resumen se realiza detección al mismo tiempo que se sigue al objeto.

3.5.1. Algoritmos de seguimiento relevantes

Entre los algoritmos desarrollados para el seguimiento de objetos destacan:

- **BoT-SORT:** algoritmo de seguimiento de múltiples objetos que combina información sobre el movimiento, la apariencia y la Compensación del Movimiento de Cámara (CMC) [32].
- **OC-SORT:** el algoritmo Observation-Centric SORT (OC-SORT) [33] es una versión avanzada del algoritmo Simple Online and Real-Time Tracking (SORT) [34]. OC-SORT mejora la robustez ante occlusiones y movimientos no lineales al centrarse en las observaciones de los objetos en lugar de solo las estimaciones de estado.
- **Deep OC-SORT:** mejora el seguimiento de múltiples objetos integrando características visuales con técnicas basadas en el movimiento. Este modelo se basa en OC-SORT e introduce módulos clave como (CMC), la Apariencia Dinámica (DA) y la Ponderación Adaptativa (AW) [35].
- **Hybrid-SORT:** en Hybrid SORT [36], el vector de estado del filtro de Kalman se amplía para incluir los estados de confianza y altura, que representan la estimación de la confianza y el tamaño de la detección. Esto se utiliza para predecir la confianza y la altura de la detección mediante un modelado lineal basado en el historial de trayectorias.
- **ByteTrack:** este algoritmo es conocido por su capacidad para manejar tanto cajas de detección de alta puntuación como de baja puntuación. Emplea una estrategia de asociación de objetos en dos etapas, mejorando la precisión al incluir detecciones de baja puntuación como parte integral del proceso de asociación [37].

Trabajo a realizar

4. Trabajo realizado por WildSense y motivación de mejoras

Para entender la motivación detrás de este proyecto es importante conocer el proceso de estimación de biomasa implementado en WildSense, así como identificar debilidades y posibles soluciones. A continuación se explica el proceso, el cual además es ilustrado en la figura 4.13.

1. **Captura en video:** se registran videos de salmones mediante cámaras estéreo, obteniendo información visual, mapas de profundidad y nubes de puntos.
2. **Detección y segmentación:** las imágenes se someten a un procesamiento mediante modelos de segmentación de instancias, que generan predicciones referentes a la posición (detección) y al contorno (segmentación) de los peces presentes en la escena. Este procedimiento posibilita el aislamiento de información visual y espacial correspondiente a los objetos de interés —salmones en este caso— del fondo.
3. **Tracking:** con el fin de mitigar errores derivados de mediciones únicas, se implementa una estimación estadística basada en múltiples muestras, lo que requiere agrupar las predicciones correspondientes a cada pez. La agrupación se logra mediante algoritmos de seguimiento o “tracking”. Si bien en ocasiones la tarea de seguimiento se integra al proceso de detección en videos, en el caso de WildSense, se utilizan modelos YOLO que procesan imágenes de forma independiente sin incluir seguimiento. Por el elevado costo computacional que implicaría incorporar un modelo adicional específico para el seguimiento, se opta por utilizar la técnica de seguimiento basado en detección, aprovechando las detecciones ya obtenidas.
4. **Post-procesado:** se realiza filtrado de muestras deficientes. Estas constituyen casos donde el pez está incompleto, cubierto, muy alejado para realizar una estimación apropiada o el número de muestras rastreadas son muy pocas.
5. **Estimación de volumen:** finalmente se toman los datos de profundidad tomados con las cámaras estéreo y se emparejan a modelos tridimensionales creados en Blender, con los cuales es posible realizar estimaciones de volumen y masa.

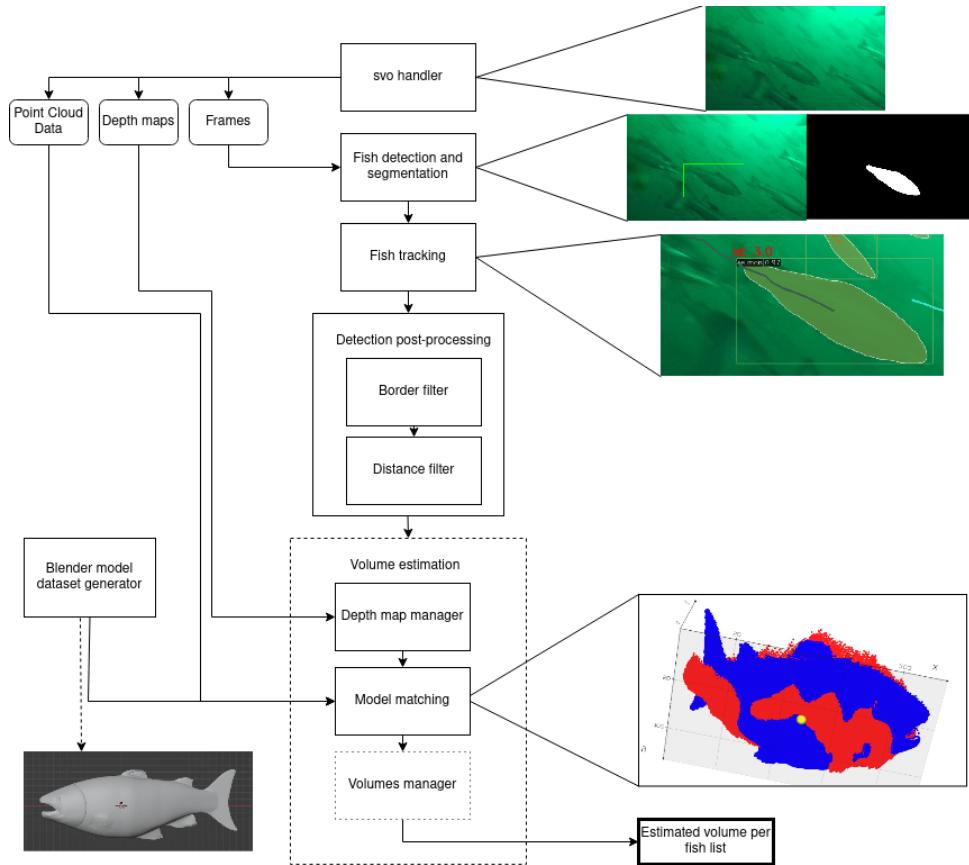


Figura 4.13: Diagrama de flujo con el proceso estimación de volumen y biomasa de salmones implementada por WildSense [2].

Cabe mencionar que las etapas de seguimiento y filtrado de detecciones resultan computacionalmente costosas y dado que muchas predicciones son eventualmente descartadas, sería preferible excluirlas desde un principio evitando segmentarlas del todo. Otro problema que existe es el “parpadeo”, que consiste en instancias segmentadas de forma intermitente a lo largo de un video, resultando en seguimientos truncados, lo que hace que sean excluidos de las estimaciones o, en caso de no serlo, reducen la información disponible por estimación, deteriorando su calidad.

Estos problemas justifican la creación de una base de datos que contenga únicamente instancias de interés, con el fin de disminuir el costo computacional en la etapa de post-procesado. Se espera además, que los modelos entrenados se especialicen en segmentar exclusivamente los peces de interés, reduciendo la omisión de instancias relevantes, en consecuencia, minimizando el “parpadeo”, mejorando la calidad del seguimiento y en consecuencia de las estimaciones.

5. Metodología

Para alcanzar los objetivos planteados en el proyecto se propone el siguiente flujo de trabajo:

1. Adquirir o desarrollar una base de datos orientada a la segmentación de instancias.
2. Adquirir o desarrollar una base de datos para tareas de seguimiento.
3. Llevar a cabo el entrenamiento de diversos modelos YOLO para segmentación de instancias, utilizando hiperparámetros de entrenamiento por defecto con algunas variaciones.
4. Realizar validación de los modelos previamente entrenados para obtener un “benchmark” inicial que compare los diferentes modelos en distintas circunstancias de entrenamiento, esto con el fin de encaminar la siguiente etapa.
5. Realizar la validación de los modelos entrenados con el fin de establecer un “benchmark” inicial que permita comparar el desempeño de los distintos modelos en diversas condiciones de entrenamiento y orientar la siguiente etapa.
6. Ejecutar una búsqueda de hiperparámetros para identificar la configuración óptima para el entrenamiento definitivo del modelo. Dicha búsqueda se restringe a los casos más prometedores identificados en la etapa anterior, con el objetivo de reducir el espacio de búsqueda debido a limitaciones de tiempo.
7. Proceder al entrenamiento final del modelo, empleando los hiperparámetros optimizados, y validar los resultados obtenidos.
8. Validar los modelos entrenados en la tarea de seguimiento.

Cada etapa conlleva desafíos específicos, requiere el uso de herramientas particulares y demanda el desarrollo correspondiente de código. Además, con el objetivo de facilitar la replicación de los experimentos y resultados obtenidos en este trabajo, se contempla de forma paralela el entrenamiento y validación utilizando la base de datos pública Deepfish; en dicha parte de la memoria no se modifica la base de datos ni se incluyen pruebas relacionadas con seguimiento.

6. Herramientas computacionales

En esta sección se discuten los programas y herramientas utilizadas en el desarrollado de esta memoria.

6.1. Herramientas de desarrollo y despliegue

Python se ha consolidado como el lenguaje de programación por excelencia para el desarrollo y despliegue de modelos de aprendizaje automático. Entre los “frameworks” principales destacan TensorFlow y PyTorch, las cuales se caracterizan por su robustez y adaptabilidad a diversas necesidades. Dado que esta memoria consiste en entrenar un modelo de segmentación y no modificar ni diseñar uno nuevo, es recomendable emplear una librería más sencilla para esta tarea.

Tabla 6.1: Tabla comparativa de distintas plataformas de desarrollo en Python.

Característica	Tensorflow [38, 39]	Pytorch [40, 41]	Supergradients [42]	Ultralytics [43]
Disponibilidad de modelos	✓ Múltiples modelos y backbones para usar. ✓ Permite customizar y diseñar modelos.	✓ Multiples modelos y backbones para usar. ✓ El mayor nivel de customización.	✓ Modelos de pose y de detección. ✗ No hay soporte para modelos de segmentación.	✓ Modelos YOLO para múltiples tareas. ✗ Poca a nula customización.
Entrenamiento	✓ Simple de realizar. ✓ Muy alto nivel de personalización.	✓ Algo complicado de realizar. ✓ El mayor nivel de personalización.	✓ Algo complicado de realizar. ✓ Alto nivel de personalización (en modelos soportados).	✓ Muy sencillo. ✓ Buen nivel de personalización.
Validación	✓ Implementación manual.	✓ Implementación manual.	✓ Implementación por interfaz.	✓ Implementación por interfaz.
Exportación con TensorRT	✓ Implementado.	✓ Implementado.	✗ Debe implementarse a parte utilizando Pytorch.	✓ Wrapp de Pytorch.
Dificultad.	Media	Alta	Media-alta	Fácil

6.1.1. Ultralytics

Ultralytics [43] es un “toolkit” construido sobre PyTorch, especializado en la implementación de la familia de modelos YOLO para detección, segmentación y estimación de pose en tiempo real. Proporciona “scripts” listos para usar, configuraciones predeterminadas optimizadas y facilidades para búsqueda de hiperparámetros, exportación a formatos ONNX y TensorRT, y despliegue en dispositivos borde. Su enfoque ligero y modular permite integrarlo fácilmente en “pipelines” de producción donde se requiera alta velocidad de inferencia y versatilidad en el entrenamiento. El mismo equipo de Ultralytics ha desarrollado algunos de los últimos modelos YOLO disponibles, y es esta disponibilidad de dichos modelos, al igual que la facilidad en sus entrenamientos, la razón por la que se decide usar esta biblioteca por sobre otras alternativas —como SuperGradients.

6.2. Herramientas de etiquetado

Tabla 6.2: Tabla comparativa de distintas plataformas de etiquetado.

Plataforma	Bbox	Segmentación	Tracking	Pros	Contras
Roboflow [44]	✓ Manual ✗ Asistida	✓ Manual ✓ Asistida	✗ No disponible.	- Interfaz simple. - Permite entrenar modelos dentro de su plataforma.	- Hay que pagar para tener el dataset privado.
CVAT [45]	✓ Manual ✗ Asistida	✓ Manual ✓ Asistida	✓ Manual ✓ Asistida con bbox.	- Interfaz simple. - Corre de forma local.	- Puede tener incompatibilidad con Firefox. - Segmentación asistida lenta.
V7 Labs [46]	✓ Manual ✓ Asistida	✓ Manual ✓ Asistida	✓ Manual ✓ Asistida con segmentación.	- Permite entrenar modelos dentro de su plataforma.	- Interfaz muy lenta y con lag.
Labeller [47]	✓ Manual ✓ Asistida	✓ Manual ✓ Asistida	✓ Manual ✗ Asistida	- Permite entrenar modelos dentro de su plataforma.	- Interfaz complicada y difícil de usar.
Supervisely [48]	✓ Manual ✓ Asistida	✓ Manual ✓ Asistida	✓ Manual ✓ Asistida	- Múltiples apps y opciones para configurar los proyectos.	- Tracking asistido lento.
SAM2 [49]	✓ Asistida	✓ Asistida	✓ Asistida	- Se ejecuta localmente	- Implementación manual.

6.2.1. CVAT

CVAT [45] (Computer Vision Annotation Tool) es una herramienta de código abierto diseñada para facilitar la anotación de imágenes y videos. Soporta la creación de cajas englobantes, polígonos, y keypoints de manera manual, así como la segmentación de instancias y seguimiento de objetos en videos. CVAT es conocido por su simplicidad y velocidad, lo que lo convierte en una opción ideal para proyectos que requieren anotaciones precisas. Sin embargo, la segmentación automática puede ser lenta en datasets grandes, y pueden existir incompatibilidades con el navegador Firefox.

7. Resultados

8. Conclusión

8.1. Trabajo Futuro

Referencias

- [1] WildSense. «Robótica Submarina Automatizada para Salmonicultura de Alta Mar (R-SASA)». Disp. desde: <<https://wildsense.ai/servicios/robotica-submarina-automatizada-para-salmonicultura-de-alta-mar-r-sasa>> [visitado 25 de mayo de 2025].
- [2] WildSense. *Memoria Anual 2024*, [diapositiva]. Valparaíso, Chile, 2024. Diagrama de flujo en la página 10 e imágenes en la página 14. Disp. desde: <<https://wildsense.ai/wp-content/uploads/2024/01/Memoria-Anual-2024.pdf>> [visitado 25 de mayo de 2025].
- [3] United Nations Food and Agriculture Organization (FAO). *The State of World Fisheries and Aquaculture 2019 (SOFIA): Meeting the Sustainable Development Goals*. Rome, Italy, FAO, 2018. Disp. desde: <<https://openknowledge.fao.org/handle/20.500.14283/9540es>>.
- [4] Gob. de Chile. «Levantamiento de Información de Pisciculturas en Chile y su Incorporación a la IDE de la División de Acuicultura». Fondo de Investigación Pesquera; Ministerio de Economía, Fomento y Turismo, (inf. téc.). 2017.
- [5] N. Tonachella [et al]. «An affordable and easy-to-use tool for automatic fish length and weight estimation in mariculture». *Scientific Reports*, vol. 12 (1): pág. 15642, 19 de sep. de 2022. ISSN: 2045-2322. DOI: 10.1038/s41598-022-19932-9.
- [6] M. S. Ahmed, T. T. Aurpa y M. A. Kalam Azad. «Fish Disease Detection Using Image Based Machine Learning Technique in Aquaculture». *Journal of King Saud University - Computer and Information Sciences*, vol. 34 (8, Part A): págs. 5170-5182, 2022. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.05.003.
- [7] K. He [et al]. «Mask R-CNN». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42 (2): págs. 386-397, 6 de jun. de 2018. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2844175.
- [8] S. Ren [et al]. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

- gence, vol. 39 (6): págs. 1137-1149, 1 de jun. de 2017. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2016.2577031.
- [9] Z. Huang [et al]. «Mask Scoring R-CNN». En: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (15-20 de jun. de 2019; Long Beach, CA, USA). Págs. 6402-6411. IEEE, 2020. ISBN: 978-1-7281-3293-8. DOI: 10.1109/CVPR.2019.00657.
- [10] X. Wang [et al]. «SOLO: Segmenting Objects by Locations». En: *Computer Vision – ECCV 2020* (23-28 de ago. de 2020; Glasgow, UK). Págs. 649-665, Cham. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-58523-5_38.
- [11] X. Wang [et al]. «SOLOv2: Dynamic and Fast Instance Segmentation». En: *Advances in Neural Information Processing Systems*. Págs. 17721-17732, vol. 33. Curran Associates, Inc., 2020. Disp. desde: <https://proceedings.neurips.cc/paper_files/paper/2020/file/cd3afef9b8b89558cd56638c3631868a-Paper.pdf>.
- [12] G. Jocher, A. Chaurasia y J. Qiu. «Ultralytics YOLOv8». Disp. desde: <<https://github.com/ultralytics/ultralytics>>.
- [13] M. Sohan, T. Ram y V. Ch. «A Review on YOLOv8 and Its Advancements». En: *Data Intelligence and Cognitive Informatics* (27-28 de jun. de 2023; SCAD College of Engineering and Technology, Tirunelveli, India). Págs. 529-545. Springer Nature Singapore, 2024. DOI: 10.1007/978-981-99-7962-2_39.
- [14] X. Zhao [et al]. «Fast Segment Anything». *ArXiv*, Beijing, China. 21 de jun. de 2023. DOI: 10.48550/arXiv.2306.12156.
- [15] C.-Y. Wang y H.-Y. M. Liao. «YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information». *ArXiv*. 29 de feb. de 2024. DOI: 10.48550/arXiv.2402.13616.
- [16] G. Jocher y J. Qiu. «Ultralytics YOLO11». Disp. desde: <<https://github.com/ultralytics/ultralytics>>.
- [17] R. Khanam y M. Hussain, *YOLOv11: An Overview of the Key Architectural Enhancements*, 23 de oct. de 2024. DOI: 10.48550/arXiv.2410.17725.

- [18] I. Goodfellow, Y. Bengio y A. Courville. *Deep Learning*. MIT Press, 2016. Disp. desde: <<http://www.deeplearningbook.org>>.
- [19] N. S. Keskar y R. Socher. «Improving Generalization Performance by Switching from Adam to SGD». *ArXiv*. Dic. de 2017. DOI: 10.48550/arXiv.1712.07628.
- [20] J. Deng [et al]. «ImageNet: A large-scale hierarchical image database». En: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (20-25 de jun. de 2009; Miami, FL, USA). Págs. 248-255, 2009. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.
- [21] P. Goyal [et al]. «Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour». *ArXiv*. 2017. DOI: 10.48550/arXiv.1706.02677.
- [22] K. He [et al]. «Deep Residual Learning for Image Recognition». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (27-30 de jun. de 2016; Las Vegas, NV, USA). Págs. 770-778. IEEE, 2016. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90.
- [23] A. F. Guerrero Loyola. «Análisis del modelo YOLO-V8 para la segmentación de salmones en jaulas de cultivo en proceso estimador de biomasa, en tiempo real». Memoria de titulación (Ing. Civil Electrónica). Valparaíso, Chile, UTFSM, feb. de 2024. Advisor: Marcos Zuñiga; Prof. Correferente: Gonzalo Carvajal. Disp. desde: <<https://repositorio.usm.cl/handle/123456789/74331>>.
- [24] J. E. López Blanche. «Segmentación de Salmones para estimación de masa en salmonicultura.», [Informe *Benchmark* del proyecto final en *Visión por Computador*]. Disp. desde: <https://github.com/juliopchile/CV_Project> [visitado 18 de jun. de 2025].
- [25] Y. Bengio. «Practical recommendations for gradient-based training of deep architectures». *Arxiv*. 24 de jun. de 2012. DOI: 10.48550/arXiv.1206.5533.
- [26] C. Shallue [et al]. «Measuring the Effects of Data Parallelism on Neural Network Training». *Journal of Machine Learning Research (JMLR)*. 19 de jul. de 2019. DOI: 10.48550/arXiv.1811.03600.

- [27] T. Breuel. «The Effects of Hyperparameters on SGD Training of Neural Networks». *ArXiv*. Ago. de 2015.
- [28] Y. Cheng [et al]. «Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges». *IEEE Signal Processing Magazine*, vol. 35 (1): págs. 126-136. ISSN: 1558-0792. DOI: 10.1109/MSP.2017.2765695.
- [29] B. Jacob [et al]. «Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference». En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (18 de jun.-23 de mayo de 2018; Salt Lake City, UT, USA). Págs. 2704-2713. IEEE, 2018. DOI: 10.1109/CVPR.2018.00286.
- [30] Z. Yao [et al]. «Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation». *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38: págs. 19377-19385, mar. de 2024. DOI: 10.1609/aaai.v38i17.29908.
- [31] A. Yilmaz, O. Javed y M. Shah. «Object tracking: a survey. ACM Comput Surv». *ACM Comput. Surv.*, vol. 38. Dic. de 2006. DOI: 10.1145/1177352.1177355.
- [32] J. Zhao y J. Chen. «YOLOv8 Detection and Improved BOT-SORT Tracking Algorithm for Iron Ladles». En: *Proceedings of the 2024 7th International Conference on Image and Graphics Processing* (19-21 de ene. de 2024; Beijing, China). Págs. 409-415, New York, NY, USA. Association for Computing Machinery, 2024. ISBN: 9798400716720. DOI: 10.1145/3647649.3647713.
- [33] J. Cao [et al]. «Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking». En: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (17-24 de jun. de 2023; Vancouver, BC, Canada). Págs. 9686-9696. IEEE, 2023. ISBN: 979-8-3503-0129-8. DOI: 10.1109/CVPR52729.2023.00934.
- [34] A. Bewley [et al]. «Simple online and realtime tracking». En: *2016 IEEE International Conference on Image Processing (ICIP)* (25-28 de sep. de 2016; Phoenix, AZ, USA). Págs. 3464-3468. IEEE, 2016. ISBN: 978-1-4673-9961-6. DOI: 10.1109/ICIP.2016.7533003.

- [35] G. Maggiolino [et al]. «Deep OC-Sort: Multi-Pedestrian Tracking by Adaptive Re-Identification». En: *2023 IEEE International Conference on Image Processing (ICIP)* (8-11 de oct. de 2023; Kuala Lumpur, Malaysia). Págs. 3025-3029. IEEE, 2023. ISBN: 978-1-7281-9835-4. DOI: [10.1109/ICIP49359.2023.10222576](https://doi.org/10.1109/ICIP49359.2023.10222576).
- [36] M. Yang [et al]. «Hybrid-SORT: Weak Cues Matter for Online Multi-Object Tracking». En: *Proceedings of the AAAI conference on artificial intelligence* (7; 20-27 de feb. de 2024). Págs. 6504-6512, vol. 38, 2024. DOI: [10.1609/aaai.v38i7.28471](https://doi.org/10.1609/aaai.v38i7.28471).
- [37] Y. Zhang [et al]. «ByteTrack: Multi-object Tracking by Associating Every Detection Box». En: *Computer Vision – ECCV 2022* (23-27 de oct. de 2022; Tel Aviv, Israel). Págs. 1-21, Cham, Switzerland. Springer Nature, 2022. DOI: [10.1007/978-3-031-20047-2_1](https://doi.org/10.1007/978-3-031-20047-2_1).
- [38] TensorFlow Developers. «TensorFlow», [Software]. DOI: [10.5281/zenodo.4724125](https://doi.org/10.5281/zenodo.4724125).
- [39] M. Abadi [et al]. «TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems». Google, (Whitepaper). 9 de nov. de 2015. Disp. desde: <<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf>>.
- [40] D. Bergmann y C. Stryker. «What is PyTorch?» Disp. desde: <<https://www.ibm.com/think/topics/pytorch>> [visitado 1 de jul. de 2025].
- [41] J. Ansel [et al]. «PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation». En: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)* (27 de abr.-1 de mayo de 2024; La Jolla, CA, USA). Págs. 929-947, New York, NY, USA. Association for Computing Machinery, 2024. ISBN: 9798400703850. DOI: [10.1145/3620665.3640366](https://doi.org/10.1145/3620665.3640366).
- [42] S. Aharon [et al]. «Super-Gradients», [Software]. GitHub. DOI: [10.5281/ZENODO.7789328](https://doi.org/10.5281/ZENODO.7789328).
- [43] G. Jocher, J. Qiu y A. Chaurasia. «Ultralytics YOLO». Disp. desde: <<https://github.com/ultralytics/ultralytics>>.

- [44] B. Dwyer [et al]. «Roboflow (Version 1.0)», [Software]. Disp. desde: <<https://roboflow.com>>. Computer vision.
- [45] C. Corporation. «Computer Vision Annotation Tool (CVAT)». Zenodo. DOI: 10.5281/zenodo.12771595.
- [46] «V7 Labs - Darwin», [Computer Vision Tools]. Disp. desde: <<https://www.v7labs.com/darwin>>.
- [47] «Labellerr», [Computer Vision Tools]. Disp. desde: <<https://www.labellerr.com>>.
- [48] Supervisely. «Supervisely Computer Vision platform», [Computer Vision Tools]. Supervisely. Disp. desde: <<https://supervisely.com>>.
- [49] N. Ravi [et al]. «SAM 2: Segment Anything in Images and Videos». ArXiv. 2024. DOI: 10.48550/arXiv.2408.00714.
- [50] A. M. Hafiz y G. M. Bhat. «A survey on instance segmentation: state of the art». International Journal of Multimedia Information Retrieval, vol. 9 (3): págs. 171-189, 1 de sep. de 2020. ISSN: 2192-662X. DOI: 10.1007/s13735-020-00195-x.
- [51] R. Szeliski. *Computer Vision: Algorithms and Applications*. 2.^a ed. Springer Cham, 3 de ene. de 2022, (Texts in Computer Science). DOI: 10.1007/978-3-030-34372-9.
- [52] D. Lowe. «Object recognition from local scale-invariant features». En: *Proceedings of the Seventh IEEE International Conference on Computer Vision* (20-27 de sep. de 1999; Kerkyra, Greece). Págs. 1150-1157, vol. 2. IEEE, 2002. DOI: 10.1109/ICCV.1999.790410.
- [53] N. Dalal y B. Triggs. «Histograms of oriented gradients for human detection». En: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (20-25 de jun. de 2005; San Diego, CA, USA). Págs. 886-893, vol. 1. IEEE, 2005. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177.
- [54] J. Sivic y A. Zisserman. «Video Google: a text retrieval approach to object matching in videos». En: *Proceedings Ninth IEEE International Conference on Computer Vision* (13-16 de nov. de 2003; Nice, France). Págs. 1470-1477, vol. 2. IEEE, 2008. ISBN: 0-7695-1950-4. DOI: 10.1109/ICCV.2003.1238663.

- [55] F. Perronnin, J. Sánchez y T. Mensink. «Improving the Fisher Kernel for Large-Scale Image Classification». En: Computer Vision – ECCV 2010 (5-11 de sep. de 2010; Heraklion, Crete, Greece). *Proceedings of IEEE European Conference on Computer Vision, 2010* (págs. 143-156; vol. 6314; Berlin, Heidelberg). Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-15561-1_11.
- [56] A. Krizhevsky, I. Sutskever y G. E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Advances in Neural Information Processing Systems*. Págs. 1097-1105, vol. 25. Curran Associates, Inc., 2012. EID: 534. ISBN: 9781627480031. Disp. desde: <https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
- [57] K. Simonyan y A. Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». En: 3rd International Conference on Learning Representations (ICLR 2015) (7-9 de mayo de 2015; San Diego, CA, USA). *ICLR 2015 Conference Track Proceedings* (págs. 1-14). Computational y Biological Learning Society, 2015. DOI: 10.48550/arXiv.1409.1556.
- [58] K. He [et al]. «Deep Residual Learning for Image Recognition». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (27-30 de jun. de 2016; Las Vegas, NV, USA). Págs. 770-778. IEEE, 2016. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90.
- [59] G. Huang [et al]. «Densely Connected Convolutional Networks». En: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (21-26 de jul. de 2017; Honolulu, HI, USA). Págs. 2261-2269. IEEE, 2017. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.243.
- [60] C. Szegedy [et al]. «Going deeper with convolutions». En: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (7-12 de jun. de 2015; Boston, MA, USA). Págs. 1-9. IEEE, 2015. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298594.
- [61] P. Sermanet [et al]. «OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks». International Conference on Learning Representations.

- tions (ICLR) (Banff), 719 Broadway, 12th Floor, New York, NY 10003. 24 de feb. de 2014. DOI: [10.48550/arXiv.1312.6229](https://doi.org/10.48550/arXiv.1312.6229).
- [62] M. D. Zeiler y R. Fergus. «Visualizing and Understanding Convolutional Networks». En: *Computer Vision – ECCV 2014*. Págs. 818-833, Cham. Springer International Publishing, 2014. ISBN: 978-3-319-10590-1. Disp. desde: <<https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf>>.
- [63] J. Redmon [et al]. «You Only Look Once: Unified, Real-Time Object Detection». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (27-30 de jun. de 2016; Las Vegas, NV, USA). Págs. 779-788. IEEE, 2016. ISBN: 978-1-4673-8851-1. DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [64] R. Girshick [et al]. «Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation». En: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (23-28 de jun. de 2014; Columbus, OH, USA). Págs. 580-587. IEEE, 2014. ISBN: 978-1-4799-5118-5. DOI: [10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81).
- [65] R. Girshick. «Fast R-CNN». En: *2015 IEEE International Conference on Computer Vision (ICCV)* (7-13 de dic. de 2015; Santiago, Chile). Págs. 1440-1448. IEEE, 2016. ISBN: 978-1-4673-8391-2. DOI: [10.1109/ICCV.2015.169](https://doi.org/10.1109/ICCV.2015.169).
- [66] S. Liu [et al]. «Path Aggregation Network for Instance Segmentation». En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (18-23 de jun. de 2018; Salt Lake City, UT, USA). Págs. 8759-8768. IEEE, 2018. ISBN: 978-1-5386-6420-9. DOI: [10.1109/CVPR.2018.00913](https://doi.org/10.1109/CVPR.2018.00913).
- [67] D. Bolya [et al]. «YOLACT: Real-Time Instance Segmentation». En: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (27 de oct.-2 de nov. de 2019; Seoul, Korea). Págs. 9156-9165. IEEE, 2020. ISBN: 978-1-7281-4803-8. DOI: [10.1109/ICCV.2019.00925](https://doi.org/10.1109/ICCV.2019.00925).
- [68] M. Bai y R. Urtasun. «Deep Watershed Transform for Instance Segmentation». *CoRR*. 24 de nov. de 2016. DOI: [10.48550/arXiv.1611.08303](https://doi.org/10.48550/arXiv.1611.08303).
- [69] P. O. Pinheiro, R. Collobert y P. Dollár. «Learning to segment object candidates». En: *Proceedings of the 29th International Conference on Neural Information Processing Systems* (ICNN) (Montreal, Quebec, Canada). Págs. 1026-1034. MIT Press, 2015. ISBN: 978-0-262-32900-0. DOI: [10.1162/9780262329000.001](https://doi.org/10.1162/9780262329000.001).

- sing Systems - Volume 2* (1 de jun. de 2015; Montreal, Canada). Págs. 1990-1998, Cambridge, MA, USA. MIT Press, 2015. DOI: [10.48550/arXiv.1506.06204](https://doi.org/10.48550/arXiv.1506.06204).
- [70] J. Dai [et al]. «Instance-Sensitive Fully Convolutional Networks». En: *Computer Vision – ECCV 2016* (11-14 de oct. de 2016; Amsterdam, The Netherlands). Págs. 534-549, vol. 9910, Cham. Springer International Publishing, 2016. ISBN: 978-3-319-46466-4. DOI: [10.1007/978-3-319-46466-4_32](https://doi.org/10.1007/978-3-319-46466-4_32).
- [71] X. Chen [et al]. «TensorMask: A Foundation for Dense Object Segmentation». En: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (27 de oct.-2 de nov. de 2019; Seoul, Korea). Págs. 2061-2069, 2020. ISBN: 978-1-7281-4803-8. DOI: [10.1109/ICCV.2019.00215](https://doi.org/10.1109/ICCV.2019.00215).
- [72] L. Yang, Y. Fan y N. Xu. «Video Instance Segmentation». En: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (Seoul, Korea). Págs. 5187-5196. IEEE, 2020. DOI: [10.1109/ICCV.2019.00529](https://doi.org/10.1109/ICCV.2019.00529).
- [73] P. Voigtlaender [et al]. «MOTS: Multi-Object Tracking and Segmentation». En: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (15-20 de jun. de 2019; Long Beach, CA, USA). Págs. 7934-7943. IEEE, 2020. DOI: [10.1109/CVPR.2019.00813](https://doi.org/10.1109/CVPR.2019.00813).
- [74] T. Zhou [et al]. «A Survey on Deep Learning Technique for Video Segmentation». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45 (6): págs. 7099-7122, 1 de jun. de 2023. ISSN: 1939-3539. DOI: [10.1109/TPAMI.2022.3225573](https://doi.org/10.1109/TPAMI.2022.3225573).

Anexos

9. Estado del arte en modelos de segmentación de instancias

En la presente sección se añade material suplementario relativo al estado del arte en técnicas de segmentación de instancias, material que no fue incorporada al corpus principal debido a restricciones de espacio y a su no esencialidad para la comprensión integral del proyecto. Aquí se adopta un enfoque más histórico y categórico enfocado en aspectos generales de la segmentación en visión por computadora, en contraposición a la descripción detallada de las propiedades de modelos específicos en la parte de introducción. La fuente principal de información para esta sección es el trabajo «A Survey on Instance Segmentation: State of the art» [50, Hafiz y Bhat 2020].

9.1. Técnicas tradicionales para segmentación de imágenes

Previo a la adopción de las redes neuronales, las técnicas de segmentación se fundamentaban en métodos de procesamiento de imágenes y en el análisis de características visuales. Entre las técnicas descritas en «Computer Vision: Algorithms and Applications» [51, Szeliski 2022], se incluyen aquellas basadas en umbralización, detección de bordes, crecimiento de regiones y segmentación por clústeres.

- **Umbralización:** consiste en separar los objetos del fondo mediante la asignación de un valor de intensidad fijo a los píxeles, útil en imágenes con contrastes pronunciados.
- **Detección de bordes:** utiliza operadores como Sobel, Canny o Laplaciano para identificar el contorno de los objetos, aprovechando los cambios bruscos en la intensidad de los píxeles.
- **Crecimiento de regiones:** consiste en agrupar píxeles adyacentes que comparten propiedades similares, tales como color o intensidad, con el fin de formar regiones coherentes.
- **Segmentación basada en clústeres:** algoritmos como K-means clasifican los píxeles en grupos con características similares, segmentando la imagen en varias partes.

Estas técnicas, aunque útiles en casos sencillos, eran limitadas en situaciones complejas, especialmente ante escenas con variaciones de iluminación o formas de objetos poco definidas.

9.2. Técnicas con redes neuronales

Con la llegada del aprendizaje profundo, especialmente las Redes Neuronales Convolucionales (CNNs), se han propuesto varios marcos para la tarea de segmentación de instancias.

En la evolución de técnicas de segmentación prima la buena representación de características de una imagen, esencial para la detección. En un inicio se intentó con el diseño de descriptores locales (como SIFT [52] y HOG [53]) y técnicas para agrupar y abstraer descriptores en representaciones de alto nivel que destacaran las partes discriminativas (como Bag of Words [54] y Fisher Vector [55]). El inconveniente de estos métodos de representación es que requerían una ingeniería manual compleja y conocimiento especializado. En cambio, los métodos de aprendizaje profundo, como las CNNs, aprenden automáticamente representaciones de características con distintos niveles de abstracción.

Luego la tendencia ha sido el de aumentar la profundidad de la redes neuronales. AlexNet [56] tenía 8 capas, VGGNet [57] 16 y ResNet [58] junto a DenseNet [59] más de 100. Fue con VGGNet y GoogLeNet [60] que se demostró que aumentar la profundidad de la red mejora su capacidad de representación. Algunas redes no tan profundas como AlexNet, OverFeat [61], ZFNet [62] y VGGNet, aún así tienen un gran numero de parámetros debido a sus capas densas, y estas implican un mayor coste computacional que las capas puramente convolucionales.

En la actualidad para problemas de clasificación de imágenes se prefieren Redes Neuronales Convolucionales Profundas (DCNNs) basadas en detectores como lo son Mask R-CNN [7] y YOLO [63]. Usualmente se utilizan las características extraídas en las últimas capas para la representación de los objetos.

9.3. Marcos de detección: dos etapas versus una etapa

Se presentan dos categorías principales al clasificar los marcos de detección según el número de etapas: los métodos basados en regiones (dos etapas) y los métodos unificados (una etapa).

- En el caso de plataformas con elevados recursos computacionales, los **marcos de dos etapas** ofrecen una mayor precisión. Esto se debe a que la mayoría de las técnicas que han obtenido mejores resultados en desafíos reconocidos se basan en este enfoque, el cual

proporciona una estructura más flexible y se adapta de manera óptima a la detección basada en regiones, tal como se observa en Mask R-CNN [7].

- Por otra parte, los **marcos de una sola etapa**, representados por ejemplos como YOLO [63], logran mayores velocidades al prescindir de preprocesamiento, utilizar una red de fondo más ligera, contar con un menor número de regiones candidatas y aprovechar una subred de detección totalmente convolucional. No obstante, se constata que presentan limitaciones para detectar objetos pequeños, contrariamente a lo que ocurre con los marcos de dos etapas.

9.3.1. Categorización taxonómica

Una alternativa de clasificación para modelos de segmentación es la categorización taxonómica según la técnica empleada. A continuación se explican las diferentes categorías taxonómicas según es descrito en [50, Hafiz y Bhat, 2020].

Tabla 9.3: Tabla taxonómica de técnicas de segmentación con sus fortalezas y debilidades.

Grupo	Técnicas	Fortalezas	Debilidades
Clasificación de propuestas de máscara.	RCNN [64], Fast RCNN [65], Faster RCNN [8].	- Relativamente simple de implementar. - Modesta precisión de segmentación.	- Lento y difícil de optimizar. - Problemas de almacenamiento, tiempo y escala de detección durante el entrenamiento. - Pruebas lentas. - No adecuado para aplicaciones en tiempo real.
Detección seguida por segmentación.	HTC, PANet [66], Mask RCNN [7], Mask Scoring RCNN [9], MPN, YOLACT [67], SOLOv2 [11].	- Relativamente simple de entrenar. - Mejor generalización. - Relativamente más rápido. - Buena precisión de segmentación.	- Depende de un pipeline de entrenamiento complicado, difícil de entrenar y optimizar.
Etiquetado de píxeles seguido por agrupamiento.	Deep Watershed Transform [68], Instance Cut.	- Utiliza algunas técnicas investigadas recientemente. - Técnicas relativamente más simples.	- Menor precisión de segmentación. - Cálculo intenso requiere alto poder computacional. - No adecuado para aplicaciones en tiempo real.
Métodos de deslizamiento denso de ventana	Deep Mask [69], Instance FCN [70], Tensor Mask [71].	-Área relativamente inexplorada. - Modesta precisión de segmentación.	- Usan algoritmos complejos. - Difícil de entrenar y optimizar. - No adecuado para aplicaciones en tiempo real.

- **Clasificación de máscara propuesta:** Esta técnica implica la generación de propuestas de máscaras, seguida de la clasificación de las propuestas generadas.
- **Detección seguida por segmentación:** Este popular enfoque implica la detección de objetos mediante una caja seguida de la segmentación objeto-caja.

- **Etiquetado de píxeles seguido por agrupamiento:** Este enfoque implica el etiquetado categórico de cada píxel de la imagen, luego los píxeles se agrupan en instancias de objetos mediante un algoritmo de agrupación.
- **Métodos de deslizamiento denso de ventana:** Se pasa una ventana de tamaño fijo a lo largo de toda la imagen para examinar cada región en detalle. En cada posición de la ventana, el modelo realiza una predicción para determinar qué clase de objeto o región pertenece a esa área.

9.4. Segmentación de instancias en vídeo

En 2019 [72, Yang et al] introducen la tarea de segmentación de instancias en video, “Video Instance Segmentation” (VIS), que consiste en detectar, segmentar y seguir instancias individuales de objetos en videos. Ese mismo año, [73, Voigtlaender et al.] amplian la tarea de seguimiento en múltiples objetos, “Multi-Object Tracking” (MOT), para incluir el seguimiento de segmentación de instancias, denominándolo “Multi-Object Tracking and Segmentation” (MOTS). En la literatura científica y a efectos prácticos VIS y MOTS corresponden a la misma tarea y pueden ser usados indistintamente.

A diferencia de la segmentación en imágenes, todas las técnicas conocidas de VIS implementan Redes Neuronales, siendo las más comunes CNNs. En un inicio se recurrió a modelos de segmentación, como Mask R-CNN, con la incorporación de capas extras, las cuales almacenaban información de cuadros anteriores, pero en la actualidad métodos más complejos existen. Las mayores innovaciones se dan en el uso Redes Convolucionales Recurrentes (RCNNs) para el manejo de datos secuenciales y captura de características temporales, Graph Neural Networks (GNNs) para modelar las relaciones entre distintas instancias para un mejor seguimiento y segmentación, y el uso de transformadores que permiten la tokenización de parches de video con mecanismos de atención capaces de detección y segmentación en imágenes [74].

Estás técnicas resuelven simultáneamente el seguimiento y segmentación de instancias. Lamentablemente el mayor problema con VIS es la dificultad del etiquetado en bases de datos para su entrenamiento. Es por este inconveniente que suele preferirse un acercamiento con modelos cuadro a cuadro.