
Universidad de Valladolid
E.T.S.I de Telecomunicación

Trabajo Final de Máster
Máster en Ingeniería de Telecomunicación

# Certificación y Validación de Modelos de Segmentación para la Detección de Tejidos en Imágenes de Pólipos Intestinales

Autor:
Jorge Ansótegui Gandul

Tutores:
Alfonso Bahillo Martínez
Mario Fernando Jojoa Acosta

20 de septiembre de 2024

# Resumen

*Este proyecto aborda el desarrollo y validación de un sistema basado en visión por computadora para la detección automática de pólipos intestinales en imágenes de endoscopia. La investigación se centra en el uso de dos modelos avanzados de segmentación: YOLOv8 y Detectron2, aplicados a un conjunto de imágenes de pólipos, recolectadas en el país de México y segmentadas por médicos expertos en el campo oncológico. Para ello, se han creado tres versiones del dataset, incluyendo preprocesamiento y técnicas de data augmentation, con el fin de maximizar la precisión de los modelos. El estudio compara el rendimiento de ambas arquitecturas, analizando métricas clave como el Intersection over Union (IoU), el Mean Average Precision (mAP) y el Recall; alcanzando un desempeño superior al 90 % en las mejores configuraciones. Además, se evalúan los tiempos de inferencia para determinar la viabilidad de detección en tiempo real. A partir de los resultados obtenidos, se ha realizado una comparativa de ambos modelos, determinando cuál es más adecuado según el tipo de aplicación: detección en tiempo real o segmentación precisa en entornos clínicos. El trabajo también sienta las bases para futuras investigaciones en este campo de investigación, con especial énfasis en la reproducibilidad del flujo de trabajo y la optimización de hiperparámetros para mejorar el rendimiento de los modelos y regularizar los mismos, es decir evitar overfitting.*

**Palabras Clave:** Segmentación, YOLOv8, Detectron2, visión por computadora, detección de pólipos.

# Abstract

*This project focuses on the development and validation of a computer vision system for the automatic detection of intestinal polyps in endoscopy images. The research explores two advanced segmentation models: YOLOv8 and Detectron2, applied to a set of polyp images, collected in the country of Mexico and segmented by expert doctors in the oncological field. Three versions of the dataset were created, incorporating preprocessing and data augmentation techniques to maximize model accuracy. The study compares the performance of both architectures, analyzing key metrics such as Intersection over Union (IoU), Mean Average Precision (mAP) and Recall; achieving over 90 % performance in the best configurations. In addition, inference times were evaluated to determine the feasibility of real-time detection. ased on the results obtained, a comparison of both models was made to determine which is better suited for each type of application: real-time detection or precise segmentation in clinical settings. This work also lays the foundation for future research in this field of reasearch, with a strong emphasis on workflow reproducibility and hyperparameter optimization to further enhance model performance and regularize them, that is, avoid overfitting.*

**Keywords:** Segmentation, YOLOv8, Detectron2, computer vision, polyp detection.

# Agradecimientos

*A mi tutor Alfonso Bahillo Martínez, cuya supervisión y ética de trabajo han sido fundamentales para la realización de este proyecto. Su guía ha sido una de las principales razones por las cuales este trabajo ha salido adelante.*

*A mi tutor Mario Fernando Jojoa Acosta, por proporcionarme el mejor asesoramiento académico y apoyo a lo largo de todo el proyecto. Su comunicación siempre agradable y su constante disponibilidad han sido invaluables.*

*Un agradecimiento especial a Cristián Castillo Olea por proporcionarnos las imágenes utilizadas en el proyecto.*

*Quiero expresar mi más profundo agradecimiento a mi familia, cuyo apoyo incondicional y constante ha sido mi mayor fortaleza. Gracias por su comprensión y ánimo en cada paso de este camino.*

# Índice general

**Índice de Figuras** **xi**

**Índice de Tablas** **xiii**

**1. Introducción** **1**
1.1. Motivación . . . . . 1
1.2. Introducción al Proyecto . . . . . 2
1.3. Objetivos del proyecto . . . . . 3
1.3.1. Objetivos Específicos . . . . . 3
1.4. Estructura del documento . . . . . 5

**2. Estado del Arte** **7**
2.1. Endoscopia: Procedimiento, Desafíos y Avances Tecnológicos . . . . . 8
2.2. Segmentación . . . . . 9
2.2.1. Diferencias entre Segmentación, Detección de Objetos e Instanciación 9
2.2.2. Comparativa de diferentes modelos de Segmentación . . . . . 10
2.2.3. Consideraciones y Limitaciones Actuales . . . . . 12

**3. Marco Teórico** **15**
3.1. Fundamentos de la Visión por Computadora . . . . . 15
3.1.1. Definición y conceptos básicos . . . . . 16
3.1.2. Procesamiento de imágenes digitales . . . . . 16
3.1.3. Técnicas de extracción de características . . . . . 17
3.2. Segmentación de Imágenes . . . . . 17
3.2.1. Técnicas Avanzadas de Segmentación . . . . . 18
3.2.1.1. Redes Neuronales Convolucionales (CNN) . . . . . 18

3.2.1.2. Segmentación Semántica . . . . . 19
3.2.1.3. Segmentación de Instancias . . . . . 20
3.2.1.4. Técnicas de Visualización y Explicabilidad . . . . . 20
3.2.2. Arquitectura YOLOv8 . . . . . 21
3.2.2.1. Visión General de la Arquitectura . . . . . 21
3.2.2.2. Componentes Principales . . . . . 21
3.2.2.3. Proceso de Detección . . . . . 23
3.2.2.4. Funciones de Pérdida . . . . . 23
3.2.2.5. Innovaciones y Mejoras . . . . . 24
3.2.2.6. Proceso de Segmentación en YOLOv8 . . . . . 25
3.2.2.7. Aplicaciones y Versatilidad . . . . . 25
3.2.3. Arquitectura Detectron2 . . . . . 25
3.2.3.1. Visión General de la Arquitectura Faster R-CNN FPN . 26
3.2.3.2. Flujo de Datos y Procesamiento . . . . . 27
3.2.3.3. Proceso de Segmentación en Detectron2 . . . . . 27
3.2.3.4. Ventajas y Aplicaciones . . . . . 28
3.2.4. Comparación entre YOLOv8 y Detectron2 . . . . . 29
3.2.4.1. Enfoque Arquitectónico . . . . . 29
3.2.4.2. Velocidad vs Precisión . . . . . 29
3.2.4.3. Flexibilidad y Personalización . . . . . 29
3.2.4.4. Casos de Uso . . . . . 30
3.2.4.5. Ecosistema y Soporte . . . . . 30
3.3. Métricas de Evaluación en Segmentación de Imágenes . . . . . 30
3.3.1. Precisión . . . . . 31
3.3.2. Recall . . . . . 32
3.3.3. F1-Score . . . . . 33
3.3.4. Índice de Jaccard (IoU) . . . . . 33
3.4. Optimización de Hiperparámetros . . . . . 34
3.4.1. Conceptos de hiperparámetros vs. parámetros . . . . . 34
3.4.2. Técnicas de búsqueda de hiperparámetros (grid search, random search) . . . . . 34
3.4.3. Optimización bayesiana . . . . . 35

**4. Metodología** **37**
4.1. Dataset . . . . . 37
4.1.1. Origen del *Dataset* . . . . . 37

4.1.2. Creación del *Dataset* en Roboflow . . . . . 37
4.1.2.1. Segmentación de las Imágenes . . . . . 38
4.1.2.2. Creación de Diferentes Versiones del *Dataset* . . . . . 38
4.1.3. Categorización del *Dataset* . . . . . 41
4.2. Entrenamiento de Modelos . . . . . 41
4.2.1. Hiperparámetros del Entrenamiento . . . . . 41
4.2.2. Optimización de Hiperparámetros mediante *Grid Search* Adaptativo 44
4.3. Evaluación de Modelos . . . . . 45
4.3.1. Creación de un Evaluador Personalizado . . . . . 47
4.4. Implementación en Vídeo a Tiempo Real . . . . . 48

**5. Resultados Y Discusión** **51**
5.1. Resultados YOLOv8 . . . . . 51
5.2. Resultados Detectron2 . . . . . 57
5.3. Análisis y Discusión de los resultados . . . . . 62
5.3.1. Comparación entre YOLOv8 y Detectron2 . . . . . 62
5.3.2. Mejor modelo del Estudio . . . . . 62
5.3.3. Evolución a lo largo de los intentos de entrenamiento . . . . . 65
5.3.4. Razonamiento sobre la no utilización de curvas de confianza . . . 66
5.3.5. Tiempos de inferencia y capacidad de detección en tiempo real . . 67
5.3.6. Correlación entre métricas y tiempos de inferencia . . . . . 68

**6. Conclusiones y Líneas Futuras** **69**
6.1. Conclusiones . . . . . 69
6.2. Líneas Futuras . . . . . 70

**Bibliografía** **73**

**A. Desarrollo e Implementación del Software** **79**
A.1. Preparación del Entorno de Trabajo . . . . . 79
A.1.1. Google Colab . . . . . 79
A.1.2. Hardware del Laboratorio . . . . . 79
A.1.3. Configuración del Entorno Local . . . . . 80
A.1.3.1. Instalación de Conda . . . . . 80
A.1.3.2. Creación y Activación del Entorno Virtual . . . . . 80
A.1.3.3. Instalación de Repositorios y Bibliotecas . . . . . 80
A.1.4. Diferenciación de Entornos y Scripts . . . . . 81

A.2. Revisión de los *Datasets* . . . . . 81
A.3. Código de Entrenamiento . . . . . 82
A.3.1. Entrenamiento de YOLOv8 . . . . . 82
A.3.2. Entrenamiento de Detectron2 . . . . . 83
A.4. Evaluación de los Modelos . . . . . 83

# Índice de Figuras


Figura 1.1. Diagrama de flujo de los objetivos del Proyecto . . . . . 4

Figura 3.1. Esquema detallado de la arquitectura YOLOv8 [1] . . . . . 22
Figura 3.2. Avances en rendimiento y precisión de YOLO [2] a lo largo de sus versiones . . . . . 24
Figura 3.3. Arquitectura detallada de Faster R-CNN FPN en Detectron2 [3] . . 26
Figura 3.4. Máscaras (Elaboración propia) . . . . . 31
Figura 3.5. Ilustración del cálculo del Recall (Elaboración propia) . . . . . 31
Figura 3.6. Ilustración del cálculo de la Precisión (Elaboración propia) . . . . . 32
Figura 3.7. Ilustración del cálculo de IoU (Elaboración propia) . . . . . 33

Figura 4.1. Ejemplo de una imagen del *dataset* original. . . . . 39
Figura 4.2. Ejemplo de una imagen del *dataset* preprocesado. . . . . 39
Figura 4.3. Ejemplo de una imagen del *dataset* con *data augmentation*. . . . . 40
Figura 4.4. Ejemplo visual de *Grid Search* con dos parametros [4] . . . . . 44
Figura 4.5. Mascaras binarias superpuestas, visualización de la categoría de cada píxel (Elaboración propia) . . . . . 48

Figura 5.1. Ejemplo del modelo Nano, para el *Dataset* básico . . . . . 54
Figura 5.2. Ejemplo del modelo Pequeño, para el *Dataset* básico . . . . . 54
Figura 5.3. Ejemplo del modelo Nano, para el *Dataset* con preprocesado . . . . . 55
Figura 5.4. Ejemplo del modelo Pequeño, para el *Dataset* preprocesado . . . . . 55
Figura 5.5. Ejemplo del modelo Nano, para el *Dataset* con *Data Augmentation* . 56
Figura 5.6. Ejemplo del modelo Pequeño, para el *Dataset Data Augmentation* . 56
Figura 5.7. Ejemplo del modelo Nano, para el *Dataset* básico . . . . . 59
Figura 5.8. Ejemplo del modelo Pequeño, para el *Dataset* básico . . . . . 59

Figura 5.9. Ejemplo del modelo Nano, para el *Dataset* preprocesado . . . . . 60
Figura 5.10. Ejemplo del modelo Pequeño, para el *Dataset* preprocesado . . . . . 60
Figura 5.11. Ejemplo del modelo Nano, para el *Dataset Data Augmentation* . . . 61
Figura 5.12. Ejemplo del modelo Pequeño, para el *Dataset Data Augmentation* . 61
Figura 5.13. Pérdida de segmentación durante el entrenamiento y validación para el modelo YOLOv8 Nano con el *Dataset* básico (Elaboración propia) 63
Figura 5.14. Evolución de las métricas de segmentación (Precisión, Recall y mAP50) durante el entrenamiento del modelo YOLOv8 Nano con el *Dataset* básico (Elaboración propia) . . . . . 64
Figura 5.15. Evolución con el paso de los intentos del estudio por *Grid Search* (Elaboración propia) . . . . . 65
Figura 5.16. Poca fiabilidad de las *boundig boxes* (Elaboración propia) . . . . . 67

# Índice de Tablas

Tabla 2.1. Estudio comparativo de diferentes Modelos de Segmentación [5] . . . . 12

Tabla 4.1. Hiperparámetros de estudio en YOLOv8. . . . . 42
Tabla 4.2. Hiperparámetros de estudio en *Detectron2*. . . . . 43

Tabla 5.1. Hiperparámetros y sus rangos de estudio para YOLOv8 . . . . . 51
Tabla 5.2. Valores Óptimos de Hiperparámetros para YOLOv8 en Diferentes *Datasets* 52
Tabla 5.3. Métricas del mejor intento con YOLOv8 para las diferentes configuraciones 53
Tabla 5.4. Hiperparámetros y sus rangos de estudio para Detectron2 . . . . . 57
Tabla 5.5. Valores Óptimos de Hiperparámetros para Detectron2 en Diferentes *Datasets* . . . . . 58
Tabla 5.6. Métricas del mejor intento con Detectron2 para diferentes configuraciones 58

CAPÍTULO 1

# Introducción

## 1.1. Motivación

El cáncer colorrectal continúa siendo una de las principales causas de mortalidad a nivel global [6], lo que subraya la importancia crítica de la detección temprana de pólipos intestinales como estrategia preventiva. En este contexto, la integración de tecnologías de inteligencia artificial, específicamente en el campo de la visión por computadora, presenta un potencial significativo para mejorar los procedimientos de detección actuales.

Los métodos convencionales de detección [7], como las endoscopias tradicionales, presentan limitaciones inherentes. Pólipos de tamaño reducido o de tipo plano pueden pasar inadvertidos, incluso para profesionales experimentados. Esta problemática motiva la exploración de sistemas basados en inteligencia artificial que puedan complementar y potenciar la capacidad de detección de los especialistas médicos.

La motivación principal de este proyecto radica en abordar estas limitaciones mediante la validación de un *dataset* de imágenes segmentadas de pólipos intestinales. Este enfoque se fundamenta en el uso de un *dataset* cuidadosamente anotado, en combinación con modelos avanzados de detección de objetos, puede mejorar significativamente la precisión en la identificación de pólipos.

Se reconoce que la transición desde la investigación hasta la implementación clínica es un proceso complejo y prolongado. No obstante, este proyecto aspira a sentar las bases para futuras investigaciones en el campo, proporcionando un recurso valioso para el desarrollo de herramientas de apoyo al diagnóstico médico. La creación de un sistema eficiente y

confiable para la detección en tiempo real podría, en última instancia, transformar la práctica de las endoscopias, permitiendo intervenciones más precisas y efectivas.

En esencia, aunque no se busca generar cambios inmediatos en la práctica clínica, se espera que esta investigación contribuya significativamente al cuerpo de conocimiento existente y catalice nuevas oportunidades para mejorar la detección de pólipos y, por extensión, la prevención del cáncer colorrectal.

## 1.2. Introducción al Proyecto

El presente proyecto se posiciona como un estudio fundamental en la intersección entre la visión por computadora y la detección de pólipos intestinales. Su núcleo radica en la validación y evaluación de un *dataset*, compuesto por imágenes de endoscopias con anotaciones precisas de pólipos, diseñado específicamente para su uso en modelos avanzados de segmentación de objetos.

El objetivo central de esta investigación es evaluar la eficacia de dos arquitecturas de vanguardia en visión por computadora: YOLOv8 [2] y Detectron2 [8]. Estos modelos, reconocidos por su capacidad para detectar objetos en imágenes de manera eficiente y precisa, se aplicarán al desafío específico de segmentar pólipos en imágenes de endoscopias. Este enfoque busca extender las capacidades de estas arquitecturas más allá de sus aplicaciones convencionales, explorando su potencial en un contexto médico altamente especializado.

Para lograr este objetivo, se ha desarrollado un flujo de trabajo integral que abarca:

1. La preparación meticulosa del *dataset*, incluyendo técnicas avanzadas de preprocesado y aumento de datos.
2. El ajuste del entrenamiento de los modelos YOLOv8 y Detectron2 utilizando las diferentes versiones del *dataset* original, realizando un estudio de los híperparametros en diferentes rangos.
3. La evaluación exhaustiva del rendimiento de los modelos mediante métricas relevantes en el contexto clínico, como la Intersección sobre Unión (IoU), la Mean Average Precision (mAP) y el Recall.

Este proyecto no solo se centra en la evaluación comparativa de dos potentes arquitecturas de segmentación de objetos, sino que también aspira a establecer un nuevo estándar en la utilización de *datasets* específicos para aplicaciones médicas. La validación del *dataset* propuesto como una herramienta confiable para la investigación y el desarrollo de tecnologías de apoyo al diagnóstico médico constituye un aspecto crucial de este estudio.

Los resultados obtenidos permitirán:

- Determinar la idoneidad del dataset para futuros estudios y aplicaciones clínicas.
- Evaluar la eficacia de YOLOv8 y Detectron2 en la detección de pólipos intestinales.
- Identificar áreas de mejora y optimización en la aplicación de técnicas de visión por computadora en contextos médicos.

## 1.3. Objetivos del proyecto

El objetivo principal de este proyecto es cimentar el uso del *dataset*, compuesto de imágenes segmentadas de pólipos intestinales, como posible material base para investigaciones futuras, en concreto para su uso en la detección automática mediante modelos de visión por computadora. Para conseguir este objetivo se utiliza este *dataset* para realizar una comparativa de dos métodos estado del arte, YOLOv8 y Detectron2, en una misma tarea de segmentación de pólipos intestinales.

### 1.3.1. Objetivos Específicos

1. **Adecuación de un *Dataset* Segmentado y Confiable**
   - Expandir a tres versiones del *dataset*:
     a) *Dataset* básico
     b) *Dataset* preprocesado
     c) *Dataset* preprocesado y *Data Augmentation*
   - Asegurar que el *dataset* sea adecuado para el entrenamiento y validación de modelos de detección de objetos.

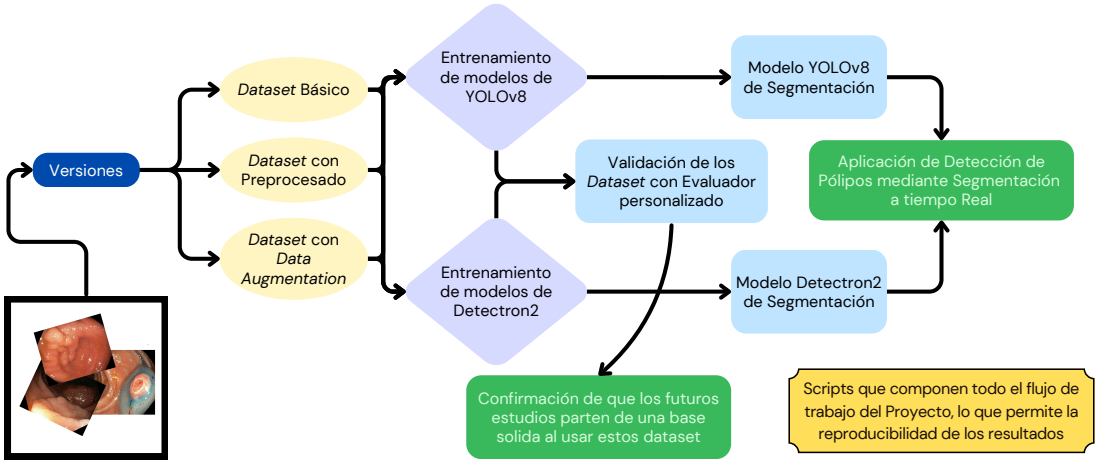


**Figura 1.1.:** Diagrama de flujo de los objetivos del Proyecto

- Implementar técnicas de segmentación para una detección más precisa que la simple utilización de *bounding boxes*.

2. **Entrenamiento y Evaluación de Modelos**
   - Entrenar dos modelos de visión por computadora:
     a) YOLOv8
     b) Detectron2
   - Utilizar las tres versiones del *dataset* para el entrenamiento.
   - Implementar un evaluador personalizado para validar la eficacia y eficicencia de los modelos entrenados.
3. **Optimización de Métricas de Evaluación**
   - Alcanzar un desempeño superior al 80 % en métricas clave:
     a) *Intersección sobre Unión* (IoU) [9]
     b) *Mean Average Precision* (mAP) [10]
     c) *Recall* [10]

- Evaluar métricas adicionales como *F1-Score* [11] y el tiempo de inferencia para complementar la validación de los resultados.

4. **Desarrollo de Modelos Eficientes para Detección en Tiempo Real**
   - Crear una aplicación de detección de pólipos mediante segmentación en tiempo real.
   - Asegurar que los modelos sean lo suficientemente ligeros y rápidos para su uso en procedimientos médicos en vivo.
5. **Validación y Reproducibilidad**
   - Confirmar que los *datasets* con los que se ha trabajado proporcionan una base sólida para futuros estudios.
   - Desarrollar *scripts* que compongan todo el flujo de trabajo del proyecto, garantizando la reproducibilidad de los resultados.
6. **Exploración de Técnicas de Mejora**
   - Investigar y aplicar diferentes opciones de *data augmentation* [12] y preprocesamiento de imágenes.
   - Optimizar los hiperparámetros de entrenamiento para maximizar el rendimiento de los modelos.

El objetivo final del proyecto es el desarrollo de soluciones prácticas aplicables en el ámbito médico. Los objetivos están diseñados para confirmar tanto el *dataset* como los modelos resultantes sean de alta calidad, ofreciendo una base sólida para futuras investigaciones y aplicaciones en la detección automática de pólipos intestinales.

## 1.4. Estructura del documento

El documento se ha ordenado de forma sencilla para presentar sistemáticamente las diferentes etapas del proyecto de investigación. El enfoque adoptado permite una progresión lógica desde los fundamentos conceptuales hasta los resultados empíricos y las conclusiones derivadas. El documento se organiza de la siguiente manera:

1. **Introducción:** Esta sección proporciona el contexto general del estudio. Se expone la problemática abordada, se justifica la relevancia de la investigación y se delinean los objetivos específicos que guían el trabajo.

2. **Estado del Arte:** Se presenta una revisión crítica de la literatura existente sobre métodos y tecnologías de visión por computadora aplicados a la detección de pólipos. Este apartado sitúa la investigación actual en el marco de los avances previos en el campo.

3. **Marco Teórico:** Se desarrollan los fundamentos conceptuales que sustentan el estudio. Se profundiza en los principios de visión por computadora y se detallan las arquitecturas de redes neuronales empleadas (YOLOv8 y Detectron2). Esta sección proporciona la base técnica necesaria para comprender las metodologías aplicadas.

4. **Metodología:** Se describe el procedimiento seguido para la creación del *dataset*, el preprocesamiento de las imágenes y el entrenamiento de los modelos. Se especifican los parámetros utilizados y las técnicas de optimización empleadas, garantizando la reproducibilidad del estudio.

5. **Resultados y Discusión:** Se presentan los hallazgos obtenidos a través de la aplicación de los modelos de detección de pólipos. Se analizan las métricas de rendimiento, como precisión, recall e IoU. Se discuten las implicaciones de los resultados, se identifican las limitaciones del estudio y se proponen posibles áreas de mejora.

6. **Conclusiones y Líneas Futuras:** Se sintetizan las principales contribuciones del estudio y se sugieren direcciones para futuras investigaciones en el campo.

CAPÍTULO 2

# Estado del Arte

El diagnóstico temprano y preciso de los pólipos intestinales es esencial para la prevención del cáncer colorrectal, una de las principales causas de mortalidad a nivel mundial [6]. A lo largo de los años, diversas técnicas han sido desarrolladas y aplicadas en la detección de estos pólipos durante las endoscopias, permitiendo a los médicos identificar y remover estas formaciones potencialmente peligrosas. Sin embargo, a pesar de los avances en la tecnología endoscópica, las técnicas tradicionales presentan limitaciones significativas, como la posibilidad de que pólipos pequeños o de tipo plano sean pasados por alto.

Con la evolución de la inteligencia artificial y la visión por computadora [13], han surgido nuevas herramientas que prometen mejorar la precisión y la eficacia en la detección de pólipos [14]. Estas tecnologías no solo complementan las habilidades del gastroenterólogo, sino que también abren nuevas posibilidades para el diagnóstico en tiempo real y el análisis automatizado de imágenes médicas. En esta sección, se revisarán tanto los métodos tradicionales como los enfoques más recientes basados en visión por computadora, ofreciendo una visión completa del estado actual de la investigación en este campo.

El objetivo de esta revisión es proporcionar un marco de referencia que contextualice el trabajo realizado en este proyecto, destacando las fortalezas y debilidades de las técnicas existentes y justificando la necesidad de innovaciones en el ámbito de la detección de pólipos intestinales.

## 2.1. Endoscopia: Procedimiento, Desafíos y Avances Tecnológicos

La endoscopia es una intervención invasiva fundamental para la detección de pólipos intestinales y otras anomalías en el tracto digestivo. Este procedimiento consiste en la inserción de un endoscopio, un tubo largo y flexible equipado con una cámara, a través del tracto digestivo para examinar posibles anomalías. Durante la endoscopia, el paciente debe estar sedado para minimizar el malestar y la incomodidad. Aunque es una técnica eficaz, se recomienda realizarla con poca frecuencia debido a su naturaleza invasiva y los riesgos asociados [15].

A pesar de su eficacia, la endoscopia presenta varios desafíos y áreas de mejora. Uno de los principales problemas es la preparación intestinal previa al procedimiento, que puede ser incómoda y difícil de cumplir para muchos pacientes. La preparación adecuada es crucial para garantizar una visualización clara del colon, pero la adherencia a las instrucciones de preparación puede ser baja [15].

Además, la sedación utilizada durante la endoscopia conlleva riesgos, especialmente en pacientes de edad avanzada o con comorbilidades. La evaluación preoperatoria y la monitorización durante el procedimiento son esenciales para minimizar estos riesgos. Otra área de mejora es la detección de pólipos pequeños, que pueden pasar desapercibidos durante el examen. La implementación de tecnologías avanzadas, como la inteligencia artificial y la detección de objetos, podría mejorar la precisión y la detección de estas anomalías [15].

El uso de sistemas asistidos por inteligencia artificial ha demostrado aumentar significativamente el número de pólipos detectados con éxito durante las endoscopias. Estos sistemas pueden identificar pólipos que podrían ser pasados por alto por el ojo humano, reduciendo así la necesidad de procedimientos repetidos y mejorando los resultados para los pacientes [15].

En resumen, aunque la endoscopia es una herramienta valiosa para la detección de pólipos intestinales, existen varias áreas en las que se puede mejorar la precisión del diagnóstico mediante el uso de tecnologías avanzadas.

## 2.2. Segmentación

### 2.2.1. Diferencias entre Segmentación, Detección de Objetos e Instanciación

En el ámbito de la visión por computadora, es crucial entender las diferencias entre las técnicas de segmentación[16, 9], detección de objetos e instanciación, ya que cada una tiene aplicaciones y metodologías distintas.

**Segmentación Semántica**

La segmentación semántica es una técnica que asigna una etiqueta a cada píxel de una imagen, clasificando cada región de la imagen en una categoría específica. Esta técnica es especialmente útil en aplicaciones donde es necesario entender la estructura y composición de la imagen a nivel de píxeles, como en la detección de tejidos en imágenes médicas. La segmentación semántica no distingue entre diferentes instancias del mismo objeto; por ejemplo, todos los píxeles que pertenecen a "pólipos" serán etiquetados de la misma manera, sin importar cuántos pólipos haya en la imagen.

**Detección de Objetos**

La detección de objetos, por otro lado, se centra en identificar y localizar objetos dentro de una imagen mediante el uso de cuadros delimitadores (bounding boxes). Esta técnica no proporciona información a nivel de píxeles, sino que indica la presencia y ubicación de objetos específicos. Es útil en aplicaciones donde la localización de objetos es más importante que la segmentación detallada, como en sistemas de vigilancia o en la identificación de vehículos en imágenes de tráfico.

**Instanciación (Segmentación de Instancias)**

La segmentación de instancias combina aspectos de la segmentación semántica y la detección de objetos. No solo clasifica cada píxel de la imagen, sino que también distingue entre diferentes instancias del mismo objeto. Por ejemplo, en una imagen con múltiples pólipos, la segmentación de instancias etiquetará cada pólipo individualmente, permitiendo un análisis más detallado y preciso. Esta técnica es particularmente útil en aplicaciones médicas avanzadas donde es crucial identificar y analizar cada instancia de un objeto por separado.

### 2.2.2. Comparativa de diferentes modelos de Segmentación

En el contexto de nuestro proyecto, hemos optado por utilizar una técnica de segmentación estándar. A diferencia de la segmentación por instancias o la segmentación semántica, nuestra aproximación se centra en la identificación de una única categoría de interés: los pólipos. Esta decisión se ha tomado con el objetivo de simplificar el estudio y facilitar el desarrollo inicial del modelo. Al enfocarnos en una sola categoría, podemos concentrar nuestros esfuerzos en optimizar la precisión y eficiencia del modelo en la detección de pólipos, estableciendo una base sólida para futuras mejoras y expansiones del sistema.

En el ámbito de la segmentación de instancias, existen diversos modelos que destacan por sus capacidades y aplicaciones específicas. En esta sección, se describiran los modelos de estado del arte que se usan en las tareas anteriormente descritas, destacando sus fortalezas y debilidades, así como sus usos prácticos. Esta comparativa servirá para justificar la elección de los modelos YOLOv8 y Detectron2 en nuestro proyecto.

**BEiT3**

BEiT3 [17] es un modelo de segmentación que se enfoca en la precisión y la sensibilidad al detalle. Es especialmente útil en aplicaciones que requieren la detección y segmentación de detalles finos y estructuras complejas, como la monitorización ambiental mediante imágenes satelitales y aéreas. Aunque BEiT3 ofrece una precisión superior, su complejidad y requerimientos de recursos pueden ser un obstáculo para su implementación en dispositivos con capacidades limitadas.

**SAM (Segment Anything Model)**

SAM [18] es un modelo diseñado para la segmentación interactiva, permitiendo a los usuarios proporcionar indicaciones iniciales a través de cuadros delimitadores o descripciones textuales. Este modelo es altamente amigable para el usuario, permitiendo a personas no técnicas realizar tareas de segmentación complejas con mínimo esfuerzo. SAM es ideal para la edición interactiva de imágenes y la detección personalizada de objetos. Sin embargo, su dependencia de la interacción del usuario puede no ser adecuada para aplicaciones que requieren segmentación completamente automatizada.

**YOLOv8**

YOLOv8 [2] (You Only Look Once, versión 8) es un modelo de segmentación de instancias que se ha ganado una reputación por su velocidad y eficiencia. Diseñado para aplicaciones en tiempo real, YOLOv8 es capaz de procesar imágenes rápidamente, lo que

lo hace ideal para escenarios donde la latencia es crítica, como en vehículos autónomos y sistemas de vigilancia. Una de las principales ventajas de YOLOv8 es su capacidad para operar en dispositivos con recursos limitados, como teléfonos móviles y sistemas embebidos, sin sacrificar significativamente la precisión.

Sin embargo, esta velocidad y eficiencia vienen con algunas limitaciones. Aunque YOLOv8 es excelente para aplicaciones en tiempo real, puede no ser la mejor opción para tareas que requieren una precisión extremadamente alta y una segmentación detallada. En comparación con otros modelos más precisos, YOLOv8 puede presentar una menor sensibilidad a los detalles finos y a las estructuras complejas.

**Detectron2**

Detectron2 [8], desarrollado por Facebook AI Research (FAIR), es un modelo de segmentación de instancias que se destaca por su alta precisión y flexibilidad. Detectron2 es conocido por su capacidad para manejar tareas complejas de segmentación con un alto grado de exactitud, lo que lo hace ideal para aplicaciones en las que la precisión es primordial, como en la segmentación de imágenes médicas y en la restauración digital de obras de arte.

Una de las principales fortalezas de Detectron2 es su robustez y capacidad para mantener un alto rendimiento en diversos conjuntos de datos y condiciones desafiantes. Además, Detectron2 ofrece una gran flexibilidad en términos de personalización y ajuste fino, permitiendo a los investigadores adaptar el modelo a sus necesidades específicas.

No obstante, esta precisión y flexibilidad tienen un costo en términos de velocidad y requisitos computacionales. Detectron2 puede ser más lento y requerir más recursos computacionales en comparación con modelos como YOLOv8, lo que puede limitar su uso en aplicaciones en tiempo real o en dispositivos con recursos limitados.

**Justificación de la Elección de Modelos**

La elección de los modelos YOLOv8 y Detectron2 para nuestro proyecto se basa en una combinación de sus fortalezas complementarias. YOLOv8 se utilizará para tareas que requieren una rápida segmentación en tiempo real, aprovechando su eficiencia y capacidad para operar en dispositivos con recursos limitados. Esto es particularmente útil en aplicaciones donde la velocidad es crítica y una segmentación detallada no es tan crucial.

Por otro lado, Detectron2 se empleará en tareas que requieren una alta precisión y una segmentación detallada, como en la detección y análisis de pólipos en imágenes médicas. La robustez y flexibilidad de Detectron2 permiten un análisis más exhaustivo y preciso, lo que es esencial para aplicaciones médicas avanzadas.

Al combinar estos dos modelos, podemos aprovechar lo mejor de ambos mundos: la velocidad y eficiencia de YOLOv8 para aplicaciones en tiempo real, y la precisión y flexibilidad de Detectron2 para tareas que requieren un análisis detallado y exacto. Esta estrategia nos permite abordar una amplia gama de desafíos en la segmentación de imágenes, optimizando tanto la eficiencia como la precisión de nuestro sistema.

**Tabla 2.1.:** Estudio comparativo de diferentes Modelos de Segmentación [5]

| Model | mAP box [COCO] | mAP mask [COCO] | License | Stars | Forks | Issues active/close |
|---|---|---|---|---|---|---|
| YOLOv9-seg | 53.3 | 43.5 | GPL-3.0 | 8.4k | 469 | 240/181 |
| YOLOv8-seg (Mejor) | 36.7 - 53.4 | 20.5 - 43.4 | AGPL-3.0 | 24.8k | 4.9k | 6656/701 |
| YOLOv7-seg | 51.4 | 41.5 | GPL-3.0 | 12.9k | 4.1k | 1413/381 |
| YOLACT++ | 36.1 | 34.1 | MIT | 5k | 1.3k | 401/389 |
| RTMDet-Ins | 40.5 - 52.4 | 35.4 - 44.6 | Apache-2.0 | 28.2k | 9.2k | 1465/6673 |
| SparseInst | 33.2 - 38.1 | 34.7 - 37.9 | MIT | 566 | 71 | 51/67 |
| Detectron2 | 36.8 - 44.3 | 32.1 - 39.5 | Apache-2.0 | 29.1k | 7.3k | 409/3055 |

### 2.2.3. Consideraciones y Limitaciones Actuales

Las limitaciones actuales en el campo de la visión computacional se centran, en gran medida, en la falta de un estándar unificado para la anotación de conjuntos de datos *datasets* [16]. Aunque disponemos de una cantidad masiva de imágenes, los modelos de inteligencia artificial evolucionan constantemente, lo que implica la necesidad de anotaciones y etiquetas cada vez más complejas y específicas. Esto es particularmente relevante en los modelos semánticos, que requieren clasificaciones precisas y detalladas. Como consecuencia, se hace imprescindible que un equipo de profesionales se dedique a la correcta anotación y clasificación de imágenes, una tarea que ha sido problemática desde los inicios de esta tecnología. Para abordar estos desafíos, se han propuesto soluciones que combinan diversas tecnologías de clasificación, incluidas redes neuronales de distintos tipos.

Sin embargo, otro obstáculo importante es la falta de acceso a los recursos computacionales necesarios para entrenar estos modelos, lo que limita la capacidad de muchos

investigadores para realizar estudios avanzados. Cabe destacar que los modelos preentrenados ofrecen una solución parcial a este problema, ya que permiten a los investigadores desarrollar modelos de detección de objetos utilizando conjuntos de datos mucho más pequeños que los requeridos para un entrenamiento desde cero, donde los pesos se ajustan desde el inicio.

CAPÍTULO 3

# Marco Teórico

El marco teórico de este estudio sobre la detección de pólipos intestinales mediante visión por computadora abarca una amplia gama de conceptos y técnicas fundamentales en el campo de la inteligencia artificial y el procesamiento de imágenes médicas. Esta sección proporciona los cimientos teóricos necesarios para comprender los métodos avanzados utilizados en la detección automatizada de pólipos, abarcando desde los principios básicos de la visión por computadora hasta las consideraciones éticas y regulatorias en la aplicación de la IA en la medicina.

La estructura de este marco teórico refleja la progresión lógica del procesamiento de imágenes médicas, comenzando con los fundamentos de la visión por computadora, avanzando hacia técnicas de aprendizaje profundo, y culminando en la evaluación y optimización de modelos. Además, se abordan las importantes consideraciones éticas y regulatorias que son cruciales en cualquier aplicación de IA en el ámbito médico.

## 3.1. Fundamentos de la Visión por Computadora

La visión por computadora es un campo interdisciplinario que busca dotar a las máquinas de la capacidad de interpretar y comprender información visual del mundo real, emulando y en ocasiones superando las capacidades del sistema visual humano [19]. En el contexto de la detección de pólipos intestinales, la visión por computadora proporciona las herramientas fundamentales para el análisis automático de imágenes de endoscopia, permitiendo una detección más precisa y eficiente de anomalías potencialmente cancerosas [20].

En esta sección se explicará, a nivel de operación más bajo, qué información se obtiene de las imágenes en visión por computadora, qué clase de operaciones se realizan sobre dicha información, y con qué técnicas se obtienen los resultados. Esta estructura facilitará la comprensión de cómo se transforman las imágenes crudas en datos útiles para la detección de pólipos.

### 3.1.1. Definición y conceptos básicos

La visión por computadora se define como la ciencia y tecnología de las máquinas que ven, donde "ver" métricas en este contexto implica que la máquina es capaz de extraer información de una imagen que es necesaria para resolver alguna tarea [19]. Los conceptos básicos incluyen:

**Imagen digital**: Representación matricial de intensidades de luz, donde cada elemento (píxel) contiene información sobre el brillo o color en un punto específico [21].

**Espacio de color**: Sistema de organización de colores, como RGB (Red, Green, Blue) o HSV (Hue, Saturation, Value), crucial para la representación y procesamiento de imágenes [22].

**Resolución**: Medida de la cantidad de detalle que una imagen puede capturar, expresada en píxeles o puntos por pulgada (DPI) [21].

**Ruido**: Variación aleatoria de brillo o color en las imágenes que puede afectar la calidad y el procesamiento [23].

### 3.1.2. Procesamiento de imágenes digitales

El procesamiento de imágenes digitales involucra la manipulación de imágenes para mejorar su calidad o extraer información útil. Las técnicas fundamentales incluyen:

**Filtrado**: Operaciones que modifican o realzan una imagen, como la reducción de ruido o la detección de bordes. Ejemplos comunes son los filtros gaussianos para suavizado y el operador Sobel para detección de bordes [21]. **Transformaciones morfológicas**: Operaciones basadas en la forma de las características de la imagen, como la dilatación y la erosión, útiles para la limpieza y segmentación de imágenes [24]. **Ecualización del histograma**: Técnica para ajustar el contraste de una imagen, redistribuyendo los valores de intensidad de los píxeles [21]. **Transformaciones geométricas**: Operaciones que

alteran la posición de los píxeles en la imagen, como rotaciones, escalados y traslaciones [25].

### 3.1.3. Técnicas de extracción de características

La extracción de características es el proceso de identificar y cuantificar aspectos relevantes de una imagen para su posterior análisis. Algunas técnicas comunes son:

**Descriptores de bordes**: Métodos como el detector de Canny que identifican cambios abruptos en la intensidad de la imagen, útiles para delinear estructuras [26].

**Descriptores de textura**: Técnicas como las matrices de co-ocurrencia de niveles de gris (GLCM) que cuantifican patrones repetitivos en la imagen [27].

**Descriptores de forma**: Métodos que capturan la geometría de los objetos en la imagen, como los momentos de Hu o los descriptores de Fourier [28].

**Puntos de interés**: Técnicas como SIFT (Scale-Invariant Feature Transform) o SURF (Speeded Up Robust Features) que identifican puntos distintivos en la imagen, invariantes a transformaciones como escala y rotación [29].

**Descriptores basados en aprendizaje**: Métodos que utilizan redes neuronales para aprender automáticamente características relevantes de las imágenes, como las características extraídas por las capas convolucionales en una CNN [30].

Estas técnicas de extracción de características son fundamentales en la detección de pólipos, ya que permiten identificar y cuantificar las propiedades visuales que distinguen los pólipos del tejido intestinal normal.

## 3.2. Segmentación de Imágenes

La segmentación de imágenes es un proceso fundamental en visión por computador que consiste en dividir una imagen en múltiples segmentos o regiones, cada uno correspondiente a un objeto o parte de un objeto en la escena. Este proceso es crucial para la comprensión y análisis automatizado de imágenes, sirviendo como base para tareas más complejas como el reconocimiento de objetos, el seguimiento de movimiento y la comprensión de escenas.

Las técnicas de segmentación de imágenes se pueden clasificar principalmente en dos categorías: basadas en discontinuidades (edge-based) y basadas en similitudes (region-based).

Las técnicas basadas en discontinuidades se enfocan en detectar bordes en la imagen, que son cambios abruptos en la intensidad de los píxeles. Entre los métodos más destacados están los operadores de primer orden, como Sobel y Canny, que usan derivadas para identificar bordes, y los de segundo orden, como el Laplaciano, que detecta cambios más sutiles en la imagen.

Por otro lado, las técnicas basadas en similitudes agrupan píxeles en regiones homogéneas según criterios como intensidad o color. Ejemplos clave incluyen el crecimiento de regiones, que expande áreas desde "semillasïniciales, y la división y fusión de regiones, que segmenta la imagen en cuadrantes y luego combina las áreas similares.

Sin embargo, con el avance de la inteligencia artificial y el aprendizaje profundo, han surgido nuevos enfoques que combinan estas técnicas tradicionales con métodos basados en redes neuronales.

### 3.2.1. Técnicas Avanzadas de Segmentación

Con el advenimiento del aprendizaje profundo, han surgido técnicas más sofisticadas que combinan aspectos de los enfoques basados en bordes y en regiones, superando muchas de las limitaciones de los métodos tradicionales.

#### 3.2.1.1. Redes Neuronales Convolucionales (CNN)

Las CNNs [31] han revolucionado el campo de la visión por computador, incluyendo la segmentación de imágenes. Estas redes son capaces de aprender automáticamente características relevantes de las imágenes en diferentes niveles de abstracción.

- **Arquitectura:** Consisten en capas convolucionales, de agrupación y completamente conectadas.
- **Aprendizaje de Características:** Las primeras capas aprenden características de bajo nivel como bordes y texturas, mientras que las capas más profundas capturan características de alto nivel como formas y patrones complejos.

- **Transferencia de Aprendizaje:** Las CNNs pre-entrenadas en grandes conjuntos de datos pueden adaptarse eficazmente a tareas específicas de segmentación con menor cantidad de datos de entrenamiento.
- **Mapas de Características:** En las etapas intermedias de las CNNs, se generan mapas de características que representan la activación de diferentes filtros convolucionales. Estos mapas son fundamentales para comprender cómo la red percibe y procesa la información visual.
  - **Posición:** Los mapas de características se encuentran típicamente después de cada capa convolucional y antes de las capas de agrupación.
  - **Función:** Cada mapa de características resalta diferentes aspectos de la imagen de entrada, como bordes, texturas o patrones específicos.
  - **Interpretación:** El análisis de estos mapas puede proporcionar insights sobre qué características son más relevantes para la tarea de segmentación.
- **Extracción de Características y Explicabilidad:** La capacidad de visualizar y comprender las características extraídas por las CNNs es crucial para la explicabilidad de los modelos.
  - **YOLO:** En el contexto de YOLO, la extracción de características se realiza a través de una red troncal *Backbone* que genera mapas de características a diferentes escalas. Estos mapas son luego utilizados por las capas de detección para localizar y clasificar objetos.
  - **Visualización:** Técnicas como la de DeepDream [32], desarrollada por Google, permiten visualizar las características aprendidas por la red. En el caso de YOLO, esto podría aplicarse para entender qué patrones activan fuertemente las diferentes partes de la red responsables de la detección de objetos específicos.
  - **Interpretación:** La visualización de los mapas de características en YOLO puede ayudar a comprender cómo el modelo identifica diferentes tipos de pólipos, basándose en sus formas, texturas y contextos espaciales.

#### 3.2.1.2. Segmentación Semántica

La segmentación semántica [33] busca asignar una etiqueta de clase a cada píxel de la imagen, proporcionando una comprensión detallada de la escena.

- **Arquitecturas Encoder-Decoder:** Redes como U-Net y SegNet utilizan una estructura de codificador-decodificador para capturar información contextual y producir mapas de segmentación de alta resolución.
- **Dilated Convolutions:** Permiten aumentar el campo receptivo sin aumentar el número de parámetros, capturando contexto a múltiples escalas.
- **Atención Espacial:** Mecanismos de atención que permiten a la red focalizarse en regiones relevantes de la imagen para mejorar la precisión de la segmentación.

#### 3.2.1.3. Segmentación de Instancias

La segmentación de instancias va un paso más allá de la segmentación semántica, identificando y segmentando instancias individuales de objetos, incluso si son de la misma clase.

- **Mask R-CNN:** Extiende Faster R-CNN añadiendo una rama para predecir máscaras de segmentación para cada instancia detectada.
- **YOLO Series:** Evolución de los modelos YOLO para incluir capacidades de segmentación de instancias, como en YOLOv8.
- **Enfoques Bottom-Up:** Métodos como DeepMask y SharpMask que generan propuestas de segmentación y luego las clasifican.

#### 3.2.1.4. Técnicas de Visualización y Explicabilidad

La comprensión de cómo las redes neuronales toman decisiones es crucial para su aplicación en contextos médicos sensibles como la segmentación de pólipos.

- **DeepDream:** Esta técnica, originalmente desarrollada por Google, se puede adaptar para visualizar las características que las redes de segmentación consideran importantes.
  - **Funcionamiento:** DeepDream amplifica las activaciones en ciertas capas de la red, permitiendo visualizar los patrones que la red ha aprendido a reconocer.
  - **Aplicación en Segmentación:** Al aplicar DeepDream a modelos como YOLO adaptados para la segmentación de pólipos, se pueden generar imágenes que

resalten las características que el modelo considera más relevantes para la identificación y delimitación de pólipos.

- **Mapas de Atención:** Técnicas que visualizan las regiones de la imagen en las que el modelo se está enfocando al realizar la segmentación.
  - **Grad-CAM:** Genera mapas de calor que indican las regiones más influyentes en la decisión del modelo.
  - **Interpretación:** En el contexto de la segmentación de pólipos, estos mapas pueden ayudar a los médicos a entender por qué el modelo ha clasificado ciertas áreas como pólipos.

Estas técnicas de visualización y explicabilidad no solo mejoran la comprensión de los modelos de segmentación, sino que también aumentan la confianza en su uso en aplicaciones médicas críticas como la detección y segmentación de pólipos.

### 3.2.2. Arquitectura YOLOv8

YOLOv8, la última iteración de la serie de modelos YOLO (You Only Look Once), representa un avance significativo en el campo de la detección de objetos en tiempo real. Desarrollado por Ultralytics, este modelo mantiene los principios fundamentales de sus predecesores mientras incorpora mejoras sustanciales en su arquitectura y rendimiento.

#### 3.2.2.1. Visión General de la Arquitectura

La arquitectura de YOLOv8 se compone de varios elementos clave que trabajan en conjunto para lograr una detección de objetos eficiente y precisa. La Figura 3.1 muestra una representación esquemática de esta arquitectura [34].

#### 3.2.2.2. Componentes Principales

##### Backbone

El backbone de YOLOv8 es la columna vertebral de la red neuronal, responsable de la extracción de características a múltiples escalas. Esta estructura jerárquica, representada en la Figura 3.1 como una serie de bloques etiquetados de C1 a C5, permite al modelo

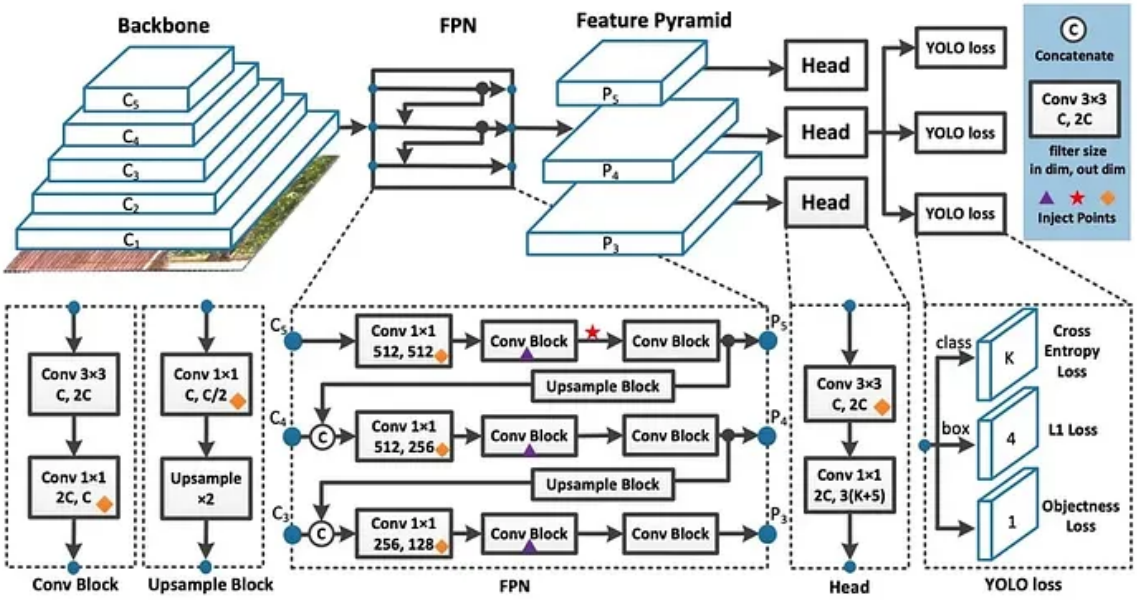


**Figura 3.1.:** Esquema detallado de la arquitectura YOLOv8 [1]

capturar información desde características de bajo nivel hasta patrones más complejos y abstractos.

El backbone implementa una serie de capas convolucionales y bloques residuales optimizados para maximizar la eficiencia computacional sin comprometer la calidad de las características extraídas. Esta optimización es crucial para mantener la capacidad de YOLOv8 de operar en tiempo real.

**Feature Pyramid Network (FPN)**

Tras el backbone, se encuentra la Feature Pyramid Network (FPN), un componente crucial para la detección de objetos a múltiples escalas. La FPN, visible en la sección central de la Figura 3.1, permite al modelo combinar características de diferentes niveles de abstracción.

Este proceso implica una serie de operaciones de upsample y concatenación, que fusionan información de capas profundas (con receptive fields más grandes) con capas más superficiales (que preservan detalles espaciales). El resultado es una representación rica en información que facilita la detección precisa de objetos de diversos tamaños.

**Anchor Boxes y Detection Heads**

Las anchor boxes, aunque no explícitamente representadas en la Figura 3.1, juegan un papel fundamental en el proceso de detección. Estas cajas predefinidas actúan como plantillas para la predicción de la localización y tamaño de los objetos.

Las detection heads, mostradas en la parte derecha de la figura como "Head", utilizan la información de las anchor boxes junto con las características extraídas para generar predicciones finales. Cada head se especializa en detectar objetos a una escala específica, aprovechando las diferentes resoluciones proporcionadas por la FPN.

#### 3.2.2.3. Proceso de Detección

El proceso de detección en YOLOv8 sigue un flujo secuencial [35]:

1. La imagen de entrada pasa a través del backbone para la extracción de características.
2. La FPN procesa y combina estas características a múltiples escalas.
3. Las detection heads, en conjunto con las anchor boxes, generan predicciones para cada celda de la cuadrícula de salida.
4. Cada predicción incluye:
    - Coordenadas de la bounding box
    - Score de confianza (objectness)
    - Probabilidades de clase
5. Se aplica un proceso de supresión no máxima (NMS) para filtrar predicciones redundantes.

#### 3.2.2.4. Funciones de Pérdida

YOLOv8 emplea múltiples funciones de pérdida para optimizar diferentes aspectos del proceso de detección:

- **YOLO Loss**: Combina errores de localización, confianza y clasificación.
- **Cross Entropy Loss**: Para la clasificación de objetos.
- **IoU Loss**: Mejora la precisión de las bounding boxes.

- **Objectness Loss**: Optimiza la detección de la presencia de objetos.

Estas funciones de pérdida, representadas en la parte inferior derecha de la Figura 3.1, trabajan en conjunto para refinar las predicciones del modelo durante el entrenamiento.

#### 3.2.2.5. Innovaciones y Mejoras

YOLOv8 introduce varias mejoras respecto a sus predecesores:

- **Arquitectura Optimizada**: Backbone y FPN mejorados para un mejor equilibrio entre velocidad y precisión.
- **Técnicas Avanzadas de *Data Augmentation***: Mejora la generalización y robustez del modelo.
- **Estrategias de Entrenamiento Refinadas**: Incluye técnicas como Mosaic augmentation y adaptative image scaling.
- **Inferencia Optimizada**: Mejoras en la eficiencia computacional para aplicaciones en tiempo real.

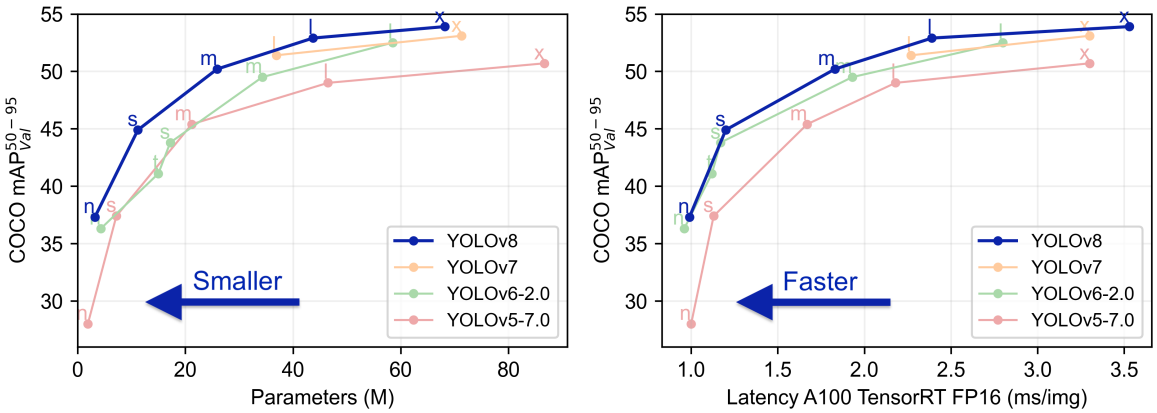


**Figura 3.2.:** Avances en rendimiento y precisión de YOLO [2] a lo largo de sus versiones

Como puede verse en 3.2, el algoritmo You Only Look Once (YOLO) ha recorrido un largo camino desde su creación, y YOLOv8 se erige ahora como un testimonio de su continua evolución.

#### 3.2.2.6. Proceso de Segmentación en YOLOv8

El proceso de segmentación en YOLOv8 se puede desglosar en los siguientes pasos:

1. **Extracción de Características:** La red utiliza su backbone para extraer características jerárquicas de la imagen de entrada.
2. **Detección de Objetos:** Se predicen bounding boxes y clases para los objetos detectados utilizando anclas predefinidas y regresión.
3. **Generación de Máscaras:** Para cada objeto detectado, se genera una máscara de segmentación utilizando una rama adicional en la arquitectura. Esta rama consiste en capas convolucionales transversales que producen un mapa de características de alta resolución.
4. **Refinamiento:** Las máscaras se refinan utilizando técnicas de upsampling y convoluciones para mejorar la precisión de los bordes. Se aplica un umbral para obtener la máscara binaria final.
5. **Post-procesamiento:** Se aplican técnicas como Non-Maximum Suppression (NMS) para eliminar detecciones redundantes y mejorar la precisión general.

#### 3.2.2.7. Aplicaciones y Versatilidad

La arquitectura de YOLOv8 lo hace adecuado para una amplia gama de aplicaciones, desde sistemas de vigilancia y análisis de imágenes médicas hasta vehículos autónomos y robótica. Su capacidad para manejar diferentes tamaños de entrada y sus variantes especializadas permiten su adaptación a diversos escenarios del mundo real, manteniendo un equilibrio entre velocidad y precisión.

En conclusión, YOLOv8 representa un avance significativo en el campo de la detección de objetos, combinando una arquitectura sofisticada con técnicas de optimización avanzadas para lograr un rendimiento excepcional en una variedad de aplicaciones prácticas.

### 3.2.3. Arquitectura Detectron2

Detectron2, desarrollado por Facebook AI Research (FAIR), representa un avance significativo en el campo de la visión por computadora, ofreciendo un framework modular

y escalable para tareas como detección de objetos y segmentación de instancias. Esta sección se enfoca en la arquitectura Faster R-CNN con Feature Pyramid Network (FPN), un componente fundamental de Detectron2.

#### 3.2.3.1. Visión General de la Arquitectura Faster R-CNN FPN

La arquitectura Faster R-CNN FPN en Detectron2 se compone de tres bloques principales [3], como se ilustra en la Figura 3.3:

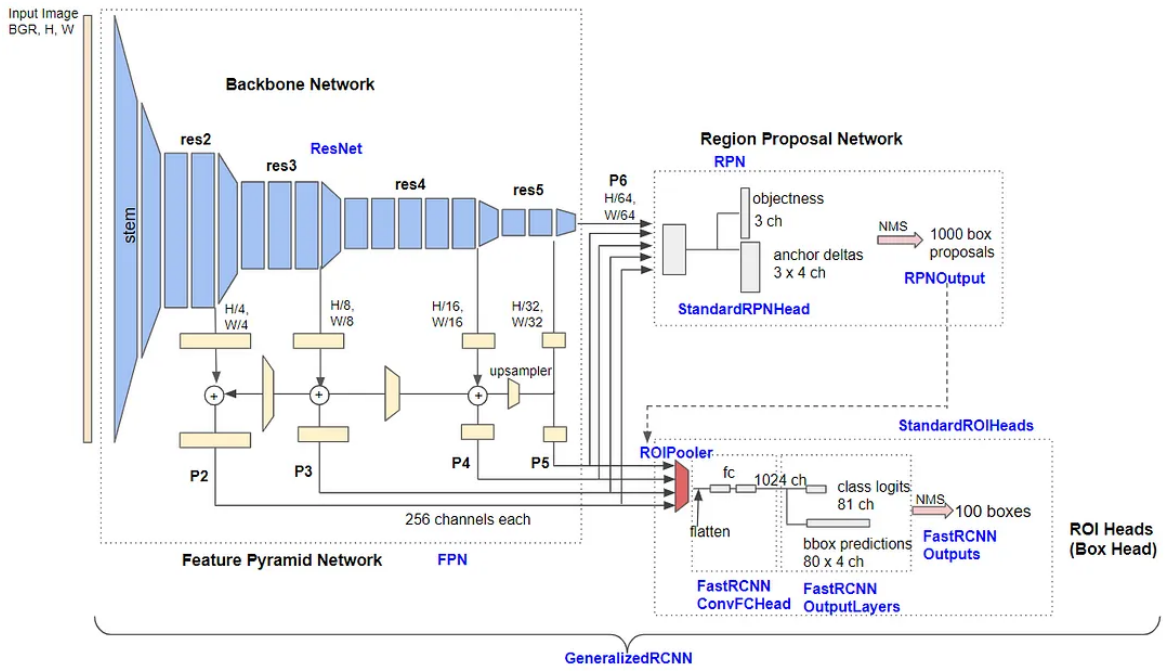


**Figura 3.3.:** Arquitectura detallada de Faster R-CNN FPN en Detectron2 [3]

**Backbone Network**

El backbone network, basado en una arquitectura ResNet con FPN, es responsable de extraer mapas de características a múltiples escalas. Las salidas se denominan P2 (escala 1/4), P3 (1/8), P4 (1/16), P5 (1/32) y P6 (1/64), proporcionando una representación rica en información de la imagen de entrada.

**Region Proposal Network (RPN)**

La RPN opera sobre las características multi-escala para generar propuestas de regiones de objetos. Por defecto, produce 1000 propuestas de cajas con puntuaciones de confianza asociadas.

#### Box Head

Este componente refina las propuestas de la RPN. Recorta y transforma los mapas de características utilizando las cajas propuestas, aplicando luego capas fully-connected para obtener localizaciones de cajas refinadas y resultados de clasificación. Finalmente, se aplica non-maximum suppression (NMS) para filtrar las predicciones, resultando en un máximo de 100 cajas por defecto.

### 3.2.3.2. Flujo de Datos y Procesamiento

El flujo de datos a través de la arquitectura Faster R-CNN FPN en Detectron2 sigue un proceso secuencial:

1. La imagen de entrada pasa por el backbone network, generando mapas de características a múltiples escalas (P2-P6).
2. La RPN procesa estos mapas para generar propuestas de regiones de objetos.
3. Las propuestas se utilizan para recortar y transformar los mapas de características relevantes.
4. El Box Head refina estas propuestas, realizando clasificación y ajuste fino de las coordenadas de las cajas.
5. Se aplica NMS para eliminar detecciones redundantes.
6. El resultado final es un conjunto de detecciones de objetos con clasificaciones y coordenadas precisas.

### 3.2.3.3. Proceso de Segmentación en Detectron2

El proceso de segmentación en Detectron2 utilizando Mask R-CNN se puede describir en los siguientes pasos:

1. **Extracción de Características:** La imagen de entrada pasa a través del backbone y FPN para generar mapas de características multi-escala.
2. **Generación de Propuestas:** La RPN genera propuestas de regiones que potencialmente contienen objetos.

3. **ROI Align:** Las características correspondientes a cada propuesta de región se extraen y se alinean con precisión utilizando ROI Align, que preserva la información espacial mejor que su predecesor, ROI Pooling.
4. **Clasificación y Refinamiento de Bounding Boxes:** Una rama de la red clasifica las propuestas en categorías específicas y refina las coordenadas de las bounding boxes.
5. **Generación de Máscaras:** Para cada instancia detectada, una rama separada genera una máscara de segmentación píxel a píxel. Esta rama consiste en una serie de capas convolucionales seguidas de una capa de deconvolución para producir una máscara de alta resolución.
6. **Post-procesamiento:** Se aplican técnicas como Non-Maximum Suppression (NMS) para eliminar detecciones redundantes y se realiza un umbral en las máscaras para obtener la segmentación final.

#### 3.2.3.4. Ventajas y Aplicaciones

La arquitectura Faster R-CNN FPN en Detectron2 ofrece varias ventajas:

- **Detección Multi-escala**: Capacidad para detectar objetos de diversos tamaños con alta precisión.
- **Modularidad**: Facilita la experimentación y la implementación de nuevas ideas en investigación.
- **Rendimiento**: Equilibrio entre velocidad y precisión, adecuado para aplicaciones en tiempo real.
- **Flexibilidad**: Adaptable a diversas tareas de visión por computadora más allá de la detección de objetos.

Estas características hacen que Detectron2 sea particularmente útil en aplicaciones como análisis de imágenes médicas, sistemas de asistencia a la conducción, y análisis de video para deportes y moda.

En conclusión, la arquitectura Faster R-CNN FPN implementada en Detectron2 representa un avance significativo en el campo de la visión por computadora, ofreciendo

un framework robusto y flexible para una amplia gama de aplicaciones de detección y segmentación de objetos.

### 3.2.4. Comparación entre YOLOv8 y Detectron2

Aunque tanto YOLOv8 como Detectron2 (específicamente su implementación de Mask R-CNN [36]) son herramientas poderosas para la segmentación de instancias, difieren en varios aspectos importantes:

#### 3.2.4.1. Enfoque Arquitectónico

**YOLOv8:** Utiliza un enfoque de una sola etapa, realizando la detección y segmentación simultáneamente. Esto resulta en una arquitectura más compacta y generalmente más rápida.

**Detectron2 (Mask R-CNN):** Emplea un enfoque de dos etapas, primero generando propuestas de regiones y luego refinando estas propuestas para la clasificación y segmentación. Este enfoque suele ofrecer mayor precisión, especialmente en escenarios complejos.

#### 3.2.4.2. Velocidad vs Precisión

**YOLOv8:** Está diseñado para ser extremadamente rápido, lo que lo hace ideal para aplicaciones en tiempo real. Sin embargo, puede sacrificar algo de precisión en escenarios muy complejos.

**Detectron2:** Tiende a ser más preciso, especialmente en la detección de objetos pequeños o en escenas complejas. Sin embargo, esto viene a costa de una mayor complejidad computacional y tiempos de inferencia más largos.

#### 3.2.4.3. Flexibilidad y Personalización

**YOLOv8:** Ofrece una arquitectura más fija, optimizada para un rendimiento rápido out-of-the-box. La personalización suele limitarse a ajustes en el entrenamiento y la selección de backbones.

**Detectron2:** Proporciona un framework más flexible y modular, permitiendo a los investigadores y desarrolladores experimentar con diferentes componentes y arquitecturas. Esto lo hace más adaptable a una amplia gama de tareas y escenarios.

#### 3.2.4.4. Casos de Uso

**YOLOv8:** Es particularmente adecuado para aplicaciones que requieren procesamiento en tiempo real, como la videovigilancia, la conducción autónoma o la robótica.

**Detectron2:** Brilla en aplicaciones donde la precisión es crítica y el tiempo de procesamiento es menos restrictivo, como el análisis de imágenes médicas, la inspección industrial de alta precisión o la investigación en visión por computador.

#### 3.2.4.5. Ecosistema y Soporte

**YOLOv8:** Tiene una comunidad activa y está respaldado por Ultralytics, lo que garantiza actualizaciones frecuentes y mejoras continuas.

**Detectron2:** Cuenta con el respaldo de Facebook AI Research, lo que proporciona una base sólida para la investigación y el desarrollo. Su naturaleza modular lo hace popular en entornos académicos y de investigación.

## 3.3. Métricas de Evaluación en Segmentación de Imágenes

La evaluación cuantitativa de los algoritmos de segmentación de imágenes es crucial para comparar diferentes métodos y determinar su eficacia en tareas específicas, como la detección de pólipos intestinales. Las métricas de evaluación proporcionan una medida objetiva del rendimiento del algoritmo, comparando la segmentación producida con una segmentación de referencia o *Ground Truth* [37].

En la Figura 3.4 se muestran un ejemplo de ambas máscaras, la creada por los expertos en el campo oncológico y la predicción del modelo.

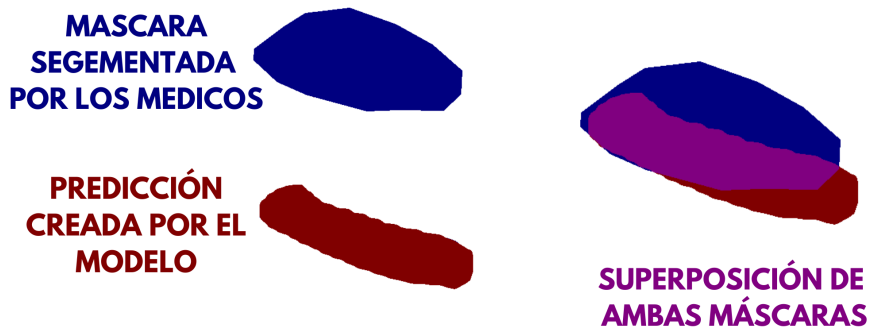


**Figura 3.4.:** Máscaras (Elaboración propia)

En la Figura 4.5 se visualiza la distribución de False Negatives, False Positives, True Negatives y True Positives, que se va ha usar para calcular las métricas explicadas a continuación.

### 3.3.1. Precisión

La **Precisión** es una métrica fundamental en la segmentación de imágenes, especialmente en la detección de objetos, donde se define como la proporción de verdaderos positivos (TP) frente al total de elementos clasificados como positivos, es decir, la suma de verdaderos positivos y falsos positivos (FP). Su fórmula es:

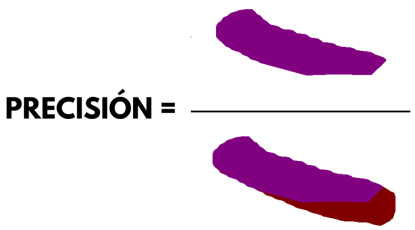


**Figura 3.5.:** Ilustración del cálculo del Recall (Elaboración propia)

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{3.1}$$

Una alta precisión indica que el modelo comete pocos errores al clasificar píxeles o regiones como parte del objeto de interés. Sin embargo, no considera los falsos negativos, lo que puede ser problemático en aplicaciones críticas donde la detección de todos los objetos es crucial.

### 3.3.2. Recall

El **Recall**, también conocido como sensibilidad, mide la capacidad del modelo para identificar correctamente todos los píxeles o regiones del objeto de interés. Se calcula como la proporción de verdaderos positivos sobre la suma de verdaderos positivos y falsos negativos (FN):

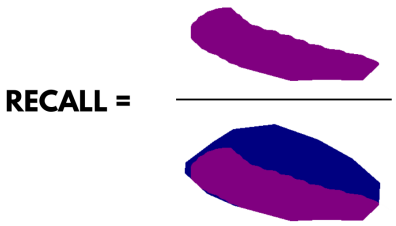


**Figura 3.6.:** Ilustración del cálculo de la Precisión (Elaboración propia)

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

El recall es esencial en contextos donde es crucial minimizar la omisión de objetos relevantes. A menudo se utiliza junto con la precisión para obtener una visión más equilibrada del rendimiento del modelo.

### 3.3.3. F1-Score

El **F1-Score** es la media armónica de la precisión y el recall, proporcionando un único valor que equilibra ambos aspectos. Es especialmente útil cuando hay un desbalance entre clases o cuando es necesario un compromiso entre precisión y recall:

$$\text{F1-Score} = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}} \tag{3.3}$$

Este valor puede interpretarse como una métrica de precisión global del sistema, considerando tanto la capacidad de detectar objetos como la exactitud en su identificación.

### 3.3.4. Índice de Jaccard (IoU)

El **Índice de Jaccard**, también conocido como **Intersection over Union (IoU)**, es una métrica clave en la segmentación de imágenes. Se define como la intersección entre la segmentación predicha y la segmentación de referencia (ground truth) dividida por su unión:

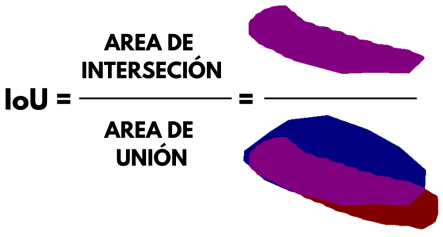


**Figura 3.7.:** Ilustración del cálculo de IoU (Elaboración propia)

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{3.4}$$

Donde $A$ representa la máscara de segmentación predicha y $B$ la máscara de segmentación de referencia. Un valor de IoU cercano a 1 indica una alta superposición y, por lo tanto, una segmentación precisa.

## 3.4. Optimización de Hiperparámetros

La optimización de hiperparámetros es un aspecto crucial en el desarrollo de modelos de aprendizaje automático y aprendizaje profundo efectivos. En el contexto de la segmentación de imágenes médicas para la detección de pólipos intestinales, la selección adecuada de hiperparámetros puede tener un impacto significativo en el rendimiento del modelo [38].

### 3.4.1. Conceptos de hiperparámetros vs. parámetros

**Parámetros**: Son los valores internos del modelo que se aprenden durante el entrenamiento a partir de los datos, como los pesos y sesgos en una red neuronal [39].

**Hiperparámetros**: Son configuraciones externas al modelo que no se aprenden de los datos, pero que influyen en el proceso de aprendizaje y la arquitectura del modelo. Ejemplos incluyen la tasa de aprendizaje, el número de capas en una red neuronal, o el número de árboles en un bosque aleatorio [40].

### 3.4.2. Técnicas de búsqueda de hiperparámetros (grid search, random search)

**Búsqueda en cuadrícula (Grid Search)**:

- Implica definir un conjunto discreto de valores para cada hiperparámetro y evaluar todas las combinaciones posibles.
- Ventaja: Exhaustivo y garantiza encontrar la mejor combinación dentro del espacio de búsqueda definido.
- Desventaja: Computacionalmente costoso, especialmente con muchos hiperparámetros [41].

**Búsqueda aleatoria (Random Search)**:

- Selecciona combinaciones de hiperparámetros al azar dentro de rangos definidos.
- Ventaja: Más eficiente que la búsqueda en cuadrícula, especialmente cuando no todos los hiperparámetros son igualmente importantes.
- Desventaja: No garantiza encontrar la combinación óptima, pero a menudo encuentra una buena solución con menos tiempo de cómputo [42].

**Búsqueda basada en población**:

- Utiliza algoritmos evolutivos o genéticos para optimizar hiperparámetros.
- Ventaja: Puede explorar espacios de búsqueda complejos y no convexos.
- Desventaja: Puede ser computacionalmente intensivo y requiere una cuidadosa configuración de los propios parámetros del algoritmo evolutivo [43].

### 3.4.3. Optimización bayesiana

La optimización bayesiana es una técnica avanzada para la búsqueda de hiperparámetros que ha ganado popularidad en los últimos años:

**Principio**: Utiliza un modelo probabilístico (generalmente un proceso gaussiano) para modelar la relación entre los hiperparámetros y el rendimiento del modelo [44].
**Proceso**:

- Construye un modelo sustituto de la función objetivo (rendimiento del modelo).
- Utiliza una función de adquisición para decidir qué combinación de hiperparámetros probar a continuación.
- Actualiza el modelo sustituto con los nuevos resultados [45].

**Ventajas**:

- Más eficiente que la búsqueda en cuadrícula o aleatoria, especialmente para funciones objetivo costosas de evaluar.
- Puede manejar espacios de búsqueda continuos y discretos [46].

**Implementaciones**: Bibliotecas como Hyperopt, Optuna, y scikit-optimize proporcionan implementaciones de optimización bayesiana [47].

CAPÍTULO 4

# Metodología

## 4.1. Dataset

En esta sección, describimos el proceso seguido para la preparación del dataset utilizado en el proyecto. Este proceso fue fundamental para asegurar la calidad y la relevancia de los datos que alimentaron los modelos de detección de pólipos intestinales.

### 4.1.1. Origen del *Dataset*

Las imágenes utilizadas en las diferentes versiones del *dataset* han sido proporcionadas por Cristián Castillo Olea, investigadora en la Universidad Autónoma de Baja California, México. Este conjunto de datos consiste en imágenes de endoscopias segmentadas, donde los pólipos intestinales han sido identificados y delimitados con precisión. Es importante destacar que las imágenes estaban ya segmentadas, lo que implica que no fue necesario realizar un etiquetado manual desde cero, pero sí fue esencial ajustar y preparar el *dataset* para su uso en la plataforma de entrenamiento seleccionada.

### 4.1.2. Creación del *Dataset* en Roboflow

Una vez recibido el conjunto de imágenes, utilizamos Roboflow, una plataforma especializada en la creación, manejo y preprocesamiento de *datasets* para visión por computadora. El uso de Roboflow nos permitió estructurar, preprocesar y aumentar las imágenes de

manera eficiente, generando así varias versiones del *dataset* adaptadas a las necesidades específicas de nuestro proyecto.

#### 4.1.2.1. Segmentación de las Imágenes

El primer paso en la preparación del *dataset* consistió en la segmentación de todas las imágenes. Inicialmente, se recibieron dos conjuntos de imágenes: uno que contenía las imágenes originales sin modificaciones y otro que incluía las mismas imágenes con una máscara superpuesta que identificaba los tejidos relevantes para el estudio.

Utilizando el conjunto que contenía las imágenes con las máscaras de los pólipos identificados, se empleó la herramienta de segmentación de Roboflow para crear un *dataset* segmentado, sin aplicar ningún preprocesamiento o modificación adicional. Posteriormente, fue necesario cargar en Roboflow las imágenes originales junto con las anotaciones generadas.

Durante este proceso, surgió un inconveniente debido a que los dos conjuntos de imágenes no presentaban el mismo tamaño, lo cual podría haber generado complicaciones dependiendo del formato del *dataset* empleado. Afortunadamente, dado que YOLO utiliza coordenadas relativas para la creación de sus máscaras, se optó por extraer el *dataset* en ese formato y, posteriormente, volver a cargarlo reemplazando las imágenes por las originales. De esta manera, se logró crear un *dataset* segmentado con las imágenes adecuadas para el estudio.

Para la creación del *dataset* inicial, se agregaron todas las imágenes al conjunto de *train* para facilitar el posterior intercambio. Una vez completado el proceso de reemplazo de imágenes, se realizó un rebalanceo de los datos, distribuyendo un 60 % de las imágenes en el conjunto de *training*, un 20 % en el conjunto de *testing* y un 20 % en el conjunto de *validating*.

#### 4.1.2.2. Creación de Diferentes Versiones del *Dataset*

A partir del *dataset* segmentado, se crearon tres versiones distintas del *dataset*, cada una con características específicas para probar la robustez de los modelos entrenados:

- ***Dataset* Original:** Este conjunto incluye las imágenes tal como fueron proporcionadas, sin ningún tipo de preprocesado o modificación. Las imágenes conservan su tamaño y aspecto de ratio original.

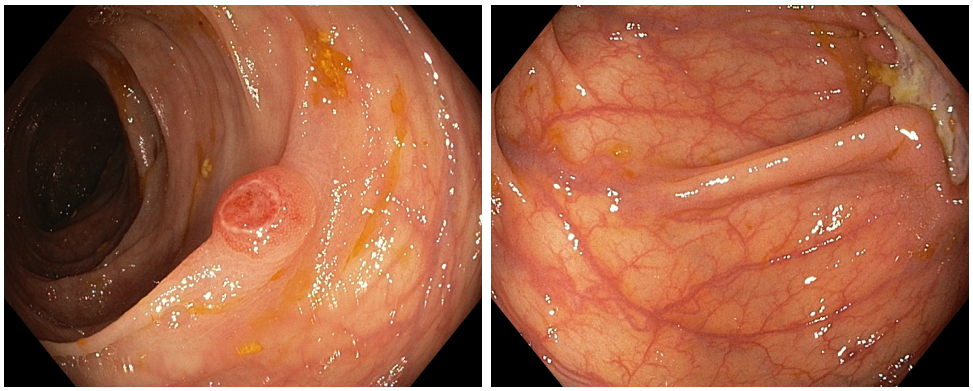

**Figura 4.1.:** Ejemplo de una imagen del *dataset* original.

- ***Dataset* Preprocesado:** En esta versión, las imágenes fueron convertidas a formato cuadrado mediante la adición de secciones negras en la parte superior e inferior, lo que permitió estandarizar el tamaño de las imágenes sin distorsionar el contenido relevante.

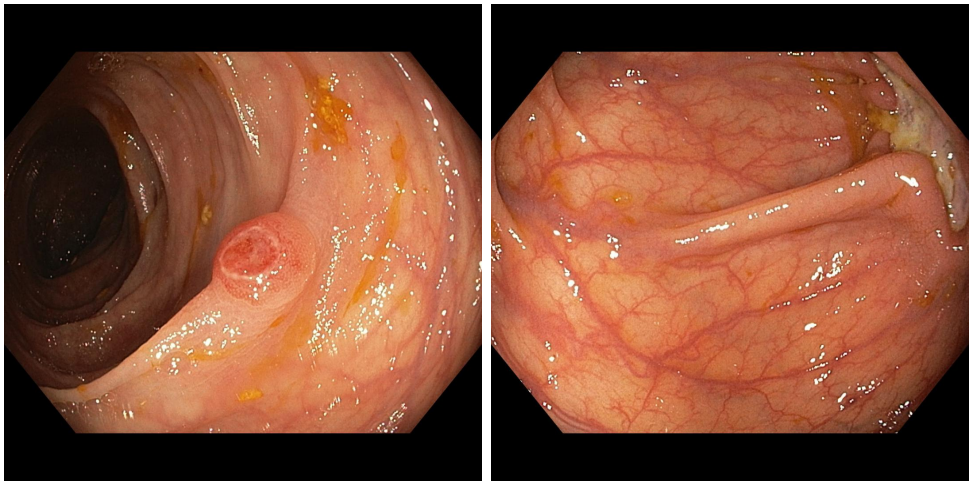

**Figura 4.2.:** Ejemplo de una imagen del *dataset* preprocesado.

- ***Dataset* con *Data Augmentation*:** Basado en el *dataset* preprocesado, se aplicaron técnicas de aumento de datos a las imágenes de la sección de entrenamiento. Estas técnicas incluyeron la rotación de las imágenes en un rango de 10º a 15º y un flip en el eje horizontal, lo que triplicó el número de imágenes disponibles para el entrenamiento.

Es importante señalar que, gracias a *Roboflow*, las imágenes del *dataset* se distribuyeron inicialmente entre entrenamiento, validación y test de acuerdo a un ratio predefinido.

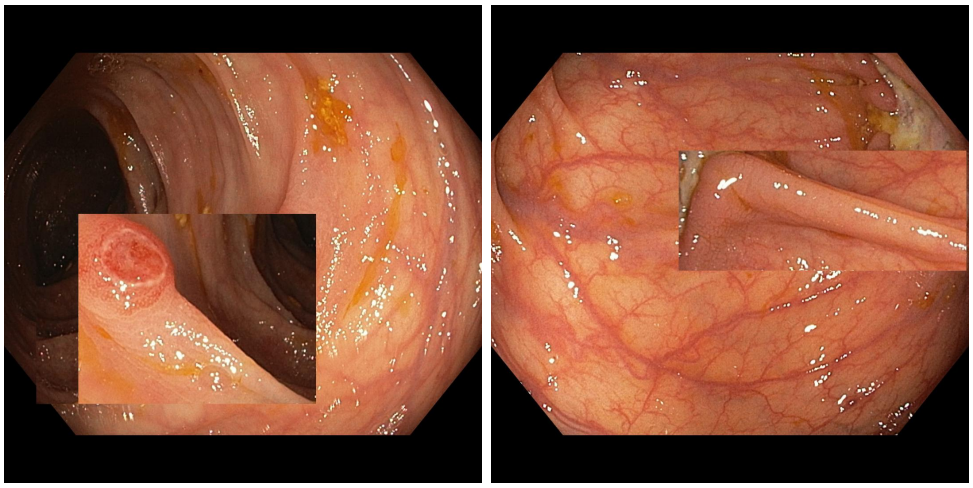

**Figura 4.3.:** Ejemplo de una imagen del *dataset* con *data augmentation*.

Posteriormente, cualquier modificación aplicada al *dataset* mantuvo esta misma estructura de distribución, lo que es crucial para la consistencia de nuestro estudio.

### 4.1.3. Categorización del *Dataset*

Para simplificar y enfocar el proceso de validación, todas las imágenes fueron categorizadas bajo una única etiqueta: pólipo. No se realizó una subcategorización adicional en función de los diferentes tipos de tejido o variaciones en los pólipos. Este enfoque nos permite centrarnos en la precisión general de la detección de pólipos, evitando la complejidad adicional que implicaría categorizar los diferentes tipos de cáncer o pólipos.

Es importante destacar que, dado que no estamos diferenciando entre varios tipos de pólipos o tejidos, este proyecto no puede ser considerado un caso de segmentación semántica. En lugar de ello, nos enfocamos en la creación de un detector de pólipos general, lo cual es esencial para los objetivos de este estudio.

Este enfoque en la preparación y manipulación del *dataset* fue clave para maximizar la eficacia del entrenamiento de los modelos de detección de objetos utilizados en este proyecto, asegurando que los datos fueran tanto representativos como variados, permitiendo así una validación rigurosa de las capacidades de los modelos.

## 4.2. Entrenamiento de Modelos

En esta sección, abordaremos el proceso de entrenamiento de los modelos de detección de pólipos intestinales utilizando dos frameworks avanzados: YOLOv8 y Detectron2. Esta parte del proyecto es crucial para evaluar la eficacia del *dataset* creado y optimizar los modelos para su aplicación en el ámbito médico.

Para el proceso de entrenamiento, se han empleado las funciones básicas proporcionadas por ambas arquitecturas, así como las funciones de validación. Estas funciones se describen con mayor detalle en las documentaciones oficiales de YOLOv8 [48] y Detectron2 [49].

### 4.2.1. Hiperparámetros del Entrenamiento

El entrenamiento de los modelos con YOLOv8 y Detectron2 requiere una configuración precisa de hiperparámetros, que afectan directamente el rendimiento y la eficacia del modelo.

**Tabla 4.1.:** Hiperparámetros de estudio en YOLOv8.

| **Hiperparámetros** | **Descripción** |
|---|---|
| **Número de épocas** (***epochs***) | Número de veces que el modelo pasa por todo el conjunto de datos de entrenamiento. |
| **Paciencia** (***patience***) | Número de épocas sin mejora en la métrica de validación antes de detener el entrenamiento. |
| **Tamaño del lote** (***batch***) | Número de imágenes procesadas simultáneamente en una pasada hacia adelante. |
| **Número de trabajadores** (***workers***) | Número de subprocesos utilizados para cargar los datos. |
| **Tamaño de la imagen** (***imgsz***) | Dimensiones de las imágenes de entrada. |
| **Periodo de guardado** (***save_period***) | Intervalo de épocas para guardar el modelo. |
| **Tasa de aprendizaje inicial** (***lr0***) | Tasa de aprendizaje inicial utilizada por el optimizador. |
| **Momento** (***momentum***) | Parámetro que ayuda a acelerar el descenso del gradiente en la dirección correcta. |
| **Decaimiento del peso** (***weight_decay***) | Parámetro de regularización que penaliza los pesos grandes para evitar el sobreajuste. |
| **Épocas de calentamiento** (***warmup_epochs***) | Número de épocas durante las cuales la tasa de aprendizaje aumenta gradualmente desde cero hasta el valor inicial. |
| **Momento de calentamiento** (***warmup_momentum***) | Valor del momento durante las épocas de calentamiento. |
| **Tasa de aprendizaje del sesgo de calentamiento** (***warmup_bias_lr***) | Tasa de aprendizaje específica para el sesgo durante las épocas de calentamiento. |
| **Tipo de optimizador** (***optimizer***) | Tipo de optimizador utilizado (por ejemplo, RMSProp, Adam, SGD). |
| **Umbral de prueba** (***conf_thres***) | Umbral de confianza para las predicciones durante la prueba. |

**Tabla 4.2.:** Hiperparámetros de estudio en *Detectron2*.

| Hiperparámetros | Descripción |
|---|---|
| **Número máximo de iteraciones** (***max_iter***) | Número máximo de iteraciones de entrenamiento. |
| **Paso de iteración máxima** (***max_iter_step***) | Incremento en el número de iteraciones para cada paso de ajuste. |
| **Paciencia** (***patience***) | Número de iteraciones sin mejora antes de detener el entrenamiento. |
| **Número de trabajadores** (***workers***) | Número de subprocesos utilizados para cargar los datos. |
| **Imágenes por lote** (***images_per_batch***) | Número de imágenes procesadas simultáneamente en una pasada hacia adelante. |
| **Tamaño mínimo y máximo de entrenamiento** (***min_max_size_train***) | Dimensiones mínimas y máximas de las imágenes de entrada durante el entrenamiento. |
| **Periodo de guardado** (***checkpoint_period***) | Intervalo de iteraciones para guardar el modelo. |
| **Tasa de aprendizaje base** (***base_lr***) | Tasa de aprendizaje inicial utilizada por el optimizador. |
| **Momento** (***momentum***) | Parámetro que ayuda a acelerar el descenso del gradiente en la dirección correcta. |
| **Gamma** (***gamma***) | Factor de reducción de la tasa de aprendizaje. |
| **Decaimiento del peso** (***weight_decay***) | Parámetro de regularización que penaliza los pesos grandes para evitar el sobreajuste. |
| **Iteraciones de calentamiento** (***warmup_iters***) | Número de iteraciones durante las cuales la tasa de aprendizaje aumenta gradualmente desde cero hasta el valor inicial. |
| **Momento de calentamiento** (***warmup_momentum***) | Valor del momento durante las iteraciones de calentamiento. |
| **Tipo de optimizador** (***optimizer***) | Tipo de optimizador utilizado (por ejemplo, Adagrad, Adam, SGD). |
| **Umbral de prueba** (***testing_threshold***) | Umbral de confianza para las predicciones durante la prueba. |

### 4.2.2. Optimización de Hiperparámetros mediante *Grid Search* Adaptativo

La optimización de hiperparámetros se llevó a cabo utilizando un enfoque de *Grid Search* [4]. Esta es una herramienta valiosa para identificar los hiperparámetros óptimos de un modelo de aprendizaje automático. En un modelo de este tipo, existen configuraciones ajustables, denominadas hiperparámetros, que pueden mejorar su rendimiento. *Grid Search* facilita la identificación automática de la mejor combinación de estos hiperparámetros.

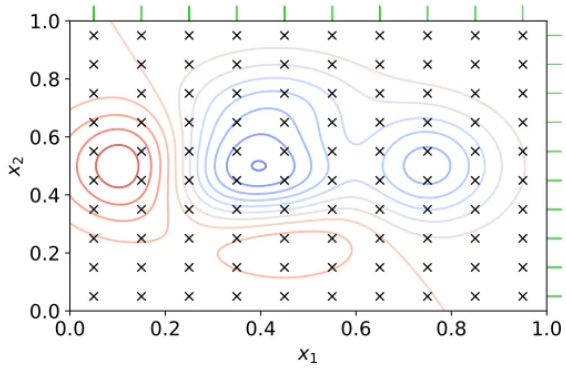


**Figura 4.4.:** Ejemplo visual de *Grid Search* con dos parametros [4]

Se plantean un conjunto de hiperparámetros de evaluación, ya descritos en el Apartado 4.2.1, y este explora de manera sistemática cada posible combinación. Para cada combinación, se evalúa el rendimiento del modelo probándolo en diferentes secciones del conjunto de datos, con el fin de medir su precisión.

Tras evaluar exhaustivamente todas las combinaciones, *Grid Search* ofrece la combinación de configuraciones que ha generado los resultados más favorables. Este enfoque simplifica el proceso de ajuste del modelo, garantizando un funcionamiento óptimo para la tarea específica, sin incurrir en costos computacionales excesivos.

#### Optimización en Modelos de Detección

En el proceso de ajuste de hiperparámetros mediante *Grid Search*, la métrica seleccionada para guiar la optimización fue el *mean Average Precision* ($mAP$) al 50 % de $IoU$ (*Intersection over Union*). Esta métrica mide la precisión media de las predicciones en función de un umbral de solapamiento entre las predicciones y las *ground truths*, y es ampliamente utilizada en la evaluación de modelos de detección de objetos. En cada

evaluación, el valor de *mAP@0.5* se devolvió como referencia para determinar la calidad de los hiperparámetros evaluados.

Este enfoque garantiza que el modelo optimice no solo la precisión general, sino que también se enfoque en un umbral de $IoU$ que sea relevante para la tarea de detección, permitiendo una evaluación más precisa del rendimiento del modelo.

Este método de optimización se ha aplicado tanto para los modelos basados en YOLOv8 como para Detectron2, asegurando que ambos sistemas se beneficien de la misma métrica de evaluación y proporcionando una comparación coherente entre ellos.

**Optuna: Comparativa con *Grid Search***

Optuna es una herramienta de optimización de hiperparámetros que, aunque parte de *Grid Search*, se diferencia en que no requiere una búsqueda exhaustiva en un espacio predefinido de hiperparámetros. Optuna implementa una búsqueda más eficiente y dirigida, utilizando técnicas de muestreo adaptativo y estrategias de estimación para explorar de manera más inteligente el espacio de búsqueda.

Mientras que *Grid Search* explora todas las posibles combinaciones de manera sistemática, Optuna ajusta su búsqueda en función de los resultados obtenidos durante el proceso de optimización, lo que permite reducir significativamente el tiempo de cómputo. Esta característica hace que Optuna sea particularmente útil en escenarios donde el espacio de hiperparámetros es amplio o cuando se dispone de recursos computacionales limitados.

## 4.3. Evaluación de Modelos

En esta sección, se detallará el proceso de evaluación de los modelos de detección de pólipos intestinales entrenados con YOLOv8 y Detectron2. La evaluación es esencial para medir la precisión y eficacia de los modelos y para garantizar que el *dataset* creado y utilizado cumple con los estándares requeridos. Evaluación de Modelos YOLOv8 y Detectron2

La evaluación de los modelos de detección de objetos es fundamental para validar su rendimiento. Sin embargo, es importante destacar que los frameworks YOLOv8 y Detectron2 ofrecen métricas de evaluación diferentes de manera nativa. Esta discrepancia

en las métricas devueltas por cada framework requiere una solución para asegurar una evaluación homogénea.

#### *YOLOv8*

El framework YOLOv8 proporciona varias métricas para evaluar el rendimiento del modelo en tareas de detección, tanto en la predicción de *Bounding Boxes* como en la segmentación de máscaras. En este estudio, el análisis se ha centrado en la parte de segmentación, utilizando las siguientes métricas clave:

- **Precisión**: Evalúa la proporción de predicciones correctas sobre el total de predicciones realizadas por el modelo. Una precisión alta indica que las detecciones positivas son mayormente correctas, lo cual es crucial para minimizar los falsos positivos en aplicaciones de detección de objetos.
- **Recall**: Mide la capacidad del modelo para identificar correctamente todos los objetos presentes en la imagen. Un valor alto de *recall* sugiere que el modelo tiene un buen rendimiento en la detección de verdaderos positivos, es decir, objetos reales que no son omitidos.
- **mAP50**: Media del promedio de precisión (Average Precision) calculada en un umbral de intersección sobre unión (IoU) del 50 %. Esta métrica mide tanto la precisión como el *recall*, evaluando el rendimiento general del modelo en diferentes clases de objetos.
- **Fitness**: Métrica compuesta que combina precisión, *recall* y mAP, utilizada para optimizar el rendimiento general del modelo. Sirve como una métrica integral que facilita el proceso de ajuste de hiperparámetros.

#### *Detectron2*

Detectron2 proporciona un conjunto de métricas centradas principalmente en la evaluación de la *Average Precision* (AP), que permite medir la precisión del modelo a diferentes umbrales de intersección sobre unión (IoU) y escalas de objetos. Las principales métricas utilizadas en este estudio incluyen:

- **AP (Average Precision)**: Valor promedio de precisión para diferentes umbrales de IoU. Es una métrica estándar que mide tanto la precisión como el *recall* en una única cifra, evaluando el rendimiento general del modelo.

- **AP50**: Precisión promedio cuando el umbral de IoU es del 50 %, similar al mAP50 en YOLOv8. Este umbral evalúa si el modelo predice con suficiente precisión a un nivel de coincidencia moderado.
- **APs**: Precisión promedio para objetos pequeños. Evalúa el rendimiento del modelo en detectar correctamente objetos de menor tamaño.
- **APm**: Precisión promedio para objetos de tamaño mediano. Indica el rendimiento del modelo para detectar objetos de dimensiones intermedias.
- **APl**: Precisión promedio para objetos grandes. Mide el rendimiento del modelo al identificar correctamente objetos de mayor tamaño.

### 4.3.1. Creación de un Evaluador Personalizado

El principal objetivo al crear un evaluador personalizado ha sido homogeneizar el análisis entre los estudios de los modelos YOLOv8 y Detectron2, dado que ambos frameworks no proporcionan las mismas métricas para su evaluación. Para garantizar una comparación justa entre los dos modelos, se ha diseñado un evaluador utilizando las imágenes de test reservadas del *dataset*, extrayendo las mismas métricas para ambos casos.

Las métricas seleccionadas incluyen precisión (*precision*), intersección sobre unión (*Intersection over Union* o *IoU*), *recall* y *F1-score*. Estas métricas, detalladas en la Sección 3.3 de este documento, abordan su importancia en la evaluación de modelos de segmentación de imágenes. Aunque YOLOv8 proporciona algunas de estas métricas de forma predeterminada, se ha observado que tienden a ser optimistas en su función de validación. Por este motivo, se ha implementado una metodología unificada que garantiza una evaluación más rigurosa y coherente entre los modelos.

El evaluador personalizado sigue el siguiente flujo de trabajo: en primer lugar, se pasan las imágenes de test por ambos modelos, que devuelven las predicciones de los objetos presentes. Estas predicciones se almacenan en el formato correspondiente para cada modelo. A continuación, se comparan los resultados obtenidos con las etiquetas originales del *dataset* (*ground truths*). Para ello, se crean máscaras binarias tanto para las etiquetas como para las predicciones de cada modelo. Estas máscaras permiten la superposición y comparación de los resultados, facilitando el cálculo de los valores de *True Positives*, *True Negatives*, *False Positives* y *False Negatives*.

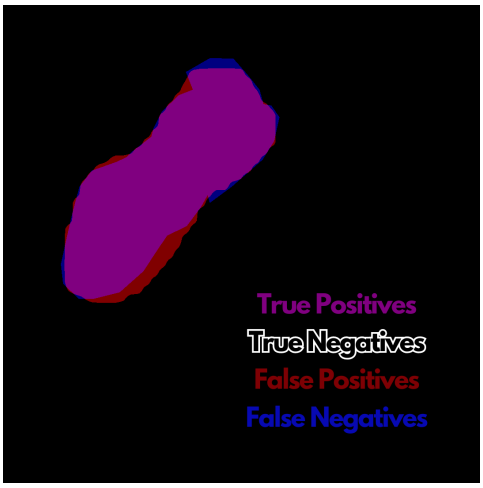


**Figura 4.5.:** Mascaras binarias superpuestas, visualización de la categoría de cada píxel (Elaboración propia)

Con base en estos valores, y como se explicó en el capítulo de *Marco Teórico*, es posible calcular las métricas de precisión, *recall*, *IoU* y *F1-score*, utilizadas para comparar el rendimiento de los dos modelos.

Además de las métricas mencionadas, se ha añadido un temporizador que calcula el tiempo promedio de inferencia para las 46 imágenes de test evaluadas en cada modelo. Este tiempo de inferencia proporciona una métrica adicional para evaluar la eficiencia de los modelos, aspecto fundamental a la hora de considerar su uso en aplicaciones en tiempo real.

También se ha incorporado una visualización gráfica de las máscaras binarias, permitiendo observar de forma más intuitiva la distribución de *False Negatives*, *False Positives*, *True Negatives* y *True Positives*. Esta visualización facilita el análisis cualitativo del rendimiento de los modelos en las diferentes imágenes de test. Además, se han dibujado los contornos de estas máscaras sobre las imágenes originales, proporcionando una representación visual adicional de los resultados.

## 4.4. Implementación en Vídeo a Tiempo Real

Para completar el proyecto, se ha desarrollado un prototipo de sistema de detección de objetos en vídeos en tiempo real, diseñado para simular la detección de pólipos en una endoscopia real. Este programa procesa un vídeo mediante un algoritmo que, tras cargar

el modelo de detección de pólipos, analiza cada frame del vídeo de manera similar a como se procesarían los frames capturados en directo desde una cámara.

El programa emplea un algoritmo de adaptación que mide el tiempo de procesamiento de cada frame en el hardware disponible, permitiendo ajustar el rendimiento del sistema a las condiciones específicas del entorno. Dado que el sistema debe realizar la inferencia del modelo y, además, devolver el frame procesado y pintar la máscara en dicho frame, el programa se ha estructurado en dos hilos principales: uno para la inferencia y otro para la visualización.

El primer hilo se encarga de procesar los frames y medir el tiempo necesario para realizar la inferencia del modelo. El segundo hilo se encarga de pintar la detección o predicción sobre el frame procesado y mostrarlo en pantalla. Esta división permite optimizar el rendimiento y garantizar que la visualización sea fluida y eficiente.

En caso de que el programa no sea capaz de procesar todos los frames en tiempo real, se adapta al ritmo del hardware, procesando únicamente uno de cada ciertos frames. Los frames que no se procesan por el modelo siguen siendo visualizados con la máscara del último frame que sí ha sido analizado. Este enfoque previene problemas de retraso, especialmente si se procesan muy pocos frames por el modelo. Con el hardware utilizado en este estudio, una NVIDIA RTX 3090 Ti, es posible procesar al menos un frame de cada tres en tiempo real, y en algunos casos, incluso todos los frames a la máxima velocidad con el tamaño reducido de los modelos, como el Nano para Detectron2.

Este enfoque garantiza un equilibrio entre la precisión del modelo y la velocidad de procesamiento, adaptándose a las capacidades del hardware y a las necesidades del sistema en tiempo real.

CAPÍTULO 5

# Resultados Y Discusión

## 5.1. Resultados YOLOv8

En esta sección se presentan los rangos de estudio de los hiperparámetros usados para optimizar los modelos de YOLOv8 y esos valores óptimos con los que se han conseguido las mejores métricas. La descripción de estos hiperparámetros se encuentra en la Tabla 4.1. Los rangos de estudio, para YOLOv8, de los hiperparámetros se muestran en la Tabla 5.1.

**Tabla 5.1.:** Hiperparámetros y sus rangos de estudio para YOLOv8

| **Hiperparámetro** | **Rango de Tuning** |
|---|---|
| Número de workers | (16, 24) |
| Tamaño del Batch | (8, 32) |
| Tamaño de Imagen | (320, 1024) |
| Tasa de Aprendizaje (LR) | (1e-5, 1e-2) |
| Momento | (0.7, 0.99) |
| Decaimiento del Peso | (1e-5, 1e-2) |
| Épocas de Calentamiento | (100, 1000) |
| Momento en Calentamiento | (0.0, 0.95) |
| Sesgo de LR en Calentamiento | (1e-5, 1e-2) |
| Optimizador | (RMSProp, Adam, SGD) |
| Umbral de Testeo | (0.4, 0.6) |

La Tabla 5.2 muestra los rangos de los hiperparámetros estudiados para optimizar el rendimiento de YOLOv8. Y en la Tabla 5.3 se ven las métricas correspondientes a dichos resultados.

Para poner en contexto los valores de las métricas obtenidas, en estas tablas solo se están mostrando lo mejores valores, pero comparado con los resultados de los diferentes intentos realizados en la tarea de *Gride Search* se puede decir que según la combinación de hiperparámetros seleccionada los resultados son muy dispares. incluso obteniendo modelos que no superan el 30 % en ninguna de las métricas de este estudio.

**Tabla 5.2.:** Valores Óptimos de Hiperparámetros para YOLOv8 en Diferentes *Datasets*

| **Hiperparámetro** | **Dataset Básico** | | **Dataset Preprocesado** | | **Dataset Augmentation** | |
|---|---|---|---|---|---|---|
| | **Nano** | **Pequeño** | **Nano** | **Pequeño** | **Nano** | **Pequeño** |
| Workers | 21 | 18 | 17 | 21 | 22 | 22 |
| Batch | 24 | 16 | 16 | 24 | 24 | 24 |
| Imagen | 704 | 832 | 448 | 1024 | 448 | 832 |
| LR | 0.00613 | 0.00032 | 0.00001 | 0.00004 | 0.00027 | 0.0000353 |
| Momento | 0.98738 | 0.91128 | 0.98067 | 0.76008 | 0.75335 | 0.83775 |
| Weight Decay | 0.00022 | 0.00006 | 0.00004 | 0.00019 | 0.00002 | 0.00933 |
| Warmup Épocas | 259 | 620 | 899 | 149 | 768 | 197 |
| Warmup Mnt | 0.22596 | 0.42016 | 0.76920 | 0.52820 | 0.46999 | 0.20755 |
| Warmup LR | 0.00051 | 0.00012 | 0.00482 | 0.00427 | 0.00009 | 0.00170 |
| Optimizador | SGD | Adam | RMSProp | RMSProp | Adam | Adam |
| Umbral Test | 0.49812 | 0.48192 | 0.53238 | 0.53946 | 0.55492 | 0.48527 |

Viendo los resultados se puede apreciar la disparidad de la selección que genera los mejores resultados para las diferentes combinaciones de pesos pre-entrenados y los *datasets*. Por ejemplo para unos prefiere un optimizador y para otros uno diferente, los tamaños de la normalización de imagen varían bastante, etc.

En este proyecto el problema principal se debe a la necesidad imperiosa de poder computacional de los modelos de aprendizaje automático como lo son los métodos de visión por computadora. Dados estos problemas a pesar de hacer uso de una versión de *Grid Search Adaptativo*, no se han podido realizar los suficientes intentos para completar todas las combinaciones posibles dentro de los rangos elegidos.

**Tabla 5.3.:** Métricas del mejor intento con YOLOv8 para las diferentes configuraciones

| Métrica | Básico | | Preprocesado | | *Data Augmentation* | |
|---|---|---|---|---|---|---|
| | **Nano** | **Pequeño** | **Nano** | **Pequeño** | **Nano** | **Pequeño** |
| **Recall** | 0.88870 | 0.86150 | 0.85538 | 0.86070 | 0.90914 | 0.88452 |
| **Precisión** | 0.91722 | 0.89066 | 0.91621 | 0.81401 | 0.88694 | 0.84628 |
| **IoU** | 0.82271 | 0.77910 | 0.79333 | 0.71925 | 0.81472 | 0.77038 |
| **F1 Score** | 0.90273 | 0.87583 | 0.88475 | 0.83670 | 0.89790 | 0.83854 |
| **Inferencia (seg)** | 0.01674 | 0.01739 | 0.01231 | 0.01805 | 0.01226 | 0.01541 |

Las métricas obtenidas en la Tabla 5.3 han sido generadas haciendo uso del evaluador personalizado que se presenta en la Sección 4.3.1, en el conjunto de imágenes de test de los diferentes *Dataset*.

Como se puede ver las métricas obtenidas superan los objetivos planteados al inicio del estudio, sobre todo para la combinación de usar los pesos de tamaño Nano y el *Dataset* Básico, cuya Precisión ha llegado a superar el 90 %. El único inconveniente de este modelo en concreto es que en el despliegue tarda 4 milésimas mas que los otros modelos entrenados a partir de los mismos pesos. Aunque no parezca un tiempo significativo, a lo largo del tiempo la suma de varias de estas operaciones computacionales puede llegar a ser percibida.

A continuación se presentan diferentes ejemplos de la segmentación de pólipos realizada por los modelos Nano y Pequeño de YOLOv8, utilizando *datasets* básico, preprocesado y con *Data Augmentation*. Estas figuras permiten observar la capacidad de los modelos para generar máscaras de segmentación en diversos escenarios.

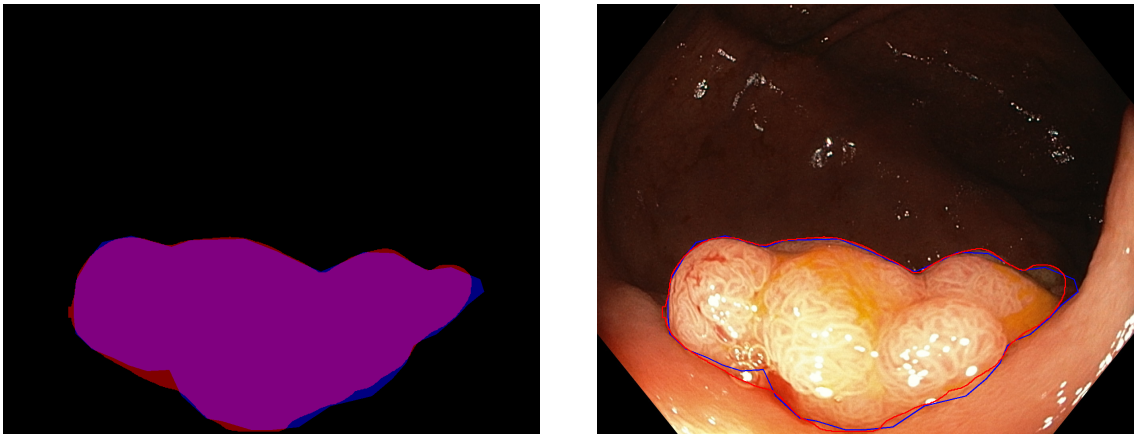

**Figura 5.1.:** Ejemplo del modelo Nano, para el *Dataset* básico

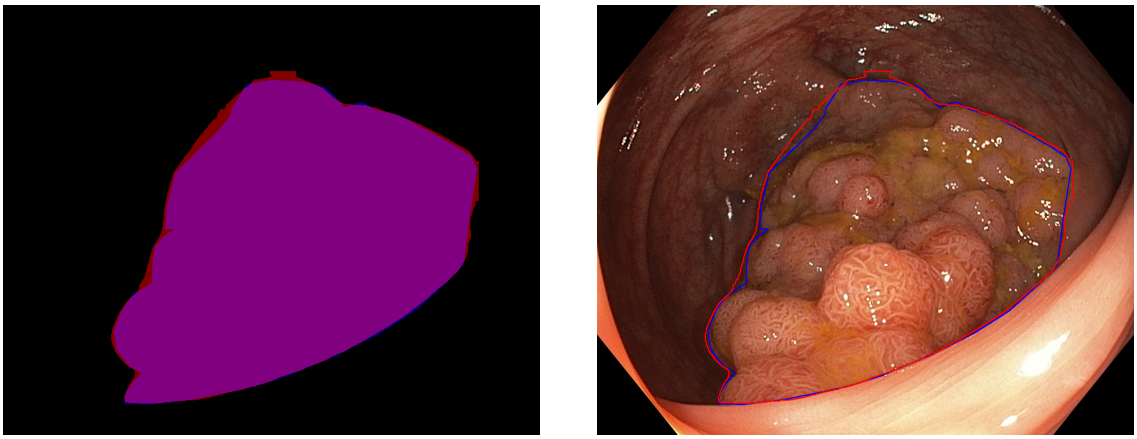

**Figura 5.2.:** Ejemplo del modelo Pequeño, para el *Dataset* básico

En las figuras 5.1 y 5.2, se observan los resultados de la segmentación utilizando el modelo Nano y Pequeño con el dataset básico. El modelo Nano logra suavizar los bordes de la segmentación, aunque con pequeñas imperfecciones, lo cual puede deberse a la resolución del modelo o al preprocesamiento de las imágenes.

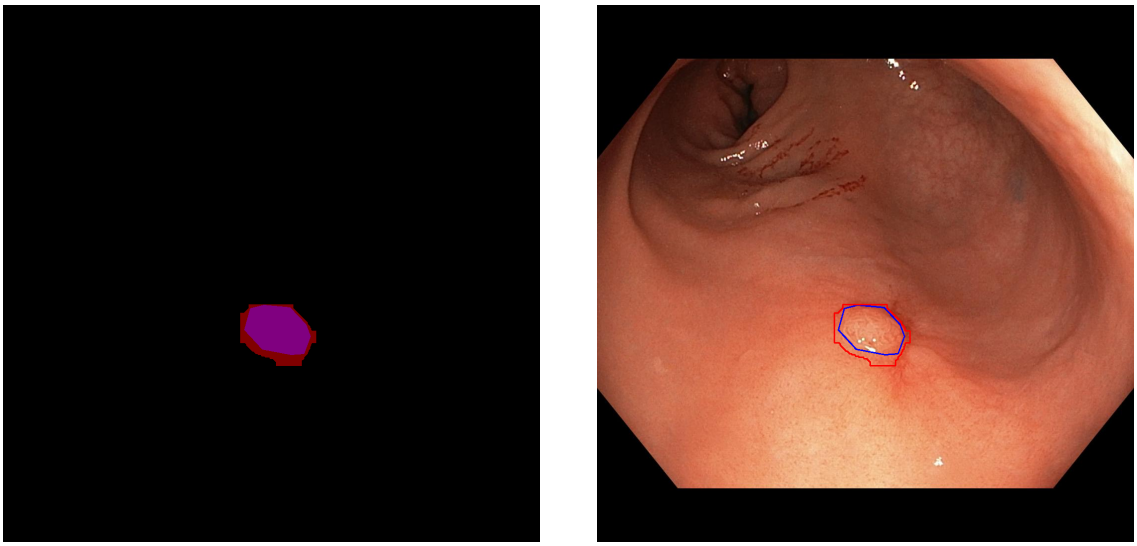

**Figura 5.3.:** Ejemplo del modelo Nano, para el *Dataset* con preprocesado

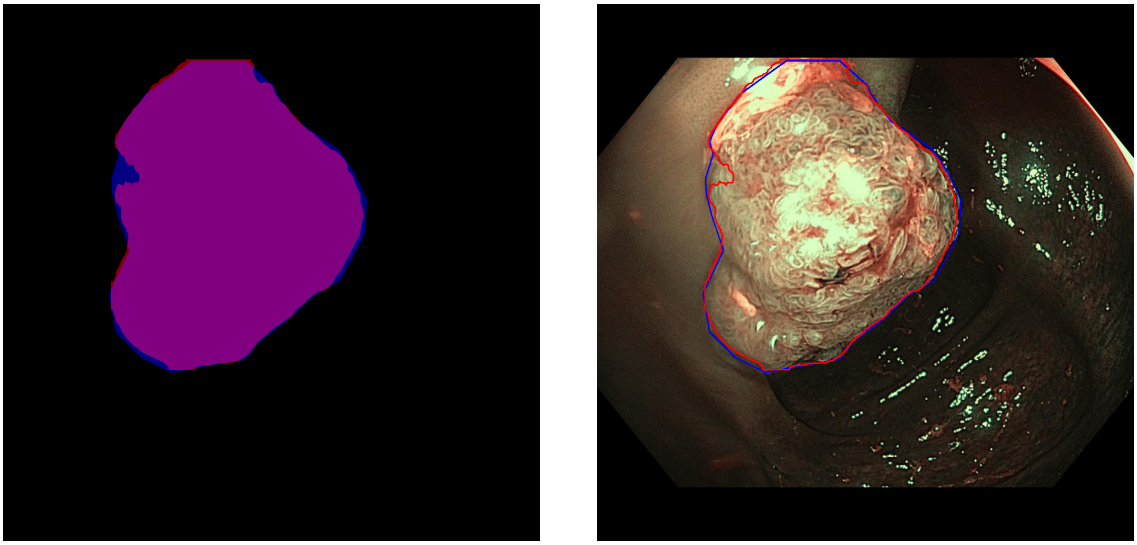

**Figura 5.4.:** Ejemplo del modelo Pequeño, para el *Dataset* preprocesado

Al analizar el *dataset* preprocesado, como se muestra en las figuras 5.3 y 5.4, el modelo Nano evidencia dificultades en la segmentación de pólipos planos, mostrando una forma limitada y menos precisa. Sin embargo, el modelo Pequeño demuestra un mejor ajuste a los bordes de los objetos, aunque persisten pequeñas irregularidades en los bordes de las máscaras.

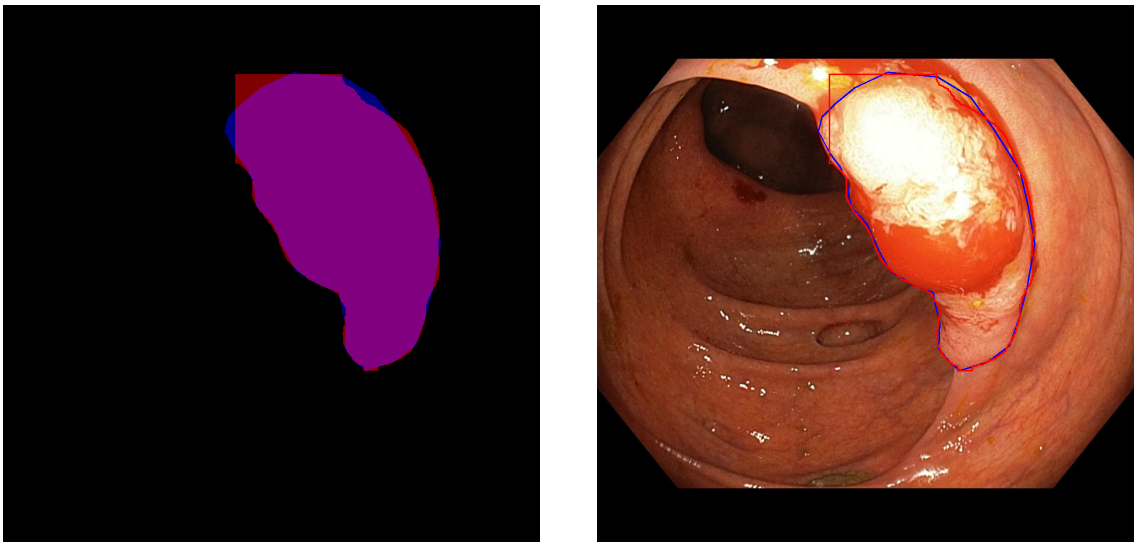

**Figura 5.5.:** Ejemplo del modelo Nano, para el *Dataset* con *Data Augmentation*

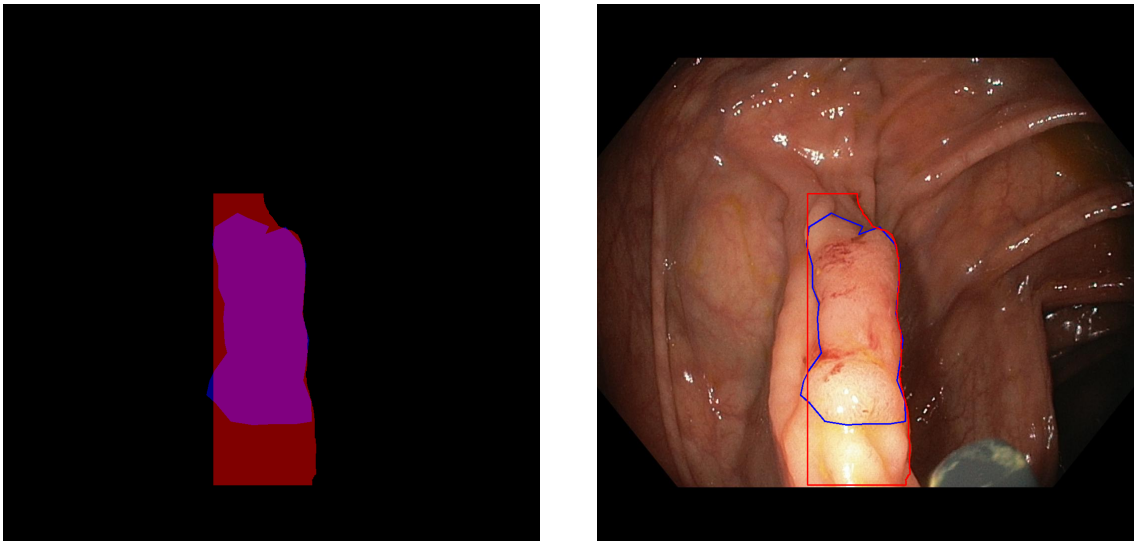

**Figura 5.6.:** Ejemplo del modelo Pequeño, para el *Dataset Data Augmentation*

Finalmente, en las figuras 5.5 y 5.6, se observa el impacto del *Data Augmentation*. A pesar de las mejoras en la variabilidad de los datos, las segmentaciones aún muestran ligeros errores en los bordes. En particular, el modelo Pequeño es en el cual se han encontrado segmentaciones con más imprecisiones.

## 5.2. Resultados Detectron2

En esta sección se presentan los rangos de estudio de los hiperparámetros usados para optimizar los modelos de Detectron2 y esos valores óptimos con los que se han conseguido las mejores métricas. La descripción de estos hiperparámetros se encuentra en la Tabla 4.2. Los rangos de estudio, para Detectron2, de los hiperparámetros se muestran en la Tabla 5.4.

**Tabla 5.4.:** Hiperparámetros y sus rangos de estudio para Detectron2

| **Hiperparámetro** | **Rango de Estudio** |
|---|---|
| Número de workers | (8, 16) |
| Tamaño del Batch | (8, 16) |
| Tamaño de Imagen | (320, 1024) |
| Tasa de Aprendizaje (LR) | (1e-5, 1e-2) |
| Gamma | (0.1, 0.9) |
| Momento | (0.7, 0.99) |
| Decaimiento del Peso | (1e-5, 1e-2) |
| Épocas de Calentamiento | (100, 1000) |
| Momento en Calentamiento | (0.0, 0.95) |
| Optimizador | (Adagrad, Adam, SGD) |
| Umbral de Testeo | (0.4, 0.6) |

La Tabla 5.5 muestra los rangos de los hiperparámetros estudiados para optimizar el rendimiento de Detectron2. Y en la Tabla 5.6 se ven las métricas correspondientes a dichos resultados.

Al igual que en la arquitectura de YOLOv8 los resultados obtenidos en el estudio por *Grid Search* son muy dispares y en estas tablas solo se muestran los hiperparámetros que han resultado en los modelos con los mejores resultados.

Las métricas obtenidas en esta sección, de nuevo, han sido generadas haciendo uso del evaluador personalizado que se presenta en la Sección 4.3.1, en el conjunto de imágenes de test de los diferentes *Dataset*.

**Tabla 5.5.:** Valores Óptimos de Hiperparámetros para Detectron2 en Diferentes *Datasets*

| **Hiperparámetro** | **Dataset Básico** | | **Dataset Preprocesado** | | **Dataset Augmentation** | |
|---|---|---|---|---|---|---|
| | **Nano** | **Pequeño** | **Nano** | **Pequeño** | **Nano** | **Pequeño** |
| Workers | 12 | 12 | 11 | 8 | 9 | 12 |
| Batch | 8 | 10 | 13 | 12 | 15 | 13 |
| Imagen | 648 | 756 | 448 | 457 | 890 | 542 |
| LR | 0.00574 | 0.00079 | 0.00267 | 0.001029 | 0.0073 | 0.00089 |
| Gamma | 0.49164 | 0.32881 | 0.88381 | 0.14064 | 0.52622 | 0.60337 |
| Momento | 0.7076 | 0.8643 | 0.8150 | 0.91994 | 0.7778 | 0.96033 |
| Weight Decay | 0.00002 | 0.00001 | 0.00009 | 0.00015 | 0.00011 | 0.00008 |
| Warmup Época | 727 | 338 | 888 | 696 | 503 | 105 |
| Warmup Mnt | 0.9351 | 0.2306 | 0.9438 | 0.78099 | 0.0670 | 0.19582 |
| Optimizador | Adagrad | Adagrad | SGD | Adagrad | SGD | SGD |
| Umbral Test | 0.5543 | 0.5921 | 0.5322 | 0.57139 | 0.5824 | 0.59047 |

En este caso de estudio lo primero que salta a la vista es que no se ha usado en ninguno de los casos el optimizador de *Adam*, pero esto puede deberse de nuevo a que no hemos probado todas las posibilidades de la matriz de estudio del *Grid Search*.

**Tabla 5.6.:** Métricas del mejor intento con Detectron2 para diferentes configuraciones

| **Métrica** | **Básico** | | **Preprocesado** | | ***Data Augmentation*** | |
|---|---|---|---|---|---|---|
| | **Nano** | **Pequeño** | **Nano** | **Pequeño** | **Nano** | **Pequeño** |
| **Recall** | 0.9041 | 0.8891 | 0.8441 | 0.8614 | 0.8906 | 0.8931 |
| **Precisión** | 0.9148 | 0.8949 | 0.9047 | 0.8787 | 0.9083 | 0.9016 |
| **IoU** | 0.8338 | 0.8051 | 0.7752 | 0.7698 | 0.8171 | 0.8138 |
| **F1 Score** | 0.9094 | 0.8920 | 0.8734 | 0.8699 | 0.8993 | 0.8973 |
| **Inferencia (seg)** | 0.0452 | 0.0454 | 0.0354 | 0.0355 | 0.0352 | 0.0356 |

En la Tabla 5.6 vemos como los resultados de las metricas obtenidos con los modelos de Detectron2 son un poco superiores que los de YOLOv8 en Precisión y Recall. Pero son bastante peores en el campo de tiempo de computación, como se puede observar, por ejemplo en la primera configuración usando el *Dataset* Básico y los pesos pre-entrenados de tamaño Nano, este modelo de Detectron2 tarda tres veces mas en realizar la inferencia que su contraparte de YOLOv8.

Vemos de nuevo que esta configuración es la que mejores resultados ofrece. Es decir que las mejoras que se intentaban obtener realizando un preprocesado y *Data Augmentation* a las imagenes del *Dataset* respecto a las métricas han sido perjudiciales.

A continuación se presentan diferentes ejemplos de la segmentación de pólipos realizada por los modelos Nano y Pequeño de Detectron2, utilizando *datasets* básico, preprocesado y con *Data Augmentation*. Estas figuras permiten observar la capacidad de los modelos para generar máscaras de segmentación en diversos escenarios.

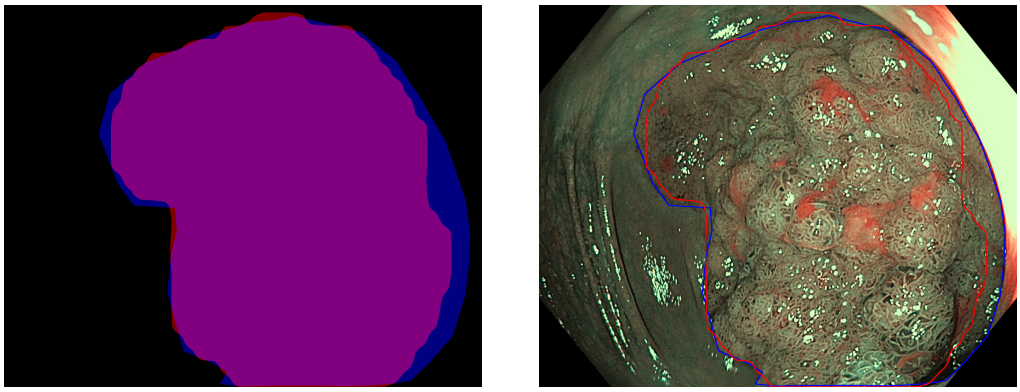

**Figura 5.7.:** Ejemplo del modelo Nano, para el *Dataset* básico

La Figura 5.7 muestra un ejemplo de segmentación realizada por el modelo Nano entrenado con el dataset básico. Se puede apreciar la precisión en la delimitación del pólipo, con una máscara que se ajusta estrechamente a los bordes de la lesión.

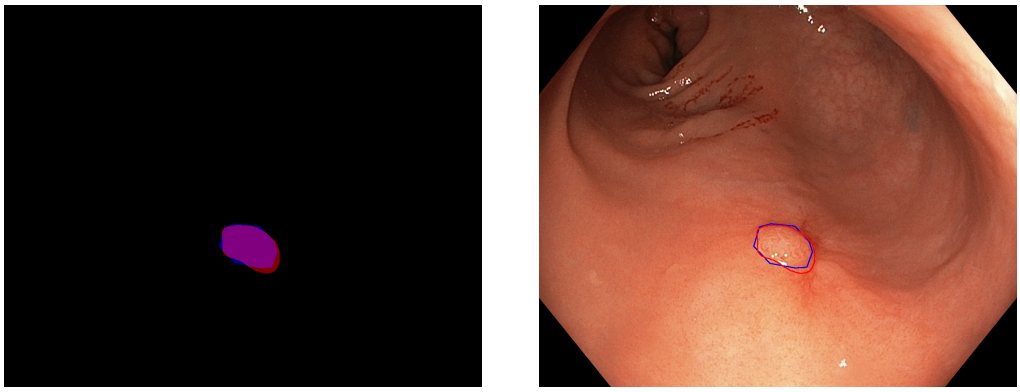

**Figura 5.8.:** Ejemplo del modelo Pequeño, para el *Dataset* básico

En la Figura 5.8, se observa el resultado del modelo Pequeño con el dataset básico. La segmentación es precisa, aunque se pueden notar ligeras diferencias en la delimitación del borde del pólipo en comparación con el modelo Nano.

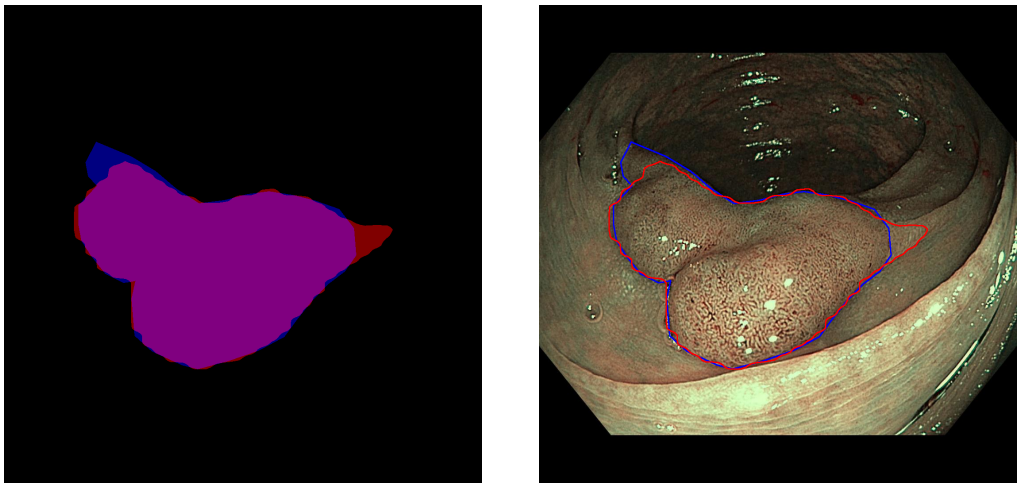

**Figura 5.9.:** Ejemplo del modelo Nano, para el *Dataset* preprocesado

La Figura 5.9 ilustra el rendimiento del modelo Nano con el dataset preprocesado. Se puede observar que la segmentación mantiene un alto nivel de precisión, aunque las métricas generales son ligeramente inferiores a las del dataset básico.

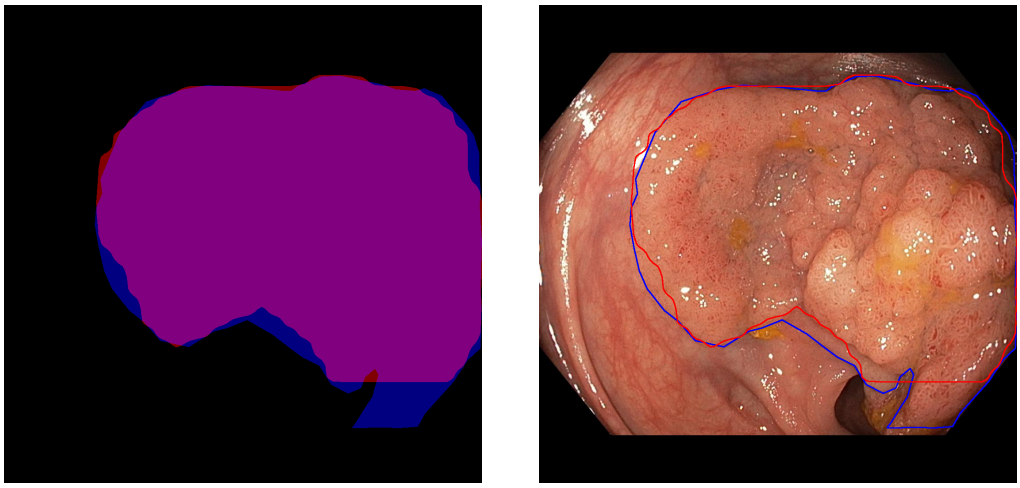

**Figura 5.10.:** Ejemplo del modelo Pequeño, para el *Dataset* preprocesado

En la Figura 5.10, se muestra un ejemplo del modelo Pequeño con el dataset preprocesado. La segmentación es precisa, pero no se observa una mejora significativa respecto al dataset básico.

La Figura 5.11 presenta un ejemplo del modelo Nano entrenado con data augmentation. La segmentación es precisa, y las métricas indican un rendimiento similar al del dataset básico.

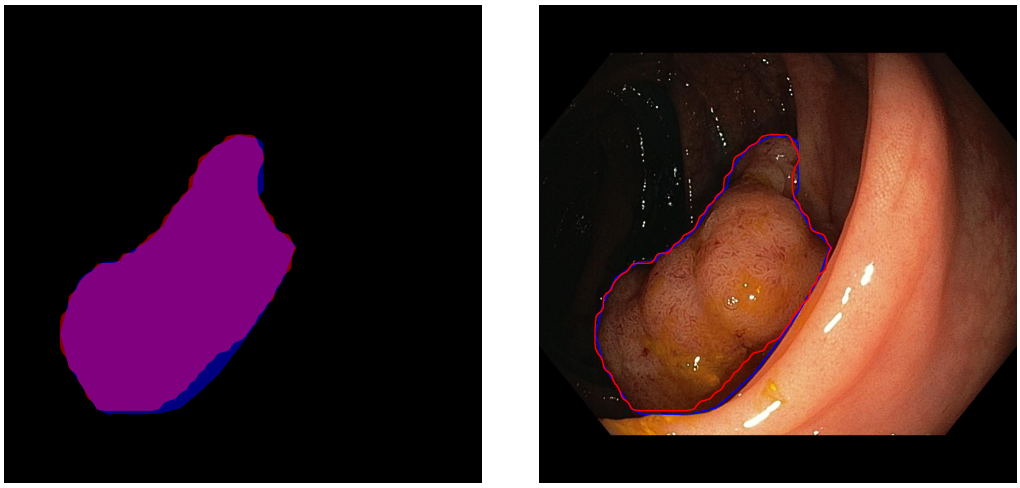

**Figura 5.11.:** Ejemplo del modelo Nano, para el *Dataset Data Augmentation*

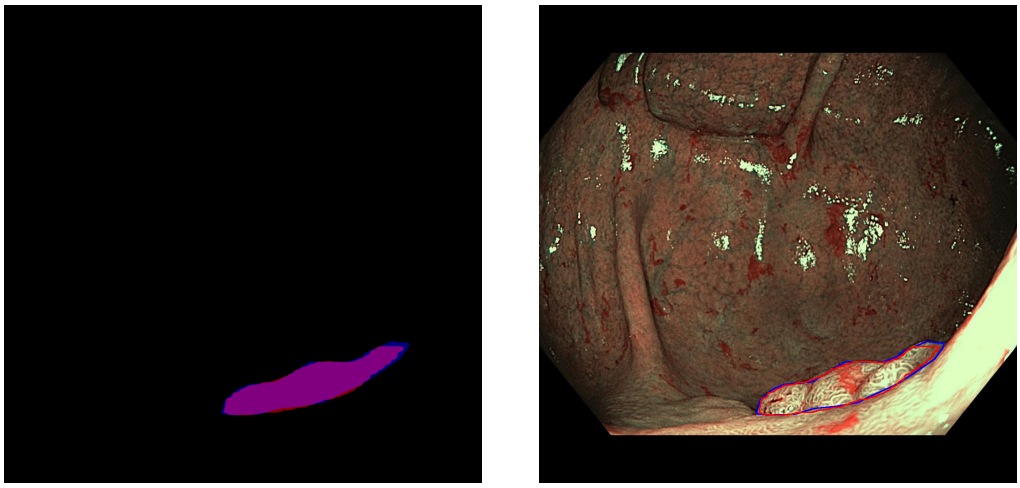

**Figura 5.12.:** Ejemplo del modelo Pequeño, para el *Dataset Data Augmentation*

Finalmente, la Figura 5.12 muestra un ejemplo del modelo Pequeño con data augmentation. La segmentación es precisa y las métricas son comparables a las obtenidas con el dataset básico.

En conclusión, los resultados obtenidos con Detectron2 demuestran su eficacia en la tarea de segmentación de pólipos intestinales. El modelo Nano con el dataset básico ha mostrado el mejor rendimiento general, lo que sugiere que Detectron2 es capaz de extraer características relevantes sin necesidad de preprocesamiento adicional o augmentación de datos en este caso particular. No obstante, es importante señalar que la elección óptima de hiperparámetros varía significativamente entre las diferentes configuraciones, lo que subraya la importancia de una cuidadosa optimización para cada caso específico.

## 5.3. Análisis y Discusión de los resultados

### 5.3.1. Comparación entre YOLOv8 y Detectron2

La comparación entre YOLOv8 y Detectron2 revela ventajas y desventajas distintivas para cada arquitectura:

- **Velocidad de entrenamiento:** Detectron2 muestra una ligera ventaja en términos de velocidad de entrenamiento.
- **Precisión:** YOLOv8 generalmente supera a Detectron2 en términos de precisión en la mayoría de los escenarios evaluados.
- **Tiempo de inferencia:** YOLOv8 destaca significativamente con tiempos de inferencia considerablemente más bajos, alineándose con su filosofía de "You Only Look Once".
- **Facilidad de uso:** YOLOv8 ofrece una interfaz más intuitiva y una documentación más completa, respaldada por una comunidad más amplia y activa.
- **Métricas proporcionadas:** YOLOv8 proporciona un conjunto más amplio de métricas y visualizaciones, incluyendo matrices de confusión y curvas de precisión-recall, que no pudieron ser replicadas completamente en el evaluador personalizado desarrollado para este estudio.

Considerando estos factores, YOLOv8 emerge como la opción preferida para esta aplicación específica de detección de pólipos.

### 5.3.2. Mejor modelo del Estudio

Se ha determinado que el modelo que presenta los mejores resultados es aquel entrenado con YOLOv8 a partir *dataset* básico y de los pesos preentrenados de tamaño Nano. Este modelo destaca por obtener las mejores métricas en Precisión, *Recall*, intersección sobre la unión ($IoU$) y la puntuación (*F1-score*). Si bien no alcanza los tiempos de inferencia más rápidos, la pérdida en velocidad es mínima en comparación con otros modelos que emplean los mismos pesos preentrenados.

En el repositorio donde se encuentra el código utilizado para generar todos estos resultados, también se incluyen los vídeos generados para replicar situaciones de desplie-

gue en tiempo real. Estos vídeos han sido fundamentales para visualizar y evaluar el comportamiento de los modelos en un entorno práctico, lo que ha sido crucial para la selección del modelo más adecuado.

A pesar de los resultados prometedores, se ha identificado cierto nivel de sobreajuste (*overfitting*) en este modelo, así como en los demás modelos considerados en este estudio. Para abordar esta situación, se ha centrado la atención en el análisis del modelo nano con el *dataset* básico, con el fin de investigar las causas subyacentes de este fenómeno. Al observar las curvas de pérdida durante el entrenamiento [50], particularmente en la fase de segmentación, se han obtenido conclusiones que permiten una comprensión más profunda del comportamiento del modelo en cuestión.

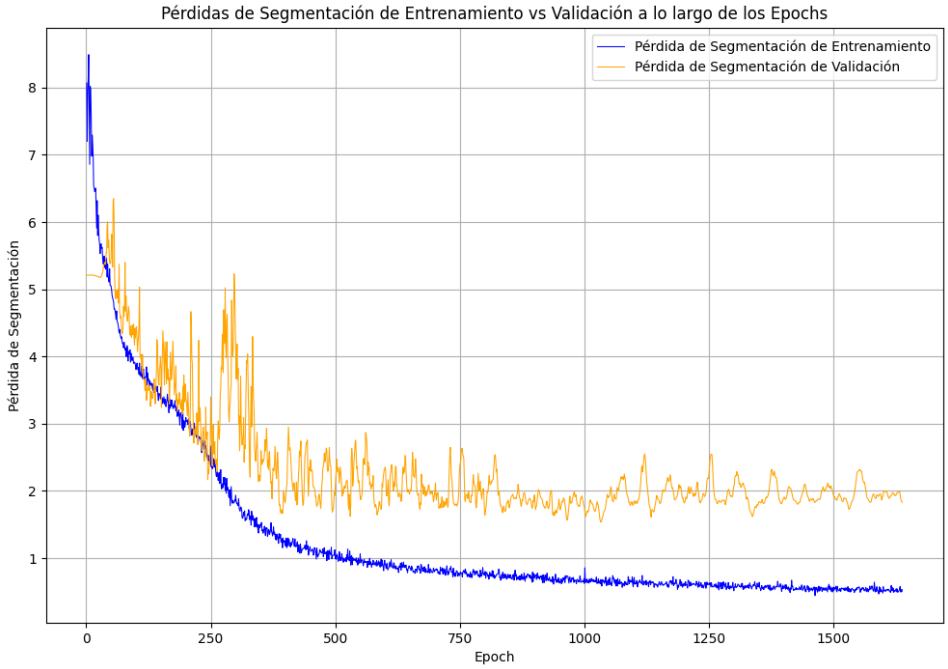


**Figura 5.13.:** Pérdida de segmentación durante el entrenamiento y validación para el modelo YOLOv8 Nano con el *Dataset* básico (Elaboración propia)

En la Figura 5.13 que muestra las pérdidas de segmentación a lo largo de las épocas, se observa una posible presencia de sobreajuste (overfitting). Como se indica, estas curvas pertenecen al proceso de entrenamiento del modelo con mejor Precisión mostrado en la Tabla 5.3. Esto se refleja en el comportamiento de las curvas de pérdida del entrenamiento y la validación para el modelo con mejores resultados, para los demás modelos generados los resultados son similares en este aspecto, para ambas arquitecturas. Aunque ambas curvas presentan una tendencia descendente hasta aproximadamente las 500 épocas, el

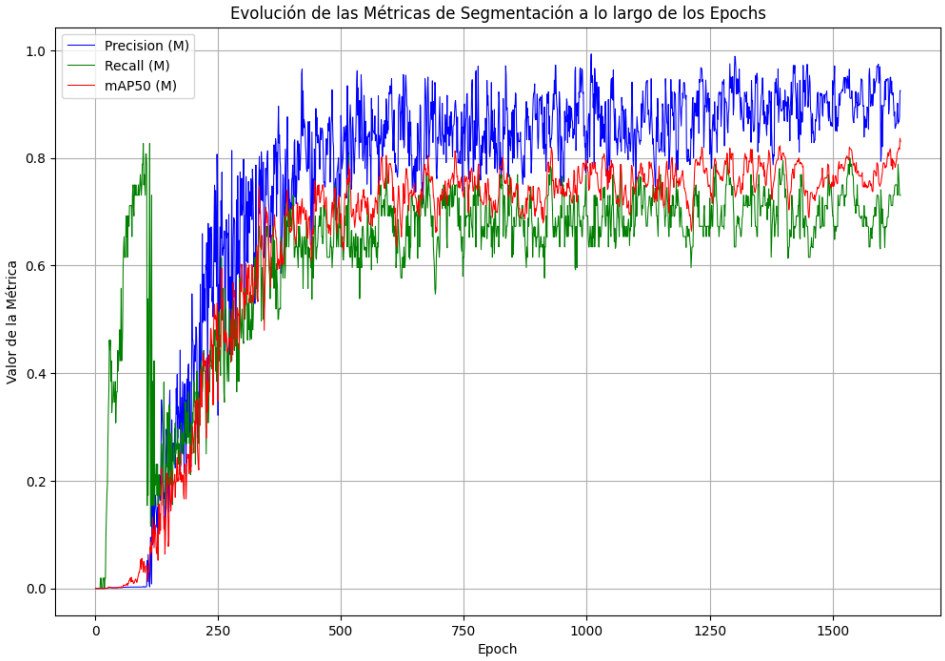


**Figura 5.14.:** Evolución de las métricas de segmentación (Precisión, Recall y mAP50) durante el entrenamiento del modelo YOLOv8 Nano con el *Dataset* básico (Elaboración propia)

modelo continúa entrenando hasta pasadas las 1500 épocas sin una mejora significativa en la pérdida de validación.

La curva de validación no presenta divergencia, lo cual indica que no empeora la calidad del modelo, pero el aumento en el área entre ambas curvas sugiere que el entrenamiento podría haberse detenido antes para evitar el sobreajuste, específicamente alrededor de las 1000 épocas, activando el mecanismo de *Early Stopping*.

El criterio de paciencia para este experimento fue de 250 épocas, lo que habría permitido detener el entrenamiento antes de que las métricas de validación dejaran de mejorar. Sin embargo, este mecanismo no fue activado, lo que permitió que el modelo continuara entrenando sin un beneficio claro en las pérdidas de validación. No fue activado hasta que las métricas mostradas en la Figura 5.14 dejaron de mejorar durante 250 epocas y saltó el *Early Stopping*.

### 5.3.3. Evolución a lo largo de los intentos de entrenamiento

Para ambas arquitecturas, YOLOv8 y Detectron2, se implementó la biblioteca Optuna para la optimización de hiperparámetros. Sin embargo, es importante señalar que los resultados obtenidos no mostraron la progresión esperada a lo largo de los diferentes intentos. Como se puede observar en las figuras adyacentes, no se evidencia una mejora consistente en las métricas de rendimiento conforme avanzan los intentos.

La función objetivo utilizada para la optimización se basó en la *mean Average Precision* al 50 % de *IoU*. No obstante, este enfoque no logró producir mejoras significativas en los intentos subsiguientes. Se realizaron experimentos adicionales utilizando una media entre precisión y *Recall* como métrica de optimización, pero estos tampoco arrojaron resultados satisfactorios.

Este comportamiento sugiere que la implementación actual de las funciones de Optuna se asemeja más a una búsqueda en cuadrícula *grid search* estándar que a una optimización bayesiana efectiva. Esta observación plantea la necesidad de reevaluar y posiblemente refinar la estrategia de optimización de hiperparámetros en futuras iteraciones del estudio.

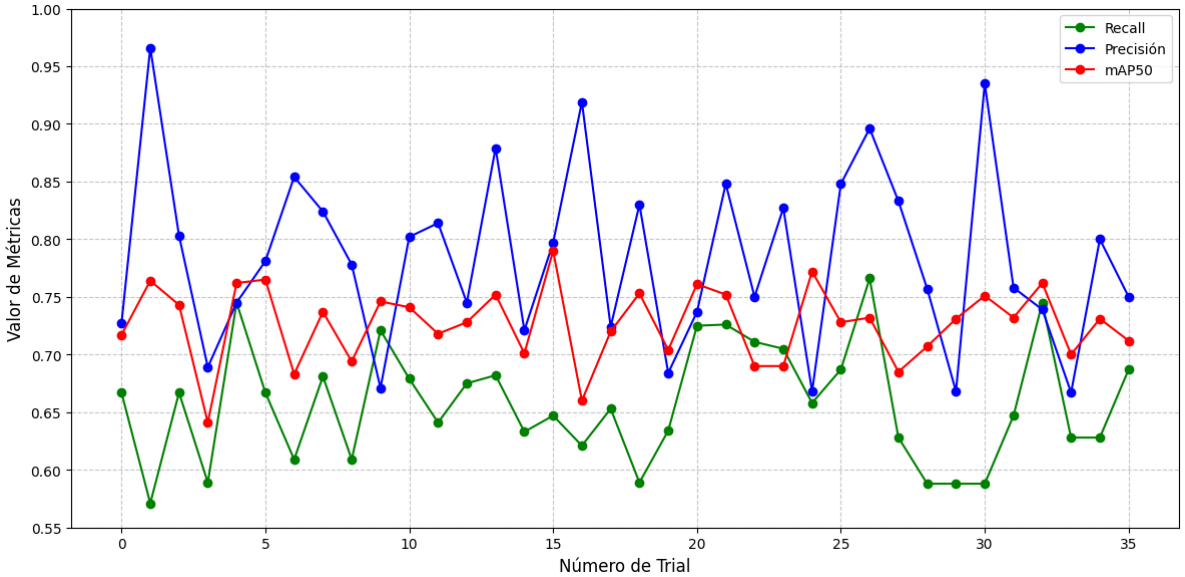


**Figura 5.15.:** Evolución con el paso de los intentos del estudio por *Grid Search* (Elaboración propia)

El hecho de que estas curvas de la Figura 5.15 no tengan una tendencia de convergencia positiva, sino que presentan un comportamiento aparentemente aleatorio con valores

nulos intermedios, puede atribuirse a varios factores relacionados con la naturaleza de la búsqueda en cuadrícula y los hiperparámetros seleccionados para el estudio:

- Diversidad de hiperparámetros: La búsqueda en cuadrícula probablemente incluyó una combinación de hiperparámetros numéricos y categóricos. Los hiperparámetros categóricos, como el tipo de optimizador o las funciones de activación, pueden causar cambios abruptos en el rendimiento del modelo entre pruebas consecutivas, lo que explica las fluctuaciones observadas.
- Interacciones complejas: Los hiperparámetros en modelos de visión por computadora a menudo tienen interacciones no lineales y complejas. La modificación de un solo parámetro puede tener efectos impredecibles en el rendimiento del modelo, especialmente cuando se combina con otros cambios.
- Sensibilidad a la inicialización: Algunos modelos de segmentación de objetos son particularmente sensibles a la inicialización de pesos. Si la búsqueda en cuadrícula no controla específicamente la semilla aleatoria, esto puede introducir variabilidad adicional en los resultados.
- Tamaño y complejidad del conjunto de datos: Si el conjunto de datos utilizado es pequeño o presenta una gran variabilidad en la complejidad de las imágenes y objetos a segmentar, esto puede resultar en un rendimiento inconsistente entre diferentes configuraciones de hiperparámetros.
- Overfitting y underfitting: Las fluctuaciones extremas en el rendimiento, incluyendo los valores nulos, pueden indicar casos de overfitting severo (donde el modelo se ajusta demasiado a los datos de entrenamiento) o underfitting (donde el modelo no logra capturar la complejidad del problema).

### 5.3.4. Razonamiento sobre la no utilización de curvas de confianza

En esta investigación, no se ha hecho uso de las curvas relacionadas con las métricas de confianza, ya que tanto en YOLO como en Detectron2, la confianza que devuelven estos modelos está basada en las *bounding boxes*. Este enfoque no resulta adecuado para el contexto de evaluación de segmentación, que es el objetivo principal de este estudio.

A modo de ejemplo, en una de las imágenes, Figura 5.16, generadas por el modelo YOLOv8 con el *dataset* básico , se puede observar una segmentación incorrecta. Aunque el modelo no logra una segmentación precisa y presenta un número considerable de falsos negativos, la *bounding box* generada abarca más del 50 % de la región que cubre la máscara de *ground truth*. Esto da lugar a una métrica de *Intersection over Union (IoU)* superior al 50 %, lo cual es engañoso ya que el verdadero objetivo de la evaluación es la segmentación, no simplemente la detección con cajas.

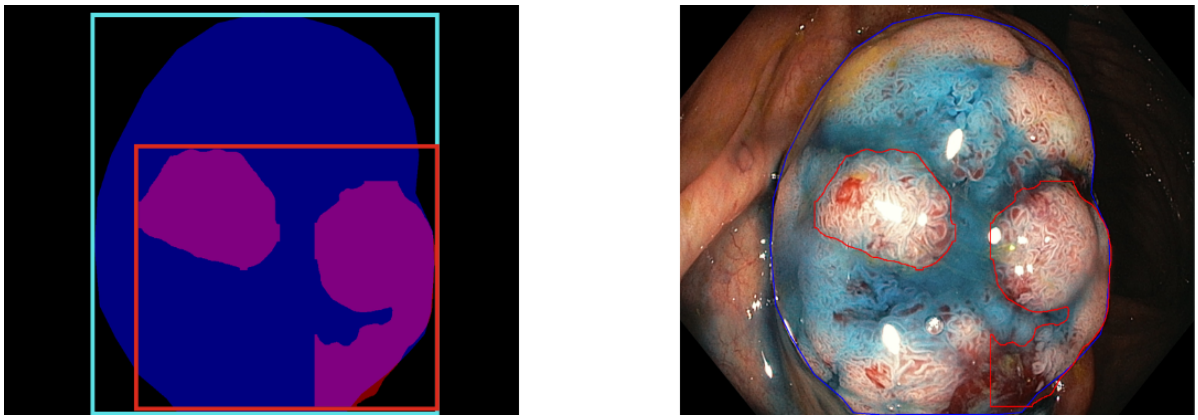

**Figura 5.16.:** Poca fiabilidad de las *boundig boxes* (Elaboración propia)

Este comportamiento es la razón por la cual no es adecuado basar las conclusiones en este tipo de gráficas de confianza. Si bien se podría haber optado por usar una gráfica de precisión-*recall*, esta no aporta información adicional relevante para el análisis. En este estudio, nos hemos centrado más en la media de las detecciones realizadas en todas las imágenes de prueba, dado que nuestro interés radica en el rendimiento global del modelo. Los mejores resultados individuales de precisión y *recall* no son suficientes para evaluar el éxito general en la tarea de segmentación.

Por lo tanto, se decidió no incluir este tipo de gráficas, ya que podrían ofrecer una visión sesgada o incompleta de los resultados reales en términos de segmentación precisa.

### 5.3.5. Tiempos de inferencia y capacidad de detección en tiempo real

Los resultados corroboran la superioridad de YOLOv8 en términos de tiempo de inferencia, un factor crítico para aplicaciones en tiempo real como la detección de pólipos durante endoscopias. Es importante señalar que los tiempos de inferencia reportados no incluyen el procesamiento adicional requerido para la visualización, como la superposición de la

segmentación en el frame y su presentación en pantalla. Este procesamiento adicional introduce un retraso que debe ser considerado en la evaluación del rendimiento en tiempo real.

**Despliegue en entornos clínicos reales**

Para validar la robustez de los modelos entrenados, se realizaron pruebas utilizando vídeos de endoscopias de dominio público. Los resultados obtenidos en estas pruebas fueron consistentes con los tiempos de inferencia observados en el conjunto de datos de prueba, lo que sugiere que los modelos no están sobre-ajustados y que la división de los conjuntos de datos fue apropiada.

Sin embargo, es crucial destacar que el rendimiento óptimo en entornos clínicos reales depende en gran medida de la similitud entre las condiciones de captura de las imágenes de entrenamiento y las de uso real. Factores como el tipo de cámara, la iluminación y las características de la lente pueden afectar significativamente la precisión de la detección. Por lo tanto, para una implementación clínica exitosa, es fundamental utilizar equipos de captura de imagen similares a los empleados en la generación del *dataset* de entrenamiento.

### 5.3.6. Correlación entre métricas y tiempos de inferencia

Se ha observado una correlación interesante entre las métricas de rendimiento y los tiempos de inferencia. En casos donde los modelos están sobreajustados, pueden mostrar métricas de evaluación elevadas en el conjunto de prueba, pero fallar en la detección cuando se aplican a videos nuevos. Paradójicamente, esto puede resultar en tiempos de inferencia más rápidos, ya que el modelo procesa rápidamente frames donde no detecta objetos de interés.

Para mitigar este problema, todos los modelos presentados en este estudio fueron validados utilizando videos de endoscopias públicamente disponibles. La consistencia en los tiempos de inferencia entre el conjunto de prueba y estos videos externos sugiere que los modelos no están sobreajustados y que mantienen su capacidad de generalización.

Esta observación resalta la importancia de una evaluación integral que vaya más allá de las métricas estándar y considere el rendimiento en condiciones lo más cercanas posible a las de uso real.

CAPÍTULO 6

# Conclusiones y Líneas Futuras

## 6.1. Conclusiones

El presente estudio ha arrojado diversas conclusiones significativas en el ámbito de la detección de pólipos mediante técnicas de visión por computadora:

1. **Eficacia en la detección de pólipos planos:** A pesar de la ausencia de segmentación semántica en el proyecto, se ha logrado una detección precisa de pólipos planos, que son tradicionalmente más difíciles de identificar. Este hallazgo fue confirmado por expertos subrayando la relevancia clínica de los resultados obtenidos.

2. **Redundancia en el preprocesamiento de datos:** Se ha observado que los modelos YOLOv8 y Detectron2 realizan transformaciones intrínsecas durante el proceso de entrenamiento. Estas incluyen ajustes de tamaño, rotaciones y otras modificaciones similares a las aplicadas en el preprocesamiento manual. Esta redundancia sugiere que el *dataset* básico, sin transformaciones adicionales, podría ser más efectivo para el entrenamiento y la detección en tiempo real.

3. **Comparación entre segmentación y detección con bounding boxes:** Se ha constatado que los modelos de segmentación proporcionan métricas inferiores en comparación con la detección básica mediante bounding boxes. Esta discrepancia justifica la implementación del evaluador personalizado, que ha sido fundamental para comparar los resultados de manera equitativa.

4. **Limitaciones hardware en el estudio de modelos de mayor escala:** Las restricciones de hardware han impedido un análisis exhaustivo de modelos de tamaño medio y superior. Los tiempos de entrenamiento observados (4-5 horas para modelos nano, 8 horas para pequeños, y 16 horas para medianos) han sido prohibitivos para un estudio más amplio dentro del marco temporal del proyecto.

5. **Importancia del workflow reproducible:** El desarrollo de un workflow que facilita la reproducibilidad de los resultados se ha revelado como un aspecto crucial del proyecto. Este enfoque permitirá la actualización continua de los modelos con nuevas muestras, mejorando la robustez y aplicabilidad de la investigación.

6. **Segmentación y democratización de la IA en aplicaciones médicas:** La implementación de algoritmos de código abierto y de propósito general para la segmentación de pólipos abre la posibilidad a estudios en *explainability*, contribuyendo a la democratización de la inteligencia artificial en aplicaciones de *health care*.

## 6.2. Líneas Futuras

Basándose en los resultados y las limitaciones identificadas, se proponen las siguientes líneas de investigación futura:

1. **Optimización de hiperparámetros:** Implementar correctamente el estudio con reglas bayesianas para la optimización de hiperparámetros. Aunque se intentó utilizar este enfoque, los resultados no fueron significativamente superiores a una búsqueda aleatoria. Una implementación más rigurosa podría mejorar sustancialmente el rendimiento de los modelos.

2. **Evaluación en tiempo real:** Adaptar el código para realizar pruebas con cámaras o webcams en entorno de laboratorio, utilizando hardware especializado. Esto permitiría obtener mediciones más precisas de los frames por segundo (FPS) alcanzables, incluyendo el tiempo de procesamiento para la visualización de las detecciones.

3. **Ampliación del estudio a otros modelos:** Expandir la investigación para incluir arquitecturas de vanguardia como Segment Anything, YOLOv9, o técnicas de segmentación instantánea. Esto podría proporcionar información adicional valiosa, como el conteo preciso de pólipos en una sola endoscopia.

4. **Mejora y expansión del *dataset*:**
    - Aumentar la cantidad de imágenes en el *dataset*.
    - Implementar segmentación semántica para identificar y balancear diferentes tipos de pólipos.
    - Utilizar herramientas como *Roboflow* para facilitar el balanceo y la ampliación del *dataset*.
5. **Estudio de modelos de mayor escala:** Realizar un análisis más exhaustivo de modelos de tamaño medio y grande, posiblemente mediante la utilización de múltiples unidades de hardware en paralelo para reducir los tiempos de entrenamiento.
6. **Mejora de las métricas de evaluación:** Estar atentos a las actualizaciones de los repositorios de YOLOv8 y Detectron2 para incorporar nuevas métricas de evaluación a medida que se desarrollen, aprovechando las ventajas del código abierto y la flexibilidad de los modelos de propósito general.

Estas líneas de investigación futuras tienen el potencial de mejorar significativamente la precisión y aplicabilidad de la detección automática de pólipos, contribuyendo así al avance de las técnicas de diagnóstico en gastroenterología.

# Bibliografía


[1] Ali, Syed Zahid: *Principles of yolov8*. https://medium.com/@syedzahidali969/principles-of-yolov8-6a90564e16c3, 2023. Accessed: 2024-09-20.

[2] Ultralytics: *Yolov8 models documentation*, 2024. https://docs.ultralytics.com/models/yolov8/, Accessed: September 18, 2024.

[3] Schwert, Hiroto: *Digging into detectron 2*, 2020. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd, Accessed: 2024-09-13.

[4] Vidhya, Analytics: *Tune hyperparameters with gridsearchcv*, 2021. https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/, Accessed: September 18, 2024.

[5] Kouidri, Allan: *Top instance segmentation models of 2024: A comprehensive guide*, 2024. https://www.ikomia.ai/blog/top-instance-segmentation-models#best-real-time-instance-segmentation-models, Accessed: 2024-09-05.

[6] Siegel, Rebecca L., Miller, Kimberly D., Sauer, Ann Goding, Fedewa, Stacey A., Butterly, Lynn F., Anderson, Joseph C., Cercek, Andrea, Smith, Robert A. y Jemal, Ahmedin: *Colorectal cancer statistics, 2020*. CA: A Cancer Journal for Clinicians, 70(3):145–164, 2020. https://acsjournals.onlinelibrary.wiley.com/doi/full/10.3322/caac.21601, Accessed: 2024-09-20.

[7] Biller, Leah H. y Schrag, Deborah: *Diagnosis and treatment of metastatic colorectal cancer: A review*. JAMA, 325(7):669–685, 2021. https://jamanetwork.com/journals/jama/article-abstract/2776334, Accessed: 2024-09-20.

[8] MetaAI: *Detectron2 models documentation*, 2024. https://ai.meta.com/tools/detectron2/, Accessed: September 18, 2024.

[9] Hearty, John: *Practical Machine Learning*. O'Reilly Media, 2021. https://www.oreilly.com/library/view/practical-machine-learning/9781098102357/ch04.html, Accessed: 2024-09-05.

[10] Ultralytics: *What are the key advantages of using ultralytics yolov8 for real-time object detection?*, 2024. https://docs.ultralytics.com/es/guides/yolo-performance-metrics/#what-are-the-key-advantages-of-using-ultralytics-yolov8-for-real-time-object-de Accessed: 2024-09-20.

[11] Ciencias In Seso: *F1 score*, 2024. https://www.cienciasinseso.com/f1_score/, Accessed: 2024-09-20.

[12] Connor Shorten, Taghi M. Khoshgoftaar Borko Furht: *A Comprehensive Review on Machine Learning Applications*. Springer, 2021. https://link.springer.com/article/10.1186/s40537-021-00492-0, Accessed: September 18, 2024.

[13] Rayed, Md. Eshmam, Islam, S.M. Sajibul, Niha, Sadia Islam, Jim, Jamin Rahman, Kabir, Md Mohsin y Mridha, M.F.: *Deep learning for medical image segmentation: State-of-the-art advancements and challenges*. Informatics in Medicine Unlocked, 47:101504, 2024.

[14] Esteva, Andre, Chou, Katherine, Yeung, Serena, Naik, Nikhil, Madani, Ali, Mottaghi, Ali, Liu, Yun, Topol, Eric, Dean, Jeff y Socher, Richard: *Deep learning-enabled medical computer vision*. npj Digital Medicine, 4:5, 2021.

[15] Ghose, Priyanka, Ghose, Arpan, Sadhukhan, Deboleena, Pal, Saurabh y Mitra, Madhuchanda: *Improved polyp detection from colonoscopy images using finetuned yolo-v5*. Multimedia Tools and Applications, 83:42929–42954, 2023. https://link.springer.com/article/10.1007/s11042-023-17138-3.

[16] Keymakr: *Semantic segmentation vs object detection: Understanding the differences*, 2024. https://keymakr.com/blog/semantic-segmentation-vs-object-detection-understanding-the-differences/, Accessed: 2024-09-05.

[17] Microsoft: *Beit-3: Image as a foreign language*. https://github.com/microsoft/unilm/tree/master/beit3, 2023. Accessed: 2024-09-20.

[18] AI, Meta: *Segment anything*. https://segment-anything.com/, 2023. Accessed: 2024-09-20.

[19] Szeliski, Richard: *Computer Vision: Algorithms and Applications*. Springer Nature, 2022.

[20] Sánchez-Peralta, L. F., Bote-Curiel, L., Picón, A., Sánchez-Margallo, F. M. y Pagador, J. B.: *Deep learning to find colorectal polyps in colonoscopy: A systematic literature review*. Artificial Intelligence in Medicine, 108:101923, 2020.

[21] Gonzalez, Rafael C. y Woods, Richard E.: *Digital Image Processing*. Pearson, 2018.

[22] Xiao, Y., Wu, J., Lin, Z. y Zhao, X.: *A deep learning-based multi-model ensemble method for cancer prediction*. Computer Methods and Programs in Biomedicine, 153:1–9, 2018.

[23] Batard, T. y Berthier, M.: *Spinor fourier transform for image processing*. IEEE Journal of Selected Topics in Signal Processing, 12(4):800–816, 2018.

[24] Hesamian, M. H., Jia, W., He, X. y Kennedy, P.: *Deep learning techniques for medical image segmentation: Achievements and challenges*. Journal of Digital Imaging, 32(4):582–596, 2019.

[25] Sotiras, A., Davatzikos, C. y Paragios, N.: *Deformable medical image registration: A survey*. IEEE Transactions on Medical Imaging, 32(7):1153–1190, 2017.

[26] Xu, H., Wang, Y., Jian, W., Wu, X. y Lv, X.: *Edge detection of color image based on quaternion fractional differential*. Mathematical Problems in Engineering, 2020.

[27] Huang, P. W., Thanh, D. N. H. y Yang, C. H.: *Texture feature extraction for land cover classification of remote sensing images based on multi-scale analysis techniques*. Remote Sensing, 12(17):2760, 2020.

[28] Ding, M., Wang, K., Wang, J., Liao, X., Zhang, L. y Han, J.: *Shape representation and matching based on skeleton graph convolution network*. IEEE Transactions on Multimedia, 2021.

[29] Tareen, S. A. K. y Saleem, Z.: *A comparative analysis of sift, surf, kaze, akaze, orb, and brisk*. En *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, páginas 1–10. IEEE, 2018.

[30] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. y Pietikäinen, M.: *Deep learning for generic object detection: A survey*. International Journal of Computer Vision, 128(2):261–318, 2020.

[31] Hui, Jonathan: *Image segmentation with mask r-cnn.* https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-ebe6d793272, 2023. Accessed: 2024-09-20.

[32] GeeksforGeeks: *Deep dream: An in-depth exploration.* https://www.geeksforgeeks.org/deep-dream-an-in-depth-exploration/, 2021. Accessed: 2024-09-20.

[33] Garcia-Garcia, Alberto, Orts-Escolano, Sergio, Oprea, Sergiu, Villena-Martinez, Victor y Garcia-Rodriguez, Jose: *A review on deep learning techniques applied to semantic segmentation.* arXiv preprint arXiv:1704.06857, 2017. https://arxiv.org/abs/1704.06857.

[34] Ali, Syed Zahid: *Principles of yolov8.* Medium, 2023. https://medium.com/@syedzahidali969/principles-of-yolov8-6a90564e16c3.

[35] Ultralytics: *Segmentación de instancias con YOLOv8*, 2024. https://docs.ultralytics.com/es/tasks/segment/.

[36] Sapkota, Ranjan, Ahmed, Dawood y Karkee, Manoj: *Comparing yolov8 and mask rcnn for object segmentation in complex orchard environments.* arXiv preprint arXiv:2312.07935, 2023. https://arxiv.org/abs/2312.07935.

[37] Taha, A. A. y Hanbury, A.: *Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool.* BMC Medical Imaging, 15(1):29, 2015.

[38] Yang, L. y Shami, A.: *On hyperparameter optimization of machine learning algorithms: Theory and practice.* Neurocomputing, 415:295–316, 2020.

[39] Feurer, M. y Hutter, F.: *Hyperparameter optimization.* En *Automated Machine Learning*, páginas 3–33. Springer, 2019.

[40] Bouthillier, X. y Varoquaux, G.: *Survey of machine-learning experimental methods at neurips2019 and iclr2020.* Informe técnico, Inria Saclay Ile de France, 2020.

[41] Bergstra, J. y Bengio, Y.: *Random search for hyper-parameter optimization.* Journal of Machine Learning Research, 13(Feb):281–305, 2015.

[42] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. y Talwalkar, A.: *Hyperband: A novel bandit-based approach to hyperparameter optimization.* The Journal of Machine Learning Research, 18(1):6765–6816, 2017.

[43] Real, E., Aggarwal, A., Huang, Y. y Le, Q. V.: *Regularized evolution for image classifier architecture search*. En *Proceedings of the AAAI Conference on Artificial Intelligence*, volumen 33, páginas 4780–4789, 2019.

[44] Snoek, J., Larochelle, H. y Adams, R. P.: *Practical bayesian optimization of machine learning algorithms*. Advances in neural information processing systems, 25, 2018.

[45] Frazier, P. I.: *A tutorial on bayesian optimization*. arXiv preprint arXiv:1807.02811, 2018.

[46] Wu, J., Chen, X. Y., Zhang, H., Xiong, L. D., Lei, H. y Deng, S. H.: *Hyperparameter optimization for machine learning models based on bayesian optimization*. Journal of Electronic Science and Technology, 17(1):26–40, 2019.

[47] Akiba, T., Sano, S., Yanase, T., Ohta, T. y Koyama, M.: *Optuna: A next-generation hyperparameter optimization framework*. En *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, páginas 2623–2631, 2019.

[48] Ultralytics: *Yolov8 documentation*, 2023. https://docs.ultralytics.com/, Accessed: 2024-09-13.

[49] Wu, Yuxin, Kirillov, Alexander, Massa, Francisco, Lo, Wan Yen y Girshick, Ross: *Detectron2*, 2019. https://github.com/facebookresearch/detectron2, Accessed: 2024-09-13.

[50] Ortega, Marcelo: *Training on detectron2 with a validation set, and plot loss on it to avoid overfitting*. https://eidos-ai.medium.com/training-on-detectron2-with-a-validation-set-and-plot-loss-on-it-to-avoid-overfi 2020. Accessed: 2024-09-20.

APÉNDICE A

# Desarrollo e Implementación del Software

El repositorio con todo el código utilizado en este estudio.

## A.1. Preparación del Entorno de Trabajo

Para la implementación de este proyecto de Fin de Máster se ha trabajado en dos entornos de trabajo distintos: Google Colab y un entorno local proporcionado por el laboratorio de la universidad. A continuación, se detallan ambos entornos y su relevancia para el desarrollo y entrenamiento de los modelos de detección de objetos.

### A.1.1. Google Colab

Google Colab es una plataforma en línea que proporciona entornos de ejecución de notebooks basados en Jupyter con acceso a recursos de computación en la nube. Su principal ventaja es la capacidad de ejecutar código Python en máquinas virtuales con acceso a GPU y TPU de forma gratuita, facilitando así el desarrollo y entrenamiento de modelos de aprendizaje automático. En el contexto de este proyecto, Google Colab se ha utilizado como una alternativa para experimentar con los modelos de detección de objetos cuando el acceso al hardware local no estaba disponible.

### A.1.2. Hardware del Laboratorio

El entorno local en el laboratorio de la universidad cuenta con un hardware especializado, crucial para el entrenamiento eficiente de modelos de detección de objetos. El entrena-

miento de estos modelos es una tarea intensiva en el uso de GPU, debido a la necesidad de realizar operaciones matemáticas en paralelo con alta velocidad. En este caso, se dispone de una tarjeta gráfica NVIDIA GeForce RTX 3090 Ti con 24 GB de VRAM. Esta GPU proporciona una capacidad de procesamiento paralelo significativa, permitiendo la ejecución rápida de los cálculos necesarios para el entrenamiento de modelos complejos.

Para acceder a este hardware, se han proporcionado las claves necesarias para la conexión SSH y la transmisión de archivos mediante FTP, facilitando así el trabajo con el entorno local.

### A.1.3. Configuración del Entorno Local

Para la configuración del entorno local, se han seguido los siguientes pasos:

#### A.1.3.1. Instalación de Conda

El primer paso en la configuración del entorno local fue la instalación de Conda, una herramienta de gestión de entornos y paquetes que facilita la creación y manejo de entornos virtuales en Python. Para instalar Conda, se descargó el instalador correspondiente desde el sitio web oficial de Anaconda o Miniconda y se siguieron las instrucciones de instalación proporcionadas.

#### A.1.3.2. Creación y Activación del Entorno Virtual

Una vez instalado Conda, se procedió a crear un entorno virtual específico para el proyecto. En este caso, se creó un entorno virtual con Python 3.10, utilizando el siguiente comando en la terminal:

```
conda create --name myenv python=3.10
```

Donde `myenv` es el nombre del entorno virtual. Posteriormente, se activó el entorno con el siguiente comando:

```
conda activate myenv
```

#### A.1.3.3. Instalación de Repositorios y Bibliotecas

Con el entorno virtual activado, se utilizó un script de instalación previamente creado para instalar los repositorios de los modelos de detección y las bibliotecas necesarias. Este script hace uso de `pip`, el gestor de paquetes de Python, para instalar las dependencias requeridas. El script incluye las siguientes instalaciones básicas:

Donde `requirements.txt` es un archivo que lista todas las bibliotecas necesarias para el proyecto, incluyendo los repositorios de YOLO y Detectron2, así como otras dependencias esenciales.

Una vez completada la instalación, el entorno está listo para ejecutar los scripts de entrenamiento, evaluación y despliegue de los modelos de detección de objetos.

### A.1.4. Diferenciación de Entornos y Scripts

A partir de ahora, los scripts utilizados en este proyecto se dividen en dos categorías: los que se ejecutan en el entorno local (laboratorio) y los que se ejecutan en Google Colab. Esta diferenciación es importante ya que, aunque la lógica detrás del código es esencialmente la misma para ambos entornos, los scripts han sido adaptados para aprovechar las características específicas de cada uno.

La mayor inversión de tiempo a sido enfocada en la creación de los scripts utilizados en el entorno local, dado que este ha sido el entorno predominante durante el estudio. Además, se considera redundante discutir ambos entornos en detalle, ya que los principios subyacentes y el funcionamiento del código son equivalentes en ambos casos.

## A.2. Revisión de los *Datasets*

Durante la preparación de los *datasets* para el entrenamiento y evaluación de los modelos de detección de objetos, se empleó una herramienta de etiquetado que permite extraer los datos en diversos formatos, como YOLOv8, Detectron2, entre otros. Sin embargo, es importante destacar que esta herramienta no siempre garantiza una conversión perfecta, ya que pueden ocurrir fallos durante el proceso de etiquetado y conversión.

Para abordar estos problemas y asegurar la consistencia en el formato de los *datasets*, se desarrolló un script específico. Este *script* tiene la capacidad de recibir los *datasets* de YOLOv8 y Detectron2, junto con las imágenes originales que aún no contienen las segmentaciones, y fusionar las anotaciones de ambos modelos. La necesidad de este script surgió al descubrir que, en algunos casos, el *dataset* de Detectron2 puede presentar problemas cuando solo se dispone de una categoría, lo que requiere una modificación en la parte de las categorías para evitar inconsistencias.

El script realiza las siguientes tareas:

- **Carga de Datos:** Lee los archivos de anotaciones de YOLO y Detectron2, junto con las imágenes originales.
- **Corrección de Anotaciones:** Identifica y corrige los errores de etiquetado y las discrepancias entre los formatos.

- **Generación del *Dataset*:** Combina las anotaciones corregidas de ambos modelos en un solo *dataset*, manteniendo la separación de las imágenes en los conjuntos de entrenamiento (*train*), prueba (*test*) y validación (*valid*).

De esta manera, se obtiene un *dataset* único con las imágenes organizadas en los conjuntos mencionados y anotaciones precisas provenientes de ambos modelos. Esto permite una evaluación más rigurosa y equitativa de los modelos de detección de objetos, asegurando que el proceso de entrenamiento y validación sea consistente y fiable.

## A.3. Código de Entrenamiento

Para llevar a cabo el entrenamiento de los modelos de detección de objetos, se desarrollaron dos scripts independientes: uno para YOLOv8 y otro para Detectron2. Ambos scripts fueron diseñados para permitir la automatización del proceso de entrenamiento y evaluación, facilitando la comparación de diferentes configuraciones y pesos preentrenados.

A continuación, se describen los aspectos más relevantes del código de entrenamiento para cada modelo:

### A.3.1. Entrenamiento de YOLOv8

Para el entrenamiento del modelo YOLOv8, el script implementa los siguientes componentes clave:

- **Rangos de Hiperparámetros:** Se definen los rangos para los hiperparámetros que se ajustarán durante el proceso de optimización. Estos rangos son esenciales para guiar la búsqueda de configuraciones óptimas.
- **Decisión del *Dataset* y Pesos Preentrenados:** Se seleccionan el *dataset* adecuado y los pesos preentrenados para el modelo. La elección de estos parámetros influye significativamente en el rendimiento del modelo.
- **Función Objetivo de Optuna:** La función objetivo de Optuna se encarga de la evaluación de los hiperparámetros para cada prueba (trial). Esta función utiliza la métrica de Mean Average Precision al 50 % (*mAP50*) para la validación del modelo.
- **Validación Interna del Modelo:** Se realiza una validación interna del modelo utilizando *mAP50* para obtener los parámetros de retroalimentación necesarios.
- **Registro de Métricas e Hiperparámetros:** Las métricas de rendimiento y los hiperparámetros utilizados en cada prueba se registran en un archivo, lo que permite evaluar y comparar todos los modelos entrenados de manera conjunta.

Se incluirán fragmentos de código relevantes para visualizar los puntos importantes del proceso de entrenamiento, como la configuración de los rangos de hiperparámetros,

la implementación de la función objetivo de Optuna, y el registro de las métricas y parámetros.

### A.3.2. Entrenamiento de Detectron2

El entrenamiento con Detectron2 sigue una lógica similar a la de YOLOv8, aunque presenta algunos pasos adicionales debido a la complejidad de la arquitectura. Estos pasos adicionales incluyen:

- **Registro de Imágenes:** Antes de pasar las imágenes a la función objetivo, es necesario registrar las imágenes de cada conjunto por separado. Este paso asegura que las imágenes se gestionen correctamente para el entrenamiento y la evaluación.
- **Implementación de Funciones Específicas:** A pesar de la similitud en la lógica general, Detectron2 requiere funciones específicas para la configuración y entrenamiento del modelo. Estos detalles son cruciales para la correcta implementación del proceso.

El resto del proceso es bastante similar al de YOLOv8, con diferencias que se centran en las particularidades de la arquitectura de Detectron2. Los scripts para Detectron2 se adaptan a estas diferencias y permiten una comparación equitativa con los modelos entrenados con YOLOv8.

## A.4. Evaluación de los Modelos

Para la evaluación de los modelos, se desarrollaron dos scripts distintos: uno para YOLOv8 y otro para Detectron2. Aunque estos scripts están diseñados para trabajar con diferentes frameworks, su objetivo es obtener resultados consistentes en términos de métricas, figuras y gráficos.

La lógica detrás de la evaluación y la forma en que se calculan las métricas es la misma para ambos scripts, y ha sido detallada en la Sección 4.3.1. Por lo tanto, en esta sección no se repetirá la explicación de la metodología utilizada para la evaluación.

A continuación, se proporciona un resumen de las características de los scripts de evaluación:

- **Script de Evaluación para YOLOv8:** Este script está diseñado para procesar las predicciones generadas por el modelo YOLOv8 y compararlas con las anotaciones de referencia. Se generan las mismas métricas, figuras y gráficos que se obtienen con el script de Detectron2, asegurando una comparación equitativa entre los modelos.
- **Script de Evaluación para Detectron2:** Similar al script de YOLOv8, este script evalúa las predicciones de Detectron2 contra las anotaciones de referencia.

También produce las métricas, figuras y gráficos necesarios para una comparación completa.

Ambos scripts están diseñados para proporcionar una salida coherente y comparable, garantizando que las métricas y visualizaciones sean uniformes entre los diferentes modelos. La sección 4.3.1 ofrece una visión detallada del proceso de evaluación y la lógica empleada para calcular las métricas. Por lo tanto, se hace referencia a esta sección para obtener una comprensión completa del método de evaluación utilizado.