UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería de la Salud

Segmentación y Análisis de Angiografías Coronarias Invasivas a través de la Aplicación de Modelos de Aprendizaje Profundo.

Segmentation and Analysis of Invasive Coronary Angiographies though the Application of Deep Learning Models.

Realizado por

Isabel Nieto Piernagorda

Tutorizado por

Ezequiel López Rubio

Cotutorizado por

Paula Ariadna Jiménez Partinen

Departamento

Lenguajes y Ciencias de la Computación

MÁLAGA, JUNIO DE 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE LA SALUD

Segmentación y Análisis de Angiografías Coronarias Invasivas a través de la Aplicación de Modelos de Aprendizaje Profundo.

Segmentation and Analysis of Invasive Coronary Angiographies through the Application of Deep Learning Models.

Realizado por
Isabel Nieto Piernagorda

Tutorizado por
Ezequiel López Rubio

Cotutorizado por
Paula Ariadna Jiménez Partinen

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2025

Fecha defensa: Julio de 2025

Abstract

Stenosis, or the pathological narrowing of blood vessels, is one of the leading causes of death from myocardial infarction and stroke worldwide. Early and accurate detection of this vascular condition is crucial to ensure successful treatment and minimize the risk of serious complications.

In this Bachelor's Thesis, the ARCADE dataset is used, available in the Zenodo repository. Of its two modalities—"syntax" and "stenosis"—the latter has been selected, encompassing a collection of medical images extracted from DICOM studies in which vascular stenosis is observed, along with their corresponding annotations in JSON format. With the aim of automatically segmenting and analyzing these images, the Python programming language and object detection models from the YOLO (You Only Look Once) family have been employed based on deep learning.

The work begins with a preliminary study for data exploration and preprocessing, highlighting both the importance of medical image processing and the relevance of early detection and diagnosis of stenosis.

Next, the images undergo specific preprocessing, and various YOLO model variants are trained on them, each adapted to identify regions affected by stenosis.

Finally, a comparative evaluation of the different YOLO variants is conducted. This process identifies the optimal parameter configuration, achieving the best balance between diagnostic accuracy and processing speed, thereby contributing to the development of automated tools to support clinicians in the early detection of coronary stenosis.

Keywords: stenosis, image segmentation, deep learning, diagnosis, Python.

Resumen

La estenosis, o estrechamiento patológico de los vasos sanguíneos, constituye una de las principales causas de muerte por infarto de miocardio y accidente cerebrovascular en todo el mundo. La detección precoz y precisa de esta patología vascular resulta determinante para garantizar el éxito del tratamiento y minimizar el riesgo de complicaciones graves.

En el presente Trabajo de Fin de Grado se utiliza el conjunto de datos ARCADE, disponible en el repositorio Zenodo. De entre sus dos modalidades, "syntax" y "stenosis", se ha seleccionado esta última, que agrupa una colección de imágenes médicas extraídas de estudios DICOM en las que se observa estenosis vascular, junto con sus correspondientes anotaciones en formato JSON. Con el objetivo de segmentar y analizar de forma automática dichas imágenes, se ha recurrido al lenguaje de programación Python y a modelos de detección de la familia YOLO (You Only Look Once) basados en aprendizaje profundo.

El trabajo comienza con un estudio preliminar de exploración y preprocesado de los datos, destacando, por un lado, la relevancia del procesamiento de imágenes médicas y, por otro, la importancia de la detección y diagnóstico precoz de la estenosis.

A continuación, las imágenes se someten a un preprocesado y, sobre ellas, se entrenan diversas variantes del modelo YOLO adaptadas a la identificación de regiones con estenosis.

Finalmente, se lleva a cabo una evaluación comparativa de las distintas variantes de YOLO. De este modo, se identifica la configuración óptima de parámetros, que ofrece el mejor equilibrio entre precisión diagnóstica y rapidez de procesamiento, contribuyendo así al desarrollo de herramientas automáticas de apoyo al profesional clínico en la detección precoz de estenosis coronaria.

Palabras clave: estenosis, segmentación de imágenes, aprendizaje profundo, diagnóstico, Python.

Índice

Índice de figuras

Índice de cuadros

1 Introducción

1.1. Contexto

Durante la última década, la comprensión de la fisiopatología de la enfermedad de la arteria coronaria (EAC) ha sufrido una evolución notable [9]. A día de hoy, esta enfermedad está considerada la principal causa de muerte en el mundo [10].

Esta patología se desarrolla cuando los vasos sanguíneos que abastecen de sangre al corazón se vuelven rígidos y se estrechan. Dicha obstrucción se debe a la acumulación de colesterol y otras sustancias en las paredes internas de la arteria, formación conocida como placa. Aunque afecta principalmente a las arterias del corazón, este proceso puede ocurrir en cualquier territorio vascular, dando lugar a bloqueos parciales o totales y favoreciendo la formación de coágulos sanguíneos. Debido a este hecho, pueden producirse enfermedades como infarto de miocardio, insuficiencia cardíaca, accidente cerebrovascular, entre otras complicaciones.

Se ha observado un aumento de personas padecientes de esta enfermedad debido a factores demográficos y genéticos como son el envejecimiento poblacional, los antecedentes familiares, el sexo, etc. No obstante, existen factores adicionales, como hábitos de vida poco saludables, que incrementan la prevalencia de esta enfermedad.

El escenario en que nos encontramos ha creado la necesidad de herramientas de diagnóstico precoz que permitan intervenir de manera rápida y eficaz en el tratamiento de los pacientes.

Para el diagnóstico de la enfermedad arterial coronaria se emplean cada vez con mayor frecuencia distintas técnicas no invasivas. Entre las más relevantes figuran la imagen de perfusión miocárdica, la ecocardiografía de estrés, la angiografía coronaria por tomografía computarizada y la resonancia magnética cardíaca. Estas modalidades aportan información complementaria acerca de la presencia y grado de estenosis coronaria, la extensión de la isquemia y la viabilidad del miocardio, facilitando su diagnóstico.

La coronariografía de rayos X es la prueba estándar para el diagnóstico y tratamiento de la enfermedad de las arterias coronarias [8]. Esta técnica consiste en introducir un catéter en un vaso sanguíneo e insertar contraste para posibilitar la visibilidad de las arterias en las imágenes de rayos X. El diagnóstico generalmente se realiza de forma visual, con lo cual, además de ser compleja su interpretación, esta se puede ver afectada por la subjetividad tanto intraobservador como interobservador. A ello se le suma que este procedimiento suele presentar algunas limitaciones: las imágenes obtenidas suelen presentar ruido, poca iluminación o bajo contraste; además, la presencia del catéter en el campo visual puede interferir con su interpretación, complicando el diagnóstico definitivo.

1.2. Motivación

Como se ha observado anteriormente, esta enfermedad es cada vez más frecuente. Dado que su diagnóstico resulta complejo, surge la necesidad de buscar soluciones que permitan una detección temprana, con el fin de conseguir un tratamiento rápido y eficaz.

Es de vital importancia que los especialistas cuenten con dispositivos sanitarios capaces de conseguir pruebas e imágenes médicas precisas. Es por ello que surge la demanda de nuevos recursos, como son algoritmos que permitan diagnosticar la EAC a través de angiografías coronarias de rayos X (XCA).

Los avances recientes en el aprendizaje profundo han mostrado un progreso considerable en el procesamiento y análisis de imágenes médicas, superando los métodos tradicionales y abriendo nuevas posibilidades para automatizar el análisis coronario cuantitativo (QCA) [8]. Un modelo que detectase la zona de estrechamiento de la lesión sería de gran utilidad tanto para el profesional sanitario, facilitando la evaluación y el abordaje clínico de la enfermedad, como para optimizar el proceso diagnóstico en general. Esta herramienta podría contribuir a superar o mitigar las limitaciones mencionadas anteriormente. Además, su integración en entornos clínicos permitiría mejorar la eficiencia del flujo de trabajo, reducir el tiempo de diagnóstico, factor decisivo en este ámbito, y, por consiguiente, aumentar la tasa de éxito en los tratamientos aplicados.

1.3. Objetivos

Se pretende implementar y evaluar modelos de aprendizaje profundo de la familia YOLO para la detección automática de estenosis coronaria en un conjunto de datos clínicos que representan imágenes estáticas de angiografía coronaria invasiva (ICA) anotadas.

Para ello, en primer lugar, se realizará un preprocesamiento de los datos. A continuación, se implementarán distintas versiones de la familia YOLO para realizar un análisis comparativo entre ambos modelos con el fin de identificar la configuración óptima de parámetros.

Posteriormente se ajustarán los hiperparámetros del modelo escogido para optimizar el rendimiento que permita obtener los mejores resultados en términos de precisión y eficacia diagnóstica.

Este análisis no solo facilitará la selección del modelo más adecuado, sino que también proporcionará una comprensión más profunda sobre el impacto de los parámetros en el rendimiento general del sistema, lo que podría servir de base para futuras mejoras y aplicaciones clínicas.

1.4. Estructura del proyecto

La redacción de este Trabajo Fin de Grado se divide en cinco capítulos. En este primer capítulo se abordan aspectos como la motivación del trabajo, los objetivos planteados, la estructura del proyecto y las tecnologías utilizadas para su desarrollo.

En el capítulo 2, se presenta el uso de la Inteligencia Artificial, una introducción sobre la enfermedad de las arterias coronarias, un estudio del estado del arte actual acerca de los trabajos realizados en este ámbito, la técnica estándar de imagen de angiografía coronaria por rayos X. Además de otras pruebas y técnicas no invasivas fundamentales a día de hoy, con el objetivo de situarse en el contexto del estudio.

A continuación, en el capítulo 3,se definen aspectos técnicos necesarios en los que se fundamenta el trabajo realizado. Para ayudar a ello de forma práctica en el capítulo 4, se expone el desarrollo del proyecto en sí: conceptos básicos, datos utilizados, el proceso seguido para tratarlos, los modelos entrenados y los resultados obtenidos, comparándolos entre sí, además de analizar los parámetros de entrenamiento. Finalmente, en el capítulo 5, se presentan las conclusiones derivadas del trabajo obtenido, así como posibles líneas de desarrollo futuro.

1.5. Tecnologías Utilizadas

1.5.1. Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Destaca por presentar una sintaxis muy legible. Es útil y válida tanto para principiantes como para expertos, ya que es una herramienta potente. Es usada en una gran variedad de campos, como el análisis de datos, la inteligencia artificial, el desarrollo de aplicaciones web o videojuegos. Es uno de los lenguajes más conocidos y utilizados a nivel mundial.

1.5.2. Pytorch

Pytorch es un framework de código abierto desarrollado por Meta (Facebook), basado en Python y en la librería Torch. Se utiliza en el desarrollo de modelos de aprendizaje automático y aprendizaje profundo debido a su flexibilidad, facilidad de uso y capacidad para ejecutar cálculos de alto rendimiento en GPU mediante CUDA. Gracias a su diseño, permite construir modelos de redes neuronales de manera intuitiva, lo que hace que sea una herramienta muy útil.

1.5.3. Google Colaboratory

Google Colaboratory, conocido como Google Colab, es una herramienta gratuita proporcionada por Google que permite escribir y ejecutar código Python en un entorno de cuadernos Jupyter basado en la nube. Una de sus ventajas es que no requiere instalación local de software, ya que todo se ejecuta en servidores remotos. Además, ofrece acceso gratuito a recursos computacionales como GPU y TPU, lo que facilita el procesamiento de tareas que requieren alto rendimiento.

2

Estado del arte

En este capítulo se expone el contexto clínico actual de la enfermedad coronaria, cómo la Inteligencia Artificial (IA) influye en su tratamiento y la importancia de pruebas como la coronariografía y otras técnicas no invasivas utilizadas comúnmente en la actualidad.

2.1. Inteligencia Artificial

La Inteligencia Artificial es un término que se escucha con más frecuencia tanto en el trabajo como en el día a día. Está siendo una potente herramienta que imita el razonamiento humano y que puede ayudar en cualquier tarea, indistintamente del ámbito en el que se encuentre.

2.2. Aprendizaje Automático *(Machine Learning)*

Según Janiesch et al. [11], dentro de la Inteligencia Artificial encontramos el Aprendizaje Automático o *Machine Learning*, donde el objetivo es entrenar al modelo para que reconozca patrones a partir de datos proporcionados con la finalidad de resolver tareas de forma automática. Esto ha hecho que, tareas que anteriormente se realizaban de manera manual, ahora se realicen de manera automática.

Como explica IBM, podemos clasificar el Aprendizaje Automático dependiendo de cómo se ejecuten sus algoritmos en cinco tipos [12]:

- Aprendizaje supervisado: En este método, los modelos se entrenan con datos etiquetados, es decir, el sistema ya conoce las respuestas correctas durante el entrenamiento. El objetivo es que aprenda a predecir resultados futuros a partir de nuevos datos similares. Es útil para tareas como clasificación o regresión.

- Aprendizaje no supervisado: Aquí, los algoritmos trabajan con datos sin etiquetar. Su objetivo es encontrar estructuras o patrones ocultos dentro de la información.
- Aprendizaje semisupervisado: Combina características de los dos anteriores. Utiliza una pequeña cantidad de datos etiquetados junto con una gran cantidad de datos sin etiquetar. Esto permite al modelo mejorar su precisión sin necesitar tantos datos etiquetados.
- Aprendizaje autosupervisado: Es una técnica donde el sistema genera automáticamente etiquetas a partir de los datos, sin intervención humana directa. Es una especie de puente entre el aprendizaje supervisado y el no supervisado.
- Aprendizaje por refuerzo: En este tipo de aprendizaje, el modelo actúa en un entorno y aprende a tomar decisiones mediante prueba y error. Recibe recompensas o penalizaciones en función de sus acciones, y con el tiempo mejora su rendimiento.

En este trabajo se ha utilizado el enfoque de aprendizaje supervisado ya que el conjunto de datos utilizado está etiquetado. El modelo puede aprender directamente la relación entre las entradas y salidas esperadas. Además, como los datos se encuentran de manera estructurada, el aprendizaje supervisado facilitará la evaluación objetiva del rendimiento mediante métricas que se expondrán a lo largo de este capítulo.

2.3. Aprendizaje Profundo *(Deep Learning)*

El Aprendizaje Profundo o *Deep Learning*, es un subconjunto del Aprendizaje Automático, basado en redes neuronales artificiales. Esta herramienta simula el funcionamiento del cerebro humano y permite aprender directamente de grandes volúmenes de datos. A diferencia del Aprendizaje Automático tradicional, donde la extracción de características es manual, en el Aprendizaje Profundo esta extracción se realiza de forma automática. Este enfoque ha obtenido resultados sobresalientes en tareas como la clasificación y segmentación de imágenes médicas.

Para el desarrollo de este trabajo, se han utilizado modelos basados en Aprendizaje Profundo, concretamente en Redes Neuronales Convolucionales *(Convolutional Neural Network, CNN)*, orientados a la segmentación automática de angiografías coronarias. Gracias al uso del aprendizaje profundo en el presente trabajo, se pueden obtener resultados robustos y fiables en un tiempo de análisis adecuado.

2.4. Redes Neuronales

Una red neuronal es un modelo de computación cuya estructura de capas permite la imitación de la transmisión de información del sistema nervioso biológico con capas de nodos conectados. Una red neuronal puede aprender de los datos, de manera que se puede entrenar para que reconozca patrones, clasifique datos y pronostique eventos futuros [1].

Como se observa en la Figura 1, la arquitectura típica de una red neuronal consta de una capa de entrada, capas ocultas y una de salida. Entre ellas están conectadas mediante nodos; cada capa utiliza la salida anterior como entrada.

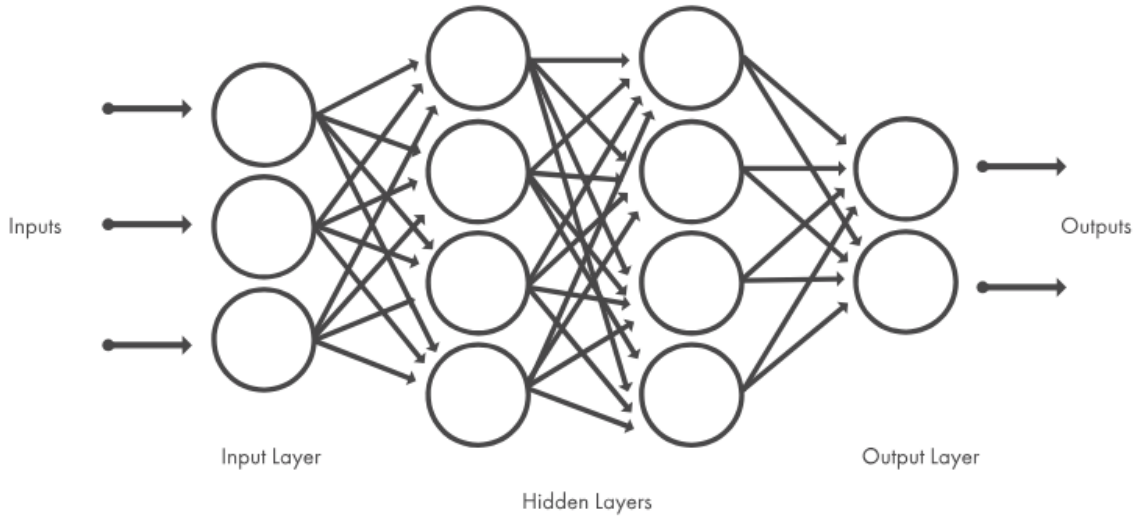

Figura 1: Estructura de una red neuronal [1].

2.4.1. Redes Neuronales Artificiales

Una red neuronal biológica está compuesta por el soma, las dendritas y el axón; las dendritas representan las entradas que recibe una neurona, su función es recibir los impulsos eléctricos de otras neuronas. Esta información es transmitida hasta llegar al soma, que es el núcleo de procesamiento. Con dicha información procesada, es transmitida a través del axón, una extensión larga y delgada que parte del soma, cuya función es llevar la información hacia otras neuronas.

Una red neuronal artificial [2] imita el procedimiento de una red neuronal biológica. De manera análoga, transmite una salida, la cual es creada por la neurona como resultado de la

suma de las entradas multiplicadas por sus pesos. Los pesos son ajustados durante el entrenamiento de la red neuronal artificial. El resultado obtenido se procesa mediante una función de activación, produciendo así un valor que es el que se envía como salida de la neurona. Podemos observar su estructura y la similitud con una neurona biológica en la Figura 2.

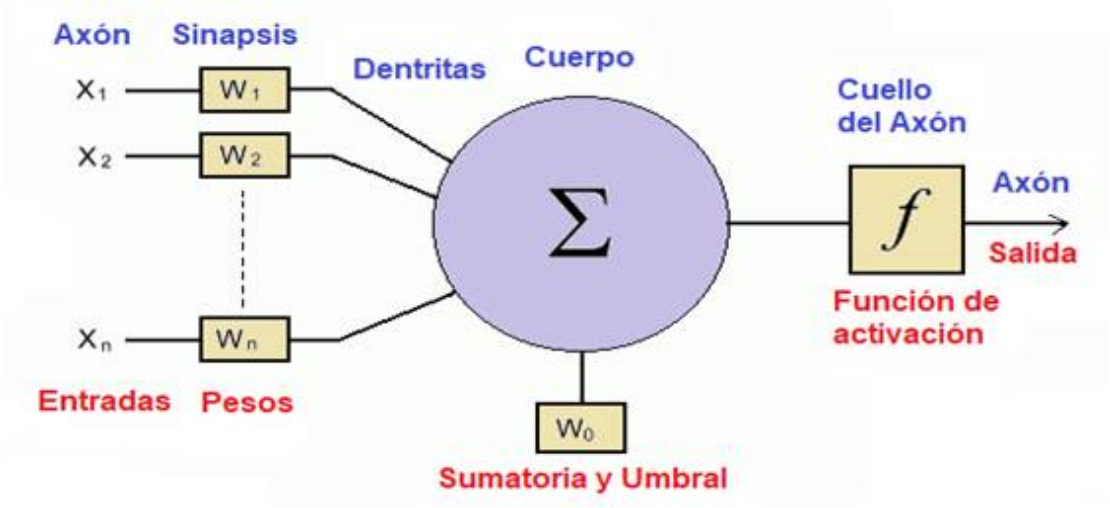

Figura 2: Estructura de una red neuronal artificial y analogía con una red neuronal biológica [2].

2.4.2. Redes Neuronales Convolucionales

Una red neuronal convolucional (CNN) es una arquitectura de red en el campo del *Deep Learning* utilizada principalmente para procesar datos con una estructura similar a una cuadrícula, como son las imágenes. Utiliza operaciones de convolución para extraer características espaciales de los datos, como son patrones o bordes de las imágenes [3].

Estas redes neuronales presentan decenas o cientos de capas, como se puede observar en la Figura 3, donde cada una de ellas está diseñada para identificar distintos aspectos o rasgos de una imagen. A continuación, se describen las capas más comunes en una CNN según el artículo mencionado en el párrafo anterior:

- Capa de entrada: En esta capa se reciben los datos iniciales, como las imágenes, y se preparan para su procesamiento en las siguientes capas. Por ejemplo, se especifica las dimensiones de las imágenes.

- Capa de convolución: Durante el entrenamiento, se aplican filtros a las imágenes con distintos niveles de resolución, y los resultados de cada operación de convolución se utilizan como entrada para la capa siguiente. Estos filtros comienzan detectando elementos muy básicos, como bordes o variaciones de brillo, y progresivamente capturan rasgos más complejos que permiten identificar de manera precisa los objetos representados. Cada uno de los filtros generan un mapa de activación que destaca la presencia de esa característica específica en distintas ubicaciones de la imagen.

- Capa de activación: Esta capa introduce no linealidades en la red, haciendo que se aprendan relaciones complejas. Una función de activación común es la ReLU (Unidad Lineal Rectificada), que reemplaza valores negativos por cero, ayudando a mejorar tanto el entrenamiento como el rendimiento.

- Capas de agrupamiento *pooling*: Su función es reducir la dimensionalidad de los mapas de activación, conservando las características más importantes y disminuyendo la carga computacional. Una técnica muy habitual es el agrupamiento máximo *max pooling*, el cual selecciona el valor máximo en regiones específicas de ese mapa.

- Capas completamente conectadas: En esta capa, cada neurona está conectada a todas las activaciones de la capa anterior. Se utilizan para combinar las características extraídas y realizar tareas como es la clasificación final.

- Capa de salida: En ella, se proporciona la predicción final del modelo, como la probabilidad de pertenencia a diferentes clases. A menudo, se utiliza una función para convertir las salidas en probabilidades que suman uno.

Por otra parte, en el desarrollo de redes neuronales, cabe destacar dos enfoques para entrenar el modelo:

- Desde cero: Es el proceso de construir y entrenar la red neurodal nueva sin utilizar conocimientos o pesos previos. Esto implica definir desde el inicio los parámetros y la arquitectura de la red. Este tipo de entrenamiento hace que se pueda adaptar la red al problema, sin embargo, presenta algunos inconvenientes: requiere más información y tiempo; cuando no se posee grandes volúmenes de datos o recursos suele utilizarse el aprendizaje por transferencia [13].

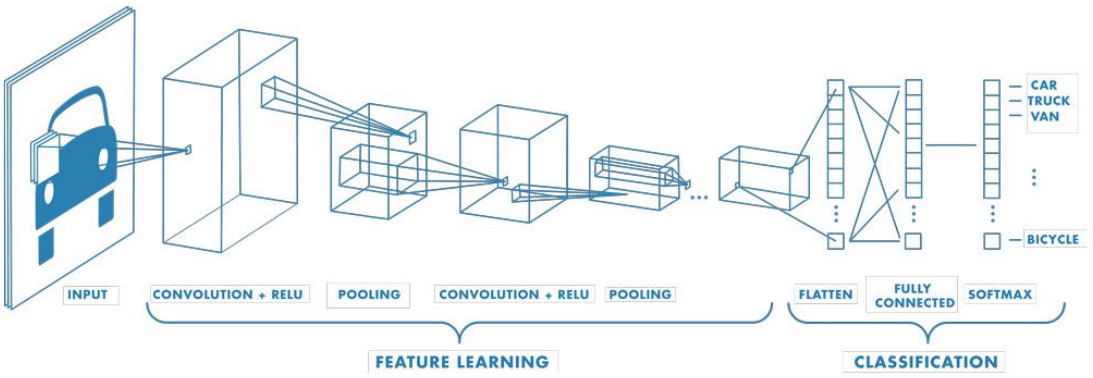

Figura 3: Estructura de una red convolucional con múltiples capas [3].

- Aprendizaje por transferencia *(Transfer learning)*: Es una técnica que consiste en aprovechar el conocimiento adquirido por un modelo entrenado en una tarea ya realizada para aplicarlo en una nueva. No se entrena desde cero, se parte de un modelo preentrenado. En las CNN este enfoque es muy común, donde las primeras capas de las redes aprenden a detectar ciertos patrones generales. Existen dos formas de aplicar el aprendizaje por trasferencia: Uso como extractor de características y ajuste fino. En el primero las capas convolucionales del modelo original se mantienen sin modificar y se utiliza la salida de ellas como entrada para las nuevas capas. En el ajuste fino además de añadir nuevas capas al final, se ajustan también de las capas originales [14].

En el siguiente estudio [15] se compara el rendimiento de las CNNs entrenadas desde cero con las que se utilizan en el aprendizaje por transferencia en el contexto de imágenes médicas. Los resultados obtenidos indican que las redes preentrenadas pueden igualar o superar el rendimiento de las entrenadas desde cero, sobre todo cuando se tienen conjuntos de datos escasos.

2.5. Enfermedad coronaria

La enfermedad de las arterias coronarias es el tipo más común de enfermedad cardíaca. Es la principal causa de muerte entre los hombres y las mujeres en los Estados Unidos [16].

Como se mencionó en el capítulo 1, esta patología se manifiesta por un estrechamiento de las arterias que suministran sangre al corazón; al darse este factor, el flujo sanguíneo no puede circular con normalidad (ver Figura 4), pudiendo incluso interrumpirse, lo que provoca angina de pecho o, en su extremo, un infarto.

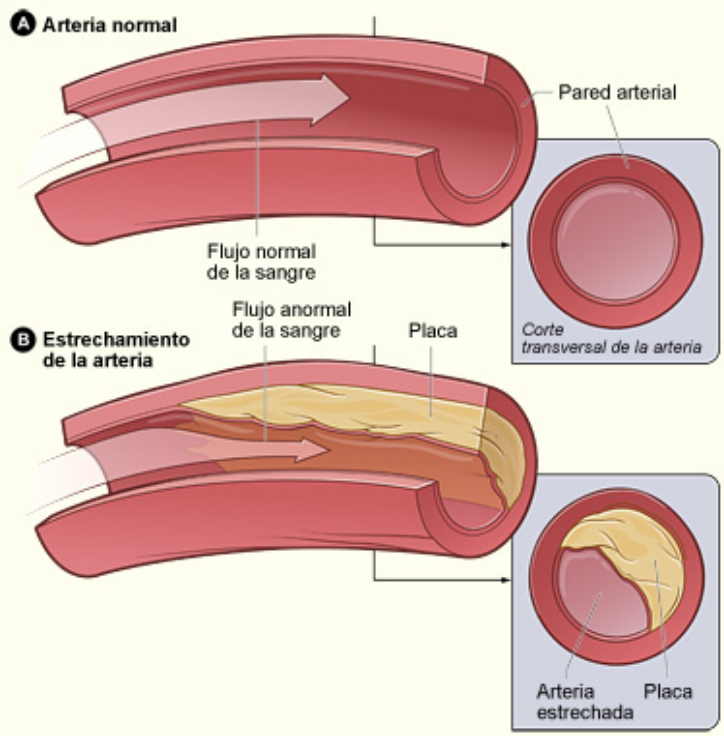

Figura 4: Arteria normal y arteria estrechada por acumulación de placa [4].

Además, esta enfermedad puede desembocar en otras alteraciones, como son insuficiencia cardíaca, donde el músculo principal no puede suministrar sangre al resto del cuerpo, o bien en alteraciones en el propio ritmo del corazón, conocidas como arritmias.

En los países más desarrollados es una de las causas principales de mortalidad. Se estima que cada año la enfermedad cardiovascular causa, en total, unos 4 millones de fallecimientos en Europa y 1,9 millones en la Unión Europea, la mayor parte por enfermedad coronaria, lo que supone un 47 % de todas las muertes en Europa y el 40 % de la Unión Europea [17]. Según el INE [18], en 2023 en España, el 26,5 % del total de muertes se debió a enfermedades cardiovasculares, siendo esta la segunda causa principal de fallecimiento.

En otros países en vías de desarrollo como son África o India, entre otros; la tasa de mortalidad correspondiente es mucho menor. Sin embargo, cambios como el aumento de la esperanza de vida o el sedentarismo harán que la tasa aumente notablemente en años posteriores. Por otro lado, si se compara la situación actual de Estados Unidos con nuestro país, se observa un descenso en el número de casos; este hecho se corresponde principalmente a la prevención y a los tratamientos actuales.

2.6. Angiografía por rayos X

Para el diagnóstico de la enfermedad que se presenta, el profesional médico realiza un examen físico y escucha el latido del corazón para comprobar si presenta alguna anomalía. Por otro lado, se tienen en cuenta los síntomas del paciente, antecedentes familiares, nivel de actividad física en la rutina de la persona; es decir, se recaba información sobre su salud. A continuación, el médico puede realizar las siguientes pruebas complementarias: electrocardiograma, prueba de esfuerzo, análisis de sangre y radiografía de tórax.

Una vez que el médico ha realizado alguna prueba como las mencionadas anteriormente y dichos análisis médicos sugieren que el paciente podría presentar EAC, se lleva a cabo la prueba más conocida: cateterismo cardíaco y angiograma coronario.

En el capítulo anterior 1, se presentó cómo la angiografía invasiva había sido considerada como el estándar de oro para diagnosticar EAC [19].

Esta prueba se realiza en un hospital, se adormece al paciente para que no sienta dolor y se procede a insertar un catéter en un vaso sanguíneo, principalmente, en el brazo, cuello o ingle. Este tubo pasará a las arterias coronarias y es ahí donde se inyecta un tinte. Posteriormente, y con ayuda de una radiografía, se observa el medio de contraste con la finalidad de observar el flujo de la sangre a través de los vasos sanguíneos y el corazón.

Aunque sea la prueba más usada, presenta algunas limitaciones ya que se obtiene la compleja estructura 3D/4D de las arterias coronarias llenas de contraste por proyecciones de rayos X 2D o imágenes siluetas, que pueden ser degradadas por artefactos de imagen, conduciendo a la subestimación de la severidad de las estenosis [20].

En el citado artículo se ha presentado como mejora a esta limitación, algoritmos de reconstrucción 3D/4D del arterial coronario a partir de imágenes de proyección 2D. Este enfoque mantiene la resolución espacial y temporal de la angiografía por rayos X sin necesidad de equipamiento adicional.

Sin embargo, la técnica presenta cuatro desafíos principales: la pérdida de información en el proceso de reconstrucción, la presencia de artefactos, el coste computacional y el incremento de la dosis de contraste. Actualmente se investigan nuevas herramientas clínicas que permitan solventar estas barreras y facilitar su aplicación en un futuro.

2.7. Imágenes médicas

La tecnología avanza diariamente, lo que representa un gran beneficio para el ámbito de la salud. Las imágenes médicas son un claro ejemplo de este hecho, cuya resolución y calidad han mejorado significativamente en los últimos años. Este progreso ha influido de manera significativa en el diagnóstico clínico y, con ello, en el tratamiento de las patologías.

Se pueden obtener las imágenes por métodos muy comunes, como son las técnicas de imagen no invasivas que contribuyen a la prevención al diagnosticar precozmente la EC y evitar las técnicas invasivas; se puede destacar, según [21] y [22]: la ecocardiografía, la tomografía computarizada multidetectores, la resonancia magnética, la cardiología nuclear y la angiotomografía coronaria. En la Figura 5, se pueden observar dos representaciones médicas del corazón y sus arterias coronarias, obtenidas mediante tomografía computarizada (TC).

En el Cuadro 1, se presentan de manera esquemática las ventajas y desventajas de las técnicas no invasivas mencionadas anteriormente.

Una vez que obtenemos las imágenes, mediante estas técnicas es necesario aplicar procedimientos de mejora. Entre ellos se encuentran la reducción de ruido mediante filtros específicos, la normalización del contraste y la segmentación de imágenes para centrarse en una zona determinada. Son algunos de los procedimientos posteriores claves en el desarrollo de herramientas automatizadas de apoyo al diagnóstico.

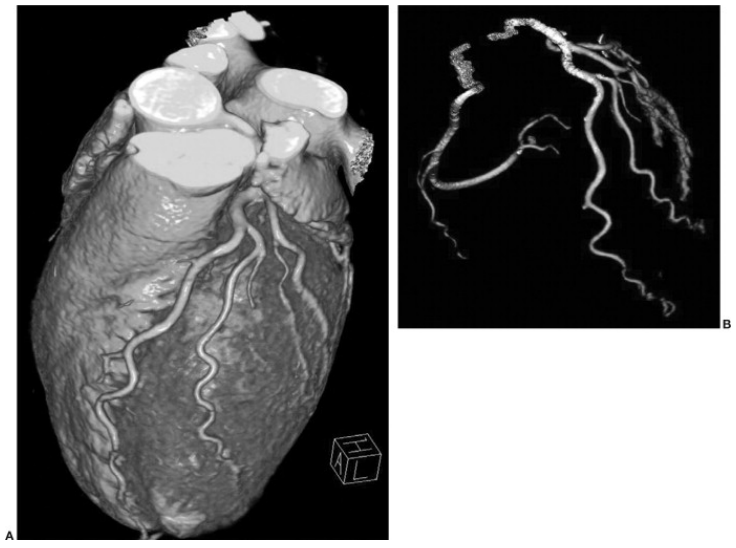

Figura 5: Representación del corazón y las arterias coronarias mediante TC [5].

Cuadro 1: Ventajas y desventajas de las técnicas no invasivas para el estudio de la enfermedad coronaria.

Técnica	Ventajas	Desventajas
Ecocardiografía	▪ Bajo coste y amplia disponibilidad. ▪ Portabilidad para exploración al lado de cama. ▪ Visualización cardíaca en tiempo real.	▪ Resultados dependen de la experiencia del operador. ▪ Limitada por ventanas acústicas.
Tomografía Computarizada Multidetectores (TCMD)	▪ Adquisición rápida de imágenes volumétricas. ▪ Elevada resolución espacial. ▪ Posibilidad de reconstrucciones multiplanares.	▪ Exposición a radiación ionizante. ▪ Requiere medio de contraste vía venosa. ▪ Artefactos por movimiento.
Resonancia Magnética (RM)	▪ Ausencia de radiación ionizante ▪ Excelente contraste tisular multiplanar. ▪ Estudios multiparamétricos.	▪ Coste elevado. ▪ Tiempo de adquisición prolongado. ▪ Contraindicada con ciertos implantes metálicos.
Cardiología Nuclear (CN)	▪ Valoración cuantitativa de la perfusión miocárdica. ▪ Alta sensibilidad diagnóstica.	▪ Uso de radiación. ▪ Baja resolución anatómica. ▪ Coste elevado por material isotópico.
Angiotomografía Coronaria	▪ Alta precisión anatómica. ▪ Evaluación no invasiva del árbol coronario.	▪ Radiación ionizante significativa. ▪ Artefactos por movimiento respiratorio.

2.8. Trabajos previos relacionados

Aunque se han encontrado algunos estudios que aplican técnicas de aprendizaje profundo para el diagnóstico de la EAC, el número de trabajos publicados es escaso, debido a que la cantidad de datos clínicos disponibles es limitada. Esto hace que sea difícil el abordaje de nuevas metodologías.

En el artículo,"Training and validation of a deep learning architecture for the automatic analysis of coronary angiography"[23], se muestra el uso de un modelo de aprendizaje profundo llamado "DeepDiscern", con el objetivo de identificar y medir la angiografía coronaria. Se recopilaron 20,612 angiogramas, de los cuales 13,373 fueron etiquetados para la identificación de 20 segmentos arteriales y 7,239 para la detección de cinco tipos de lesiones: estenosis, oclusión total, calcificación, trombosis y disección. Estas imágenes fueron anotadas por expertos y utilizadas para entrenar y validar la red neuronal.

En los resultados, destacó su precisión 0.98 en la identificación de segmentos de arteria coronaria y F1 de 0.8 en la detección de lesiones, siendo datos a destacar; por otro lado, su velocidad de procesamiento (2 segundos por imagen) lo convierte en una herramienta potencialmente útil para los cardiólogos en tiempo real, mejorando la evaluación de la gravedad y morfología de las lesiones durante intervenciones coronarias.

Otro estudio relevante, realizado por Zhang [24], "Progressive Perception Learning for Main Coronary Segmentation in X-Ray Angiography", se desarrolló un marco de aprendizaje progresivo de percepción (PPL) para solventar tres principales desafíos: la confusión semántica entre vasos principales y colaterales, el bajo contraste entre los vasos y el fondo circundante, y la ambigüedad local entre los bordes vasculares. El modelo PPL contiene módulos de contexto, interferencia y percepción de límites, lo que le permitió lograr segmentaciones más precisas y robustas en angiografías coronarias por rayos X. Gracias a esta arquitectura se obtuvo una alta precisión, Dice 0.95, superando otros métodos en cuanto a exactitud y robustez. Este rendimiento ha hecho que este modelo sea una solución eficaz para el futuro del diagnóstico en la EAC.

En el siguiente artículo, propuesto por Yang et al. [25] se presenta un estudio en que se realizó segmentación automática de vasos coronarios en imágenes de angiografía por rayos X, mediante un modelo de aprendizaje profundo basado en redes convolucionales tipo U-Net.

Uno de los objetivos principales fue automatizar el modelo lo máximo posible con la finalidad de evitar en gran medida la intervención manual entre especialistas y reducir así el tiempo de análisis.

Para ello, se entrenaron y evaluaron diferentes modelos utilizando un conjunto de datos compuesto por 3302 imágenes de angiografías de 2042 pacientes, anotadas por expertos médicos. Entre los modelos probados, el que utilizó DenseNet121 como codificador obtuvo los mejores resultados, alcanzando una puntuación F1 media de 0.917 y demostrando una alta precisión incluso en regiones estrechas causadas por estenosis.

Además, este sistema mostró una gran capacidad de generalización al ser validado con imágenes externas, y destacó por su eficiencia computacional, logrando segmentaciones en tiempo real (aproximadamente 0.04 segundos por imagen). Estos resultados evidencian la viabilidad de aplicar este tipo de redes en contextos clínicos, contribuyendo a mejorar la precisión y rapidez del análisis cuantitativo coronario (QCA).

3

Fundamentos teóricos

En este capítulo se exponen los fundamentos teóricos de las herramientas y métodos que se han llevado a cabo en el desarrollo del trabajo.

3.1. Clasificación, detección y segmentación de imágenes

En el siguiente apartado se definirán tres conceptos claves en imágenes según [26]:

- Clasificación: La clasificación de imágenes es la tarea más básica en visión por computador. Consiste en asignar una única etiqueta o categoría a una imagen completa. Ejemplo: En una foto, un sistema de clasificación únicamente dirá si en esa foto hay un objeto, pero no dirá dónde están esos objetos dentro de la imagen ni cuántos hay. Esta técnica se utiliza en el diagnóstico de enfermedades pulmonares en radiografías y en la clasificación de tumores benignos o malignos.

- Detección: La detección de objetos va un paso más allá que la clasificación. No solo identifica qué objetos hay en la imagen, sino que también indica en qué parte de la imagen se encuentran. Utiliza cuadros delimitadores (rectángulos) que encierran cada objeto detectado. Sin embargo, la detección de objetos solo proporciona una localización aproximada y no los contornos exactos de los objetos. Ejemplo: En una imagen con varios objetos, el sistema marcará cada uno de ellos con un cuadro con su respectiva etiqueta. Esta técnica es utilizada en la detección de nódulos pulmonares en TAC y en la identificación de lesiones cerebrales en resonancia magnética.

- Segmentación: La segmentación de imágenes es una técnica más avanzada y precisa. En lugar de usar cuadros, analiza la imagen a nivel de píxel y asigna una etiqueta a cada

píxel, lo que permite distinguir exactamente la forma y los límites de cada objeto o región. La técnica de segmentación es muy útil para la delimitación de órganos en estudios preoperatorios y la segmentación de tejidos tumorales en imágenes oncológicas. Existen varios tipos de segmentación:

1. Segmentación semántica: Etiqueta cada píxel según la clase a la que pertenece, pero no distingue entre diferentes instancias del mismo objeto.
2. Segmentación de instancias: Además de etiquetar la clase, diferencia cada objeto individualmente, aunque sean de la misma clase.
3. Segmentación panóptica: Combina ambas, identificando tanto la clase de cada píxel como la instancia concreta a la que pertenece

En el Cuadro 2, se exponen las ventajas y desventajas de las técnicas anteriores. La que utilizaremos en el presente estudio será la segmentación, para analizar imágenes de arterias con el objetivo de localizar y delimitar las zonas donde se presenta un estrechamiento vascular. Esta aproximación es esencial para facilitar una evaluación detallada y cuantitativa de la patología vascular en cuestión. En la Figura 6 podemos ver las diferencias de estas técnicas de manera visual.

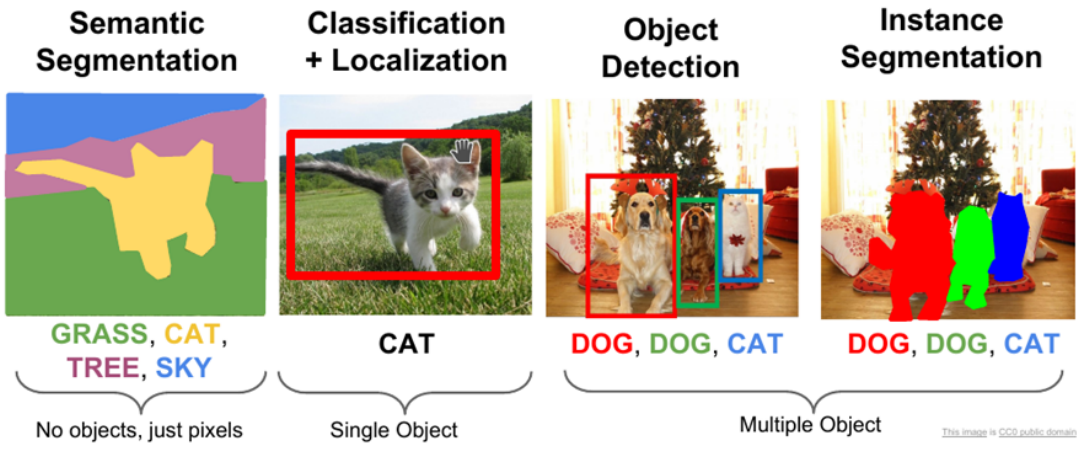

Figura 6: Diferencias en una imagen con las distintas técnicas [6].

Cuadro 2: Comparación de las ventajas y desventajas de las tres técnicas en imágenes médicas.

Técnica	Ventajas	Desventajas
Clasificación	▪ Alta precisión en el diagnóstico automático. ▪ Identifica patrones invisibles al ojo humano. ▪ Ahorro de tiempo en screening masivo.	▪ Grandes volúmenes de datos etiquetados. ▪ Difícil detectar patologías a la vez. ▪ No produce localización espacial.
Detección	▪ Localización de regiones de interés. ▪ Mejora la interpretación clínica. ▪ Útil en sistemas CAD.	▪ Posibilidad de generar falsos positivos. ▪ Requiere entrenamiento con anotaciones precisas. ▪ Menos resolución que la segmentación.
Segmentación	▪ Precisión en los bordes de estructuras anatómicas. ▪ Permite el análisis cuantitativo. ▪ Mejora la planificación quirúrgica.	▪ Alta complejidad computacional. ▪ Dependiente de la calidad del modelo.

3.2. Familia YOLO

YOLO (You Only Look Once) es un popular modelo de detección de objetos y segmentación de imágenes en tiempo real, desarrollado por Joseph Redmon en 2015. YOLO ganó popularidad por su alta velocidad y precisión [27].

Este modelo divide la imagen en una cuadrícula y, para cada celda, predice las cajas delimitadoras (bounding boxes) junto con la probabilidad de que contenga el objeto de una clase determinada, lo que hace que el procesamiento sea mucho más rápido [28].Existen múltiples versiones del modelo; en este trabajo se empleará YOLOv8 y YOLOv11 para la segmentación

de instancias a través de la implementación de Ultralytics, que facilita su uso mediante Python.

En el capítulo 4, se presenta el desarrollo del modelo de segmentación de instancias aplicado a imágenes de angiografías coronarias con el objetivo de identificar con precisión las zonas de obstrucción en los vasos coronarios. El resultado obtenido es la caja delimitadora, la región coloreada que representa la forma segmentada del vaso y un valor numérico que estima el grado de estenosis en ese segmento. Esta información permite no solo visualizar las zonas afectadas, sino también cuantificar la severidad de la obstrucción, lo cual es fundamental para apoyar el diagnóstico clínico de enfermedades coronarias. En la Figura 7 podemos observar un ejemplo visual del *framework* de YOLO.

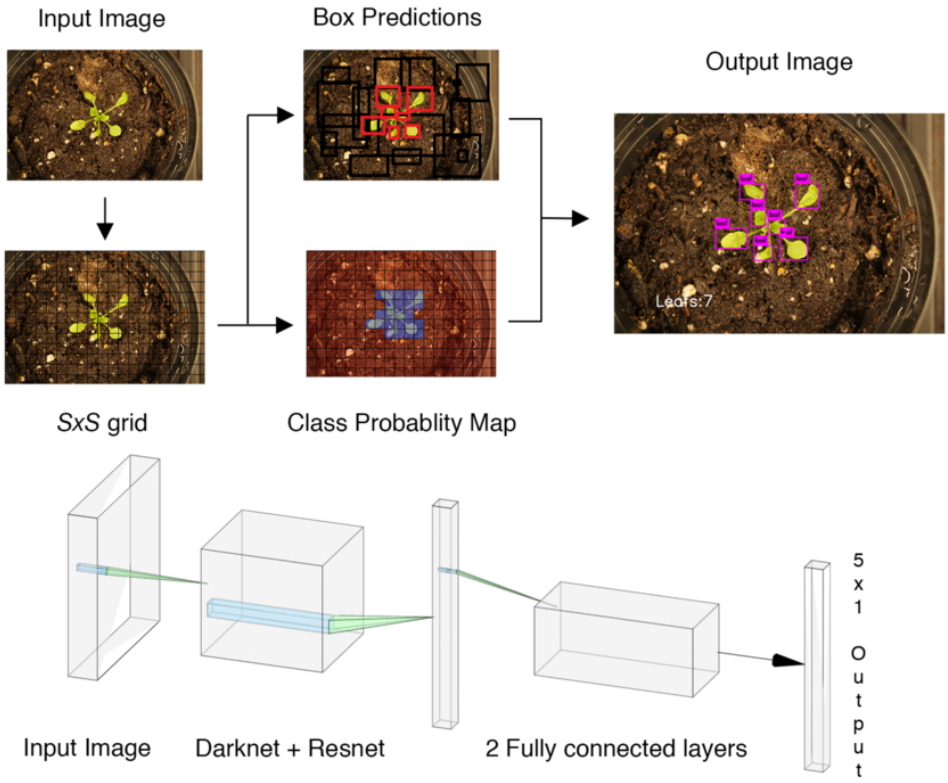

Figura 7: Ejemplo *framework* de YOLO [7].

3.3. Modelos implementados

Para realizar el entrenamiento del modelo se ha optado por utilizar la arquitectura YOLOv8 debido a su equilibrio entre precisión, velocidad y soporte para la tarea de segmentación, además de la facilidad de uso que proporciona su interfaz en Python. Por otro lado, esta versión,

basada en PyTorch, proporciona distintos tamaños de modelos, donde cada uno de ellos prioriza velocidad, precisión o eficiencia computacional. En el Cuadro 3 se puede observar una comparación entre los distintos modelos de esta versión. En este trabajo se ha seleccionado para el entrenamiento el modelo **YOLOv8n (Nano)**. Aunque existen otras variantes con mayor precisión, esta destaca por su gran velocidad de entrenamiento, bajo consumo de recursos y capacidad para ofrecer resultados en tiempo real, lo que hace que sea una opción adecuada para el contexto en que se encuentra el presente estudio.

Posteriormente, con el objetivo de realizar una comparativa entre modelos, se ha utilizado el modelo **YOLOv8x (Extra-large)**, que corresponde a la variante utilizada en el *paper* de referencia del conjunto de datos. Esta versión presenta una gran precisión; sin embargo, la velocidad es menor y el consumo de recursos es elevado. Finalmente, se ha implementado la arquitectura YOLOv11, la versión más reciente de la serie YOLO desarrollada por Ultralytics, también con el modelo nano. El propósito es analizar la comparativa de versiones y ver cómo ello afecta a las métricas de rendimiento.

En la Figura 8 se puede visualizar un ejemplo de cómo se ha llevado a cabo el entrenamiento, importando el modelo preentrenado YOLOv8n usando el método "train" para realizar la afinación. En este caso se utilizan como parámetros 100 "epochs" (épocas) y 16 "batch size" (tamaño del lote). En cada iteración de entrenamiento se almacenarán los pesos del modelo afinado.

```python
from ultralytics import YOLO

model8 = YOLO("yolov8n-seg.pt")

results8 = model8.train(
    data   = "/content/datasets/arcadefinal/arcade/stenosis_YOLO/data.yaml",
    epochs = 100,
    batch  = 16,
    imgsz  = 640,
    plots  = True
)
```

Figura 8: Ejemplo de entrenamiento con YOLOv8

Cuadro 3: Comparación de las variantes del modelo YOLOv8

Modelo	Tamaño	Velocidad	Precisión
YOLOv8n (Nano)	▪ Muy pequeño. ▪ Ideal con pocos recursos.	▪ Muy alta. ▪ Adecuado para tiempo real.	▪ Moderada. ▪ Tareas generales.
YOLOv8s (Small)	▪ Pequeño.	▪ Alta. ▪ Rápido en GPU estándar.	▪ Buena. ▪ Apta para la mayoría de aplicaciones.
YOLOv8m (Medium)	▪ Medio. ▪ Requiere más memoria.	▪ Media. ▪ Menor que versiones pequeñas.	▪ Alta.
YOLOv8l (Large)	▪ Grande. ▪ Necesita GPU con buena capacidad.	▪ Media-baja. ▪ Más lento pero estable.	▪ Muy alta. ▪ Recomendado para mayor precisión.
YOLOv8x (Extra-Large)	▪ Muy grande. ▪ Elevado consumo de recursos.	▪ Baja. ▪ Menos apto para tiempo real.	▪ Máxima. ▪ Ideal para tareas críticas.

3.4. Evaluación de los modelos: métricas de rendimiento

Con el objetivo de medir el rendimiento del modelo que se ha utilizado, es esencial utilizar métricas que permitan evaluar tanto la precisión en la localización de las estructuras como la calidad de la segmentación.

- *True Positive (TP)*: Valor real positivo y el modelo predice positivo.
- *False Positive (FP)*: Valor real negativo pero el modelo predice positivo.

- *True Negative (TN)*: Valor real negativo y el modelo predice negativo.

- *False Negative (FN)*: Valor real positivo y el modelo predice negativo.

- *Precision* (Precisión): Este parámetro mide la proporción de verdaderos positivos (VP) respecto al total de predicciones positivas realizadas por el modelo (verdaderos positivos más falsos positivos, FP). Esta métrica indica qué tan fiables son las predicciones del modelo, es decir, la probabilidad de que un objeto detectado realmente corresponda a un vaso coronario segmentado correctamente. Se calcula como:

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

- *Recall* (Sensibilidad): Esta métrica mide la proporción de verdaderos positivos detectados respecto al total de objetos reales (verdaderos positivos más falsos negativos, FN). Este parámetro refleja la capacidad del modelo para detectar todas las áreas correctas. Se calcula como:

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

- *Intersection over Union* (IoU): Es una métrica clave en segmentación que evalúa el solapamiento entre la máscara predicha y la máscara real. Se define como la relación entre la intersección y la unión de ambas máscaras:

$$\text{IoU} = \frac{|\text{Máscara Predicha} \cap \text{Máscara Real}|}{|\text{Máscara Predicha} \cup \text{Máscara Real}|}$$

- F1 Score (o DICE): Combina la precisión y la sensibilidad (recall) en una sola medida. Mide la similitud entre la segmentación predicha y la real. Su fórmula es la siguiente:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- *mean Average Precision* (mAP): Combina precisión y *recall* en distintos umbrales de confianza y de IoU para ofrecer una evaluación global del desempeño del modelo. El mAP se calcula como el promedio del *Average Precision* (AP) para todas las clases y umbrales, donde el AP es el área bajo la curva *Precision-Recall*.

3.4.1. Parámetros del modelo

Por otro lado, hay otra serie de parámetros que se tienen en cuenta a la hora de la evaluación y que son de vital importancia:

- Umbral de confianza: Establece el límite mínimo para considerar una predicción válida.
- Umbral de IoU: Determina el nivel mínimo de superposición requerido para considerar que una predicción coincide con la realidad.
- Tamaño de imagen de entrada: Influye en la calidad y resolución de la segmentación.
- Batch size (tamaño de lote): Corresponde al número de muestras que se procesan antes de actualizar los pesos del modelo. Influye en la velocidad y estabilidad del entrenamiento.
- Tasa de aprendizaje (learning rate): Controla la velocidad a la que el modelo actualiza sus pesos durante el entrenamiento. Una tasa demasiado alta puede causar que el modelo no converja, mientras que una muy baja puede hacer que el entrenamiento sea muy lento.
- Número de épocas (epochs): Cantidad de veces que el modelo procesa todo el conjunto de datos durante el entrenamiento. Más épocas permiten al modelo aprender mejor, pero un exceso puede llevar a sobreajuste (overfitting).

3.4.2. Hiperparámetros

Los hiperparámetros son ajustes de alto nivel del algoritmo que se establecen antes de la fase de entrenamiento y permanecen constantes durante la misma. Algunos de estos valores más comunes en YOLO son:

- Tamaño de lote (*batch size*): Indica el número de imágenes procesadas simultáneamente en cada iteración del entrenamiento. En los experimentos anteriores, este valor fue modificado (de 8 a 16) y se observó una mejora del rendimiento con el aumento del tamaño del lote.
- Número de épocas (*epochs*): Una época se define como una pasada completa por todos los ejemplos del conjunto de entrenamiento. De forma análoga al tamaño de lote, este

parámetro fue ajustado en los entrenamientos anteriores y se ha comprobado que al incrementar el número de épocas (de 50 a 100), el modelo obtenía un rendimiento superior, al disponer de más iteraciones para ajustar sus pesos.

- Tasa de aprendizaje (*learning rate*): Este hiperparámetro es uno de los más influyentes en el entrenamiento de redes neuronales profundas. Controla el tamaño de los paso que da el optimizador al actualizar los pesos del modelo. Si la tasa de aprendizaje es muy alta puede provocar que el modelo no converja, si por el contrario, la tasa de aprendizaje es muy baja, puede ralentizar el proceso haciendo que el modelo no alcance una buena solución. En YOLO este parámetro se define como "lr0" y su valor por defecto es 0.01. Para evaluar su impacto en el rendimiento del modelo seleccionado, en esta sección se propone realizar experimentos con distintos valores con el fin de observar como influye en la precisión y estabilidad del modelo.

- *Momentum:* Es un hiperparámetro utilizado en los algoritmos de optimización que ayuda a acelerar el entrenamiento y a suavizar las actualizaciones de los pesos.

- *Weight decay:* Hiperparámetro de regularización L2, que penaliza los pesos grandes para evitar el sobreajuste.

- Arquitectura específica: Se refiere a parámetros estructurales que definen como es la forma del modelo, como el número de capas, el número de canales de la capa, los tipos de funciones de activación utilizadas, entre otros. Estos parámetros vienen por defecto, sin embargo, también pueden ajustarse manualmente si se desea realizar una optimización más profunda. Estos elementos determinan la capacidad del modelo para extraer y procesar características complejas, y pueden tener un impacto significativo en su precisión y capacidad de generalización.

Los parámetros, métricas e hiperparámetros descritos en este apartado serán fundamentales para el análisis y evaluación del modelo en el siguiente capítulo, donde se detallará el proceso de entrenamiento, ajuste y validación de la segmentación en las imágenes de angiografías coronarias, con el fin de analizar los resultados y extraer conclusiones.

4
Desarrollo

4.1. Exploración de los datos

El conjunto de datos propuesto para este trabajo [8], está compuesto por 1500 imágenes de coronariografía por rayos X, etiquetadas por expertos médicos con el objetivo de localizar lesiones arteriales. Dichas imágenes pertenecen a pacientes con sospecha de padecer EAC. En total, se incluyen 1500 pacientes con una edad media de 45 años. Del total, el 57 % son hombres y el 43 % mujeres.

Las imágenes fueron adquiridas mediante dos angiógrafos diferentes bajo un protocolo estandarizado de adquisición. Posteriormente, fueron almacenadas en formato DICOM con una resolución de 512x512 píxeles. El conjunto de datos se construyó a partir de vídeos XCA grabados desde seis ángulos distintos, seleccionando hasta 12 imágenes por paciente según criterios de contraste, calidad y relevancia clínica. Finalmente, para este estudio se cuenta con 1500 imágenes repartidas en conjuntos de entrenamiento (1000), validación (200) y prueba (300).

Dentro de cada uno de los tres conjuntos anteriores, se encuentran las imágenes en formato.png correspondientes junto a sus anotaciones en formato .JSON denominados según el conjunto perteneciente: train.JSON, val.JSON y test.JSON. Este archivo .JSON contiene tres campos: categorías, imágenes y anotaciones.

- Categorías: Presenta un identificador único (del 1 al 26) y un nombre asociado.
- Imágenes: Incluye el identificador de cada imagen (id), sus dimensiones en píxeles (“width, height”) y el nombre del archivo (“file_name”).
- Anotaciones: Abarca un identificador único de la anotación (id), el identificador de la imagen (“image_id”) y el identificador de la categoría a la que corresponde (“catego-

ria_id"). Además cuenta con un campo ("segmentation") con las coordenadas del contorno de la máscara en el formato de XYXY. Otro campo adicional ("bbox") con las coordenadas de la caja delimitadora en formato XYWH y finalmente un último campo ("area") con el área total de la caja delimitadora.

En la Figura 9, se puede observar un ejemplo de un frame de una coronariografía perteneciente a nuestros datos. Y a la derecha, se puede ver esa misma imagen una vez que se ha realizado el proceso de segmentación.

Figura 9: Ejemplo de frame de una coronariografía de los datos y el resultado de la misma una vez se aplica la segmentación [8].

4.2. Procesado de los datos

Con el objetivo de llevar a cabo la tarea de segmentación, fue necesario preparar los datos de entrada a un formato que fuera adecuado antes de realizar el entrenamiento con la arquitectura YOLO. Los datos con los que se ha trabajado estaban en formato COCO *(Common Objects in Context)*, frecuentemente utilizado en el ámbito de la visión por computador.

Tal y como se describió en el apartado anterior, se ve el desglose de este formato, en el que encontramos una estructura estandarizada para la anotación de imágenes, incluyendo información sobre la imagen, como su categoría, cajas delimitadoras *(bounding boxes)* y máscaras de segmentación.

Para lograr el formato conveniente, fue necesario realizar la transformación de estas anotaciones, manteniendo la estructura YOLO adaptada a la tarea de segmentación:

1. Carga del archivo JSON correspondiente a cada parte del conjunto de datos (train, val y test) que contiene las anotaciones en formato COCO.

2. Creación de un diccionario para buscar el nombre del archivo de la imagen a partir del "imagen id". Se necesita el nombre para crear los archivos .txt correspondientes.

3. Agrupación de las anotaciones por imagen, ya que en COCO cada anotación es una entrada independiente y YOLO necesita las anotaciones juntas por imagen.

4. Normalización de las coordenadas. Esta tarea es esencial para que el modelo sea más robusto y preciso. En COCO en segmentación las coordenadas están en píxeles absolutos, se necesita normalizarlos entre 0 y 1 dividiendo por el ancho o alto de la imagen.

5. Asignacion de clase, todas las instancias pertenecen a una caategoría ("estenosis"), por lo que se asigna un identificador de clase ("class_id=0") en todas las anotaciones.

6. Normalización de las coordenadas del *bounding box*.

7. Generación de archivos .txt. para cada imagen con todas sus anotaciones en el formato requerido.

Este proceso ofrece ciertas ventajas. En primer lugar, la normalización va a mejorar la capacidad del modelo para generalizar, ya que las redes neuronales profundas trabajan mejor con valores pequeños y en un rango consistente. Además, al trabajar con coordenadas relativas, se logra independencia respecto al tamaño de las imágenes, porque todas las coordenadas están entre 0 y 1. Finalmente, el uso de archivos .txt individuales por imagen es compatible con los requisitos del formato YOLO.

Una vez se tienen los datos en la forma correcta, se genera el archivo data.yaml. Este archivo en formato YAML se utiliza por la librería Ultralytics YOLO para poder definir los parámetros básicos del conjunto de datos. En dicho archivo se encuentra:

- Las rutas de los subconjuntos de datos de entrenamiento (train), validación (val) y prueba (test).

- El número de clases (en este caso solo hay 1)
- El nombre de la clase (“stenosis”)

Este archivo va a permitir a YOLO localizar las imágenes y sus etiquetas correspondientes, así como interpretar de manera correcta las clases durante el proceso de entrenamiento. El siguiente paso es el entrenamiento. En él se ajustan los pesos del modelo a los datos específicos del problema presente, se realiza mediante un bucle iterativo (épocas) donde el modelo va aprendiendo a minimizar el error entre sus predicciones y las etiquetas reales del dataset. Posteriormente, se generan predicciones sobre los datos con el modelo ya afinado. Una vez se tenga esto, se procede a calcular las métricas de rendimiento comparando la imagen original con la imagen del entrenamiento. A continuación, se presentan los entrenamientos con los modelos explicados en 3.

4.3. Entrenamiento del modelo

En este apartado se detallan los diferentes experimentos realizados durante el desarrollo del modelo, incluyendo las configuraciones de entrenamiento, los ajustes de hiperparámetros y los resultados obtenidos en cada caso. Se presentan también las métricas utilizadas para evaluar el rendimiento del modelo, así como las comparaciones entre las distintas versiones entrenadas.

4.3.1. Arquitectura YOLOv8 y modelo Nano (50 *epochs* y número variado de *batch size*)

En primer lugar, se realizó un entrenamiento con la versión menos reciente YOLOv8 y con el modelo más pequeño, el modelo Nano.

- Modelo: “yolov8n-seg.pt”
- Número de épocas: 50
- Tamaño del lote: 8 y 16

En la Tabla 4 se recogen los resultados obtenidos: el primer entrenamiento se ha realizado con 50 épocas y un tamaño de lote de 8, mientras que en el segundo se ha cambiado el tamaño de lote a 16; se puede observar en la primera y segunda columna, respectivamente.

Cuadro 4: Comparativa de métricas globales con YOLOv8n con 50 iteraciones y dos configuraciones de tamaño de lote 8 y 16

Parámetros	**Iteraciones=50 / BS=8**	**Iteraciones=50/ BS=16**
Métricas globales		
True Positives (TP)	260911	250034
False Positives (FP)	499407	416205
False Negatives (FN)	506926	517803
True Negatives (TN)	77375956	77459158
IoU	0.2059	0.2112
Precisión	0.3432	0.3753
Recall	0.3398	0.3256
Dice coefficient	0.3415	0.3487
Métricas individuales (media $\pm$ desviación estándar)		
IoU	0.2240 ± 0.1689	0.2382 ± 0.1932
Precisión	0.3949 ± 0.3055	0.3935 ± 0.3144
Recall	0.3993 ± 0.2940	0.4005 ± 0.3112
Dice coefficient	0.3355 ± 0.2237	0.3460 ± 0.2513

4.3.2. Arquitectura YOLOv8 y modelo Nano (100 *epochs* y número variado de *batch size*)

En este entrenamiento, se mantiene la versión y el modelo; sin embargo, el número de épocas se ha aumentado a 100 respecto al anterior, que era 50, y el tamaño de lote ha sido 8 en la primera prueba; posteriormente, se ha realizado con 16.

- Modelo: "yolov8n-seg.pt"
- Número de épocas: 100
- Tamaño del lote: 8 y 16

En la Tabla 5 se recogen los resultados obtenidos.

Cuadro 5: Comparativa de métricas globales con YOLOv8n con 100 iteraciones y dos configuraciones de tamaño de lote 8 y 16

Parámetros	**Iteraciones=100 / BS=8**	**Iteraciones=100 / BS=16**
Métricas globales		
True Positives (TP)	335598	317883
False Positives (FP)	651231	609913
False Negatives (FN)	432239	449954
True Negatives (TN)	77224132	77265450
IoU	0.2365	0.2307
Precisión	0.3401	0.3426
Recall	0.4371	0.4140
Dice coefficient	0.3825	0.3749
Métricas individuales (media $\pm$ desviación estándar)		
IoU	0.2625 ± 0.1770	0.2496 ± 0.1684
Precisión	0.3951 ± 0.2793	0.3821 ± 0.2724
Recall	0.4929 ± 0.2941	0.4700 ± 0.2887
Dice coefficient	0.3850 ± 0.2234	0.3705 ± 0.2171

4.3.3. Arquitectura YOLOv8 y modelo *Extra-Large* (100 *epochs* y 16 *batch size*)

Este modelo ha sido el propuesto en el *paper* del conjunto de datos, por lo que se usarán los mismos parámetros y posteriormente se compararán con los resultados obtenidos en este trabajo.

- Modelo: "yolov8x-seg.pt"
- Número de épocas: 100
- Tamaño del lote: 16

En la Tabla 6 se recogen los resultados obtenidos en la primera columna y, en la segunda columna, se han añadido métricas del *paper* original del conjunto de datos. Únicamente se han

incluido aquellas métricas que fueron explícitamente indicadas en dicho trabajo (Precisión, *Recall* y *Dice coefficient*), por lo que el resto de celdas aparecen marcadas como "–" al no estar disponibles en la publicación original.

Cuadro 6: Comparativa de métricas globales con YOLOv8x y el *paper* del conjunto de datos con 100 iteraciones y tamaño de lote 16

Parámetros	**Iteraciones=100 / BS=16**	**Iteraciones=100 / BS=16**
	Métricas globales	**Métricas del paper**
True Positives (TP)	325238	–
False Positives (FP)	613147	–
False Negatives (FN)	442599	–
True Negatives (TN)	77262216	–
IoU	0.2355	–
Precisión	0.3466	0.33
Recall	0.4236	0.45
Dice coefficient	0.3812	0.38

4.3.4. Arquitectura YOLOv11 y modelo Nano (50 *epochs* y número variado de *batch size*)

Una vez se han realizado los entrenamientos con los modelos seleccionados, se procede a su repetición cambiando la versión de YOLOv8 a YOLOv11.

- Modelo: "yolov11n-seg.pt"
- Número de épocas: 50
- Tamaño del lote: 8 y 16

De la misma manera, en la Tabla 7 se recogen los resultados obtenidos con el modelo Nano en esta versión: el primer entrenamiento se ha realizado con 50 épocas y un tamaño de lote de 8, mientras que en el segundo se ha cambiado el tamaño de lote a 16; se puede observar en la primera y segunda columna, respectivamente.

Cuadro 7: Comparativa de métricas globales con YOLOv11n con 50 iteraciones y dos configuraciones de tamaño de lote 8 y 16

Parámetros	**Iteraciones=50 / BS=8**	**Iteraciones=50/ BS=16**
Métricas globales		
True Positives (TP)	222595	250365
False Positives (FP)	282933	407723
False Negatives (FN)	545242	517472
True Negatives (TN)	77592430	77467640
IoU	0.2118	0.2130
Precisión	0.4403	0.3804
Recall	0.2899	0.3261
Dice coefficient	0.3496	0.3512
Métricas individuales (media $\pm$ desviación estándar)		
IoU	0.2334 ± 0.1927	0.2261 ± 0.1726
Precisión	0.4212 ± 0.3352	0.3950 ± 0.3090
Recall	0.3595 ± 0.3036	0.3889 ± 0.2972
Dice coefficient	0.3392 ± 0.2538	0.3368 ± 0.2300

4.3.5. Arquitectura YOLOv11 y modelo Nano (100 *epochs* y número variado de *batch size*)

En este apartado se implementa el mismo modelo que en el apartado anterior con el aumento del número de épocas, 100; se varía el tamaño del lote y se recogen los resultados.

- Modelo: "yolov11n-seg.pt"
- Número de épocas: 100
- Tamaño del lote: 8 y 16

En la Tabla 8 se recogen los resultados obtenidos con el modelo Nano en esta versión: el primer entrenamiento se ha realizado con 100 épocas y un tamaño de lote de 8, mientras que en

el segundo se ha cambiado el tamaño de lote a 16; se puede observar en la primera y segunda columna, respectivamente.

Cuadro 8: Comparativa de métricas globales con YOLOv11n con 100 iteraciones y dos configuraciones de tamaño de lote 8 y 16

Parámetros	**Iteraciones=100 / BS=8**	**Iteraciones=100/ BS=16**
Métricas globales		
True Positives (TP)	310944	337874
False Positives (FP)	557300	572718
False Negatives (FN)	456893	429963
True Negatives (TN)	77318063	77302645
IoU	0.2347	0.2520
Precisión	0.3581	0.3710
Recall	0.4050	0.4400
Dice coefficient	0.3801	0.4026
Métricas individuales (media $\pm$ desviación estándar)		
IoU	0.2566 ± 0.1761	0.2710 ± 0.1636
Precisión	0.3989 ± 0.2803	0.4201 ± 0.2595
Recall	0.4697 ± 0.2973	0.4901 ± 0.2723
Dice coefficient	0.3773 ± 0.2251	0.4002 ± 0.2057

4.4. Análisis y comparación entre los modelos

En este apartado se analizan y comparan los resultados obtenidos (métricas globales y métricas individuales) a partir de los entrenamientos realizados, tanto entre las versiones de los modelos (YOLOv8 y YOLOv11), como entre las diferentes configuraciones de entrenamiento que se han aplicado en ellos. A partir de los valores recogidos en las tablas de la sección anterior, se han elaborado dos representaciones gráficas que permiten visualizar de forma más clara la evolución de cada modelo.

En la Figura 10 y en la Figura 11, se muestran los resultados de las métricas globales tras el entrenamiento con distintas configuraciones de épocas y de tamaño de lote. Las métricas consideradas fueron IoU, Precisión, *Recall* y el coeficiente *Dice*, valores que permiten evaluar el rendimiento de segmentación del modelo.

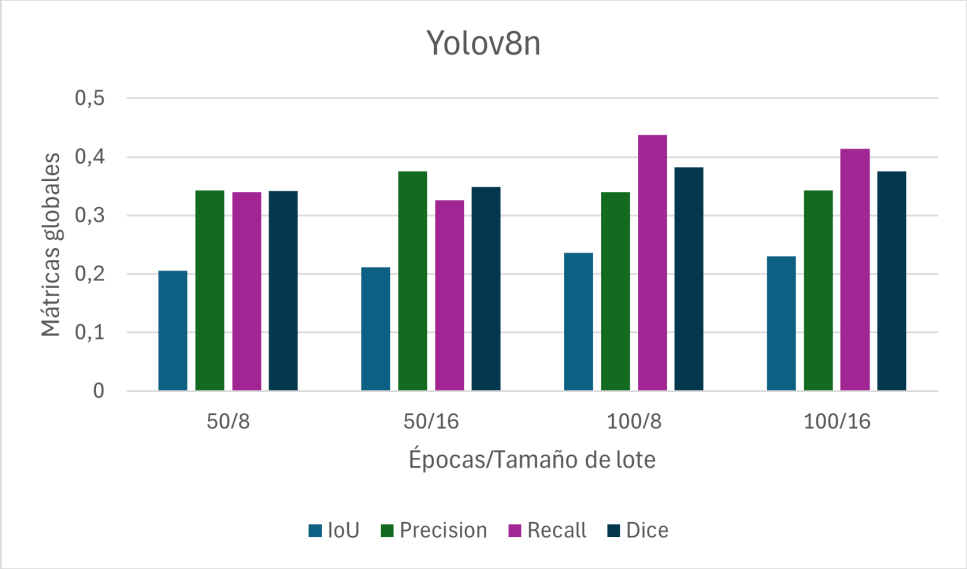

Figura 10: Gráfica de los resultados de los modelos implementados en YOLOv8n.

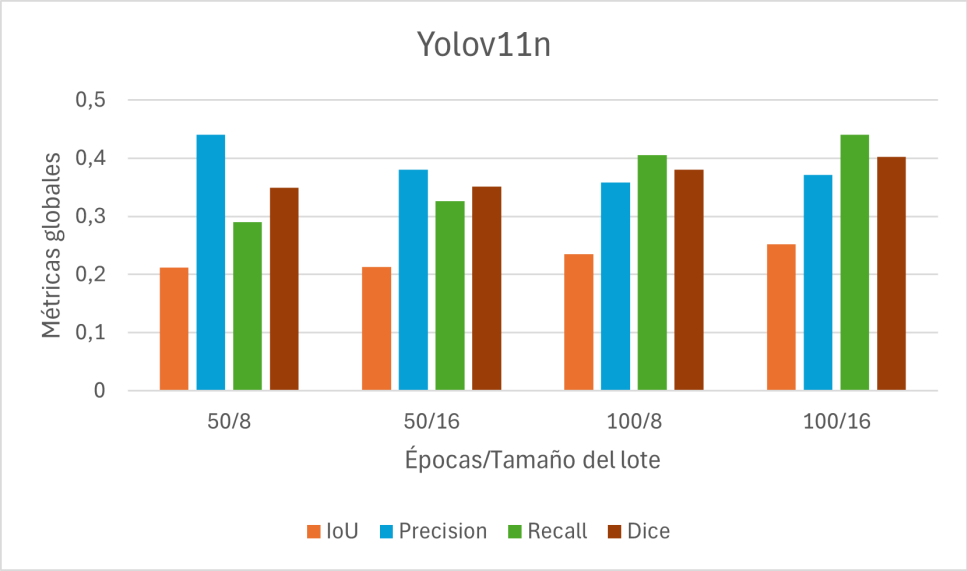

Figura 11: Gráfica de los resultados de los modelos implementados en YOLOv11n.

En general, los resultados indican que, YOLOv11n supera ligeramente a YOLOv8n en la mayoría de métricas, especialmente en configuraciones con mayor número de épocas (100) y

tamaño de lote (16). En particular:

- Yolov11n (100/16) obtuvo el mayor coeficiente *Dice* (0,4026), así como en *Recall* (0,44), lo cual sugiere que el modelo fue capaz de detectar correctamente una mayor proporción de objetos verdaderos.
- También destacó IoU (0,252), métrica crítica para tareas de segmentación, lo que hace confirmar una mejor superposición entre la predicción y la imagen original, respecto a otras configuraciones.
- Por otro lado, YOLOv8n (100/8) mostró sus mejores resultados con valores como la *Precision* (0,3401) y *Recall* (0,4371), sin embargo, los valores de *Dice* e IoU fueron inferiores respecto a la versión más reciente.

Está claro que en ambas versiones del modelo, se puede observar una tendencia creciente en las métricas al aumentar las épocas de entrenamiento, lo que indica una mejora en el modelo con el objetivo de aprender características del conjunto de datos. Sin embargo, el tamaño de *batch* también influye: un *batch size* más grande tiende a estabilizar y mejorar la generalización del modelo, como se puede observar en el rendimiento superior en el caso 100/16 frente a 100/8, especialmente en YOLOv11.

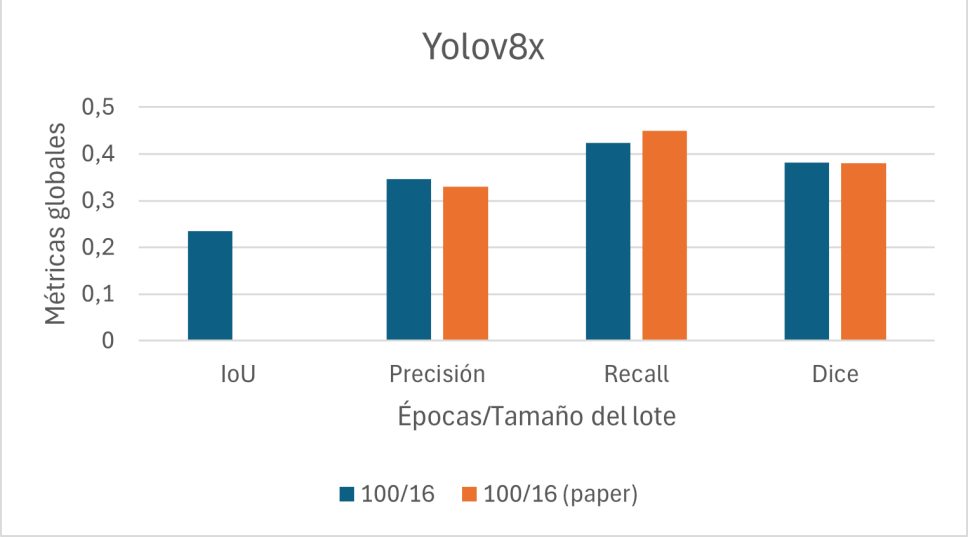

Figura 12: Gráfica de los resultados de los modelos implementados en YOLOv8x.

Por otra parte, a partir de los resultados según la Tabla 6, se puede afirmar que el modelo anterior YOLOv11n con 100 iteraciones y un *batch size* de 16 obtiene un rendimiento ligera-

mente superior al modelo *Extra-large* implementado en este trabajo y reportado en el *paper* original de los datos, como se puede ver en la gráfica de la Figura 12.

- Precisión: El modelo propuesto con el modelo *Extra-large* llega a un valor de 0,346 frente al 0,33 reportado en el *paper*, lo que indica que realiza menos falsos positivos relativos al total de predicciones positivas. Este incremento sugiere una mejora en la capacidad del modelo para evitar segmentaciones que sean incorrectas.

- *Recall:* El valor obtenido (0,4236) es algo menor que 0,45, lo que implica que el modelo propuesto podría estar dejando pasar algunas instancias positivas. No obstante, la diferencia no es sustancial y se encuentra en un margen aceptable.

- *Dice coefficient:* La métrica combinada en ambos casos es prácticamente igual, lo cual muestra un equilibrio entre ambas métricas, validando que modelo entrenado ha alcanzado un rendimiento similar.

- IoU: En el *paper* no se proporciona este valor, sin embargo, el modelo entrenado presenta un valor de 0,2355, que es aceptable para tareas complejas de segmentación.

Dado que las métricas completas del modelo de referencia no están disponibles (TP,FP,FN,TN e IoU), la comparación no puede ser exhaustiva. Sin embargo, los valores disponibles permiten concluir que el modelo desarrollado ofrece resultados válidos y eficaces para el problema que se ha planteado. En base a las métricas globales obtenidas, se concluye que YOLOv11n con 100 épocas y *batch size* de 16 es la configuración que proporciona el mejor rendimiento global. Su mejor equilibrio entre precisión y segmentación (según IoU y Dice) hace que sea la opción más adecuada para abordar la tarea de segmentación del conjunto de datos utilizado en este trabajo.

De igual forma que se han analizado y comparado las métricas globales, se analizan las métricas individuales junto a la desviación estándar. En la Figura 13 y en la Figura 14, se representan gráficamente dichas métricas individuales junto con sus correspondientes barras de error que reflejan la desviación estándar de los resultados, tras el entrenamiento con distintas configuraciones de épocas y de tamaño de lote.

De igual manera, las métricas consideradas fueron IoU, Precisión, *Recall* y el coeficiente *Dice*. Las barras de error permiten visualizar la variabilidad en los resultados obtenidos. Cuanto

más corta es la barra, más consistentes son las métricas entre las distintas muestras evaluadas.

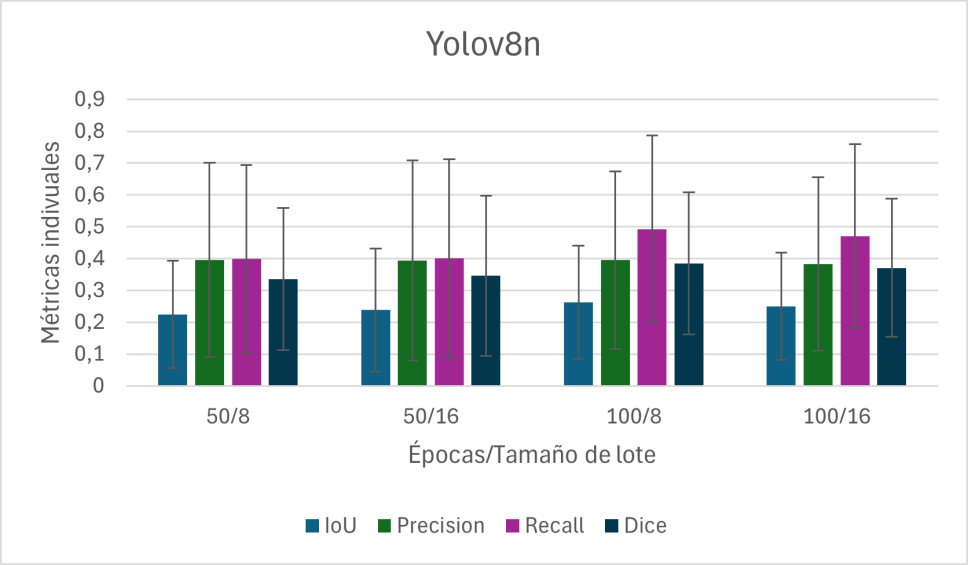

Figura 13: Gráfica de las métricas individuales con la desviación estándar de los modelos implementados en YOLOv8n.

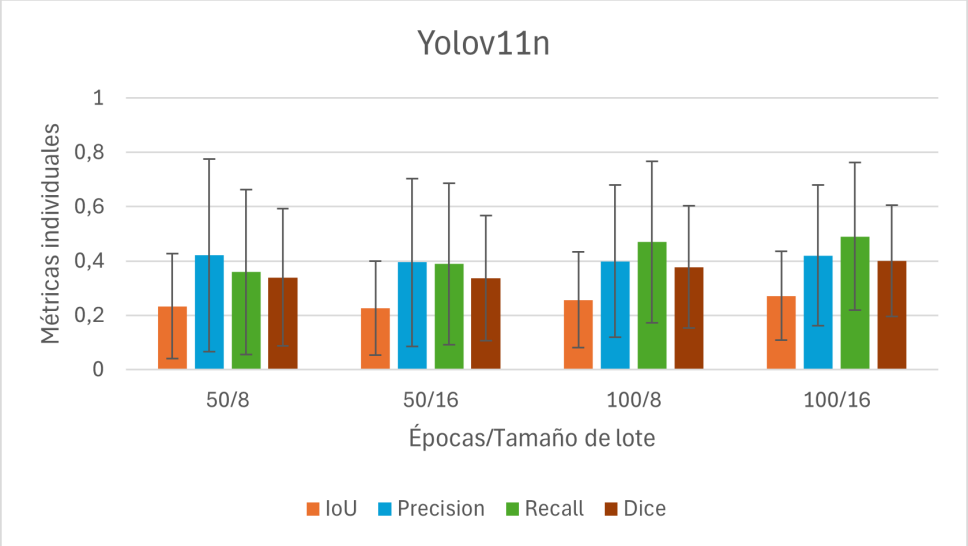

Figura 14: Gráfica de las métricas individuales con la desviación estándar de los modelos implementados en YOLOv11n.

- El modelo YOLOv8n destaca en la métrica de *Recall*, alcanzando su mejor resultado con una configuración de 100 iteraciones y *batch size* 8 (valor=0.4929). Aunque en general presenta una mayor variabilidad en comparación con YOLOv11n.

- En cuanto al coeficiente *Dice* y la precisión, si bien los resultados son correctos, se observa que están sistemáticamente por debajo de los obtenidos con YOLOv11n. La IoU también es ligeramente inferior, indicando una menor precisión espacial en la segmentación de objetos.

- YOLOv11n muestra un rendimiento superior en la mayoría de las métricas. En particular, presenta los valores más altos de IoU, Precisión y *Dice*, con especial hincapié en la configuración de 100 iteraciones y *batch size* 16, donde alcanza un valor del coeficiente *Dice* de 0.4002 y una IoU de 0.271. Además, este modelo muestra una desviación estándar más baja, lo cual sugiere una mayor estabilidad y robustez del modelo frente a otros.

- Aunque el *Recall* de YOLOv11n es ligeramente inferior al de YOLOv8n, la diferencia no es significativa y queda compensada por los mejores resultados en las demás métricas.

En base a las métricas individuales obtenidas, YOLOv8n es más eficaz (recall), aunque con más variación. YOLOv11n presenta un rendimiento más estable, siendo más preciso y con mejor capacidad para evitar falsos positivos. Por tanto, con el objetivo de obtener un modelo más fiable y consistente, YOLOv11n se posiciona como la mejor alternativa, especialmente con configuraciones de entrenamiento más extensas (100 iteraciones y batch size 16).

Finalmente, es importante tener en cuenta el tiempo de ejecución, ya que ha sido un factor determinante en el análisis y la comparación de los modelos. Estos datos se presentan en la Tabla 9.

Cuadro 9: Tiempos de ejecución para cada configuración de modelo, iteraciones y tamaño de lote

Modelo	Iteraciones / Tamaño de lote	Tiempo (min)
Yolov8n	50 / 8	25.13
Yolov8n	50 / 16	23.40
Yolov8n	100 / 8	53.77
Yolov8n	100 / 16	50.37
Yolov8x	100 / 16	**360.20**
Yolov11n	50 / 8	27.53
Yolov11n	50 / 16	24.22
Yolov11n	100 / 8	58.47
Yolov11n	100 / 16	**47.28**

Los tiempos de ejecución demuestran cómo la configuración de entrenamiento afecta directamente al rendimiento computacional. En general, utilizar un tamaño de lote mayor (*batch size:* 16) suele reducir el tiempo de entrenamiento, como se observa especialmente en configuraciones de 50 iteraciones. Sin embargo, la eficiencia también depende de la arquitectura del modelo. El modelo Yolov8x, aunque potencialmente más potente, requiere un tiempo excesivo (más de 6 horas), lo que lo hace poco práctico en contextos con recursos limitados. Por el contrario, los modelos Yolov8n y Yolov11n presentan una mejor eficiencia temporal. Destaca especialmente la configuración Yolov11n con 100 iteraciones y tamaño de lote 16, que completó el entrenamiento en 47.28 minutos, representando la opción más equilibrada entre rendimiento y costo computacional.

4.5. Modificación de hiperparámetros

Tras el análisis comparativo de los modelos entrenados, se ha concluido que la configuración más eficiente corresponde al modelo **YOLOv11n con 100 iteraciones y un *batch size* de 16.** A partir de esta configuración base, se propone llevar a cabo la optimización de hiperparámetros.

4.5.1. Tasa de Aprendizaje (*learning rate)*

Uno de los primeros hiperparámetros que se ha propuesto modificar es el *learning rate* (valor por defecto = 0.01). Este hecho se observa en la Figura 15. Se han realizado pruebas y, posteriormente, se comparan las métricas obtenidas (*IoU, Precision, Recall, Dice*) al final del entrenamiento.

```python
from ultralytics import YOLO

# Cargar el modelo preentrenado de segmentación: (modelo nano)
model = YOLO("yolo11n-seg.pt")

# Entrenae con el dataset, forzando la tarea de segmentación:
results = model.train(
    task    = "segment",
    data    = "/content/datasets/arcadefinal/arcade/stenosis_YOLO/data.yaml",
    epochs  = 100,          #numero de iteraciones.
    batch   = 16,           #número de imágenes por paso de entrenamiento.
    imgsz   = 640,         #tamaño de las imagenes.
    plots   = True,        #guardar gráficos de entrenamiento.
    lr0     = 0.005        #tasa de aprendizaje modificada
)
```

Figura 15: Entrenamiento del modelo modificando la tasa aprendizaje.

Al modificar los valores del *learning rate* a 0.001, 0.005 y 0.02, se observó que las métricas de evaluación del modelo permanecían invariables, es decir, no se apreciaban cambios, por lo que se analizó el comportamiento del entrenamiento y se pudo identificar la causa: el sistema activa por defecto la opción *optimizer=auto*, lo que implica que valores personalizados como *lr0* o *momentum* son ignorados. Es decir, el propio algoritmo selecciona automáticamente la combinación óptima de optimizador y parámetros de entrenamiento. En este caso concreto, seleccionó el optimizador *AdamW*, junto con un valor de *learning rate de 0.002* y un *momentum* de 0.9.

En consecuencia, se desactiva la opción automática añadiendo la opción *optimizer="SGD"* para poder modificar los hiperparámetros de forma manual y obtener el modelo optimizado. Se volvieron a cambiar los valores de *learning rate* y se obtuvieron las métricas presentes en la Tabla 10. De igual forma, se representaron estos datos de manera gráfica; en la Figura 16 se observa cómo cambian los parámetros dependiendo del valor de la tasa de aprendizaje propuesta.

Cuadro 10: Comparativa de métricas globales del modelo YOLOv11n con distintas tasas de aprendizaje (*lr0*)

Métricas Globales	**lr0 = 0.001**	**lr0 = 0.005**	**lr0 = 0.02**
True Positives (TP)	327719	371008	289180
False Positives (FP)	725763	723778	461374
False Negatives (FN)	440118	396829	478654
True Negatives (TN)	77149600	77151585	77413989
IoU	0.2194	0.2487	0.2353
Precisión	0.3111	0.3389	0.3853
Recall	0.4268	0.4832	0.3766
Dice coefficient	0.3599	0.3984	0.3809

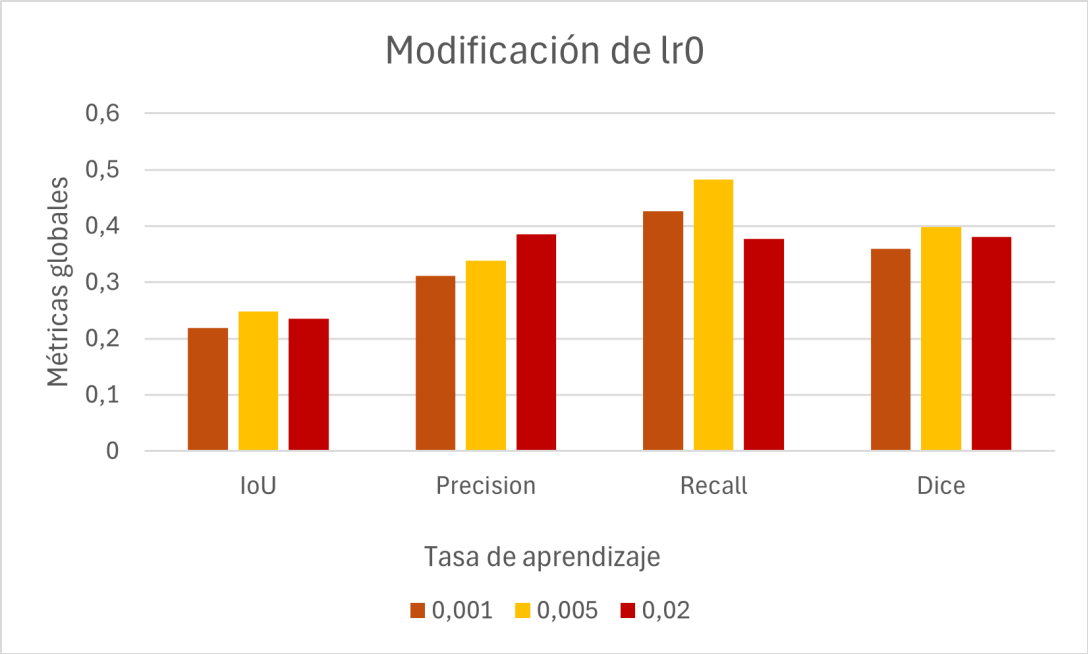

Figura 16: Gráfica de los valores de las métricas modificando la tasa de aprendizaje.

Tras modificar el hiperparámetro utilizando tres valores diferentes (0.001, 0.005 y 0.02), se observó que los resultados variaron levemente, aunque de forma significativa en algunas métricas. El valor de lr0 = 0.005 proporcionó el mejor rendimiento global, obteniendo los valores más altos en todas las métricas evaluadas: IoU (0.2487), Precisión (0.3389), Recall (0.4832) y

Dice (0.3984).Esto hace ver que una tasa de aprendizaje mayor que la predeterminada permite al modelo aprender de manera más eficiente en este caso concreto.

Además, al analizar las métricas absolutas, como los valores TP, FP, FN y TN, también se observan diferencias relevantes. El modelo con lr0 = 0.005 logró el mayor número de verdaderos positivos (371008), lo que está directamente relacionado con el valor más alto de *Recall*. Al mismo tiempo, aunque el número de falsos positivos (723778) es ligeramente superior al de lr0 = 0.02 (461374), el equilibrio alcanzado entre precisión y sensibilidad se traduce en un mejor *Dice coefficient* general.

Por el contrario, el modelo con lr0 = 0.02 muestra una disminución tanto en TP como en *Recall*, lo que podría deberse a inestabilidad durante el entrenamiento. Asimismo, el modelo con lr0 = 0.001 muestra el menor número de verdaderos positivos (327719) y la mayor cantidad de falsos negativos (440118), lo cual refleja un aprendizaje más lento e incompleto. En resumen, se concluye que el ajuste fino de la tasa de aprendizaje tiene un impacto directo no solo sobre las métricas globales, sino también sobre la distribución de aciertos y errores del modelo. **La configuración lr0 = 0.005 representa la configuración más adecuada.**

4.5.2. *Momentum*

Una vez que se ha analizado el efecto que presenta la tasa de aprendizaje, se procede a estudiar el impacto del hiperparámetro *momentum* (valor por defecto = 0.937) en el rendimiento del modelo. Para esta prueba, se mantuvo constante la configuración base y se modificaron únicamente los valores del *momentum*, a través de los valores 0.90 y 0.95. El objetivo fue analizar si afecta positivamente a la estabilidad y precisión del modelo. Los datos obtenidos de las métricas se recogen en la Tabla 11 y su representación gráfica se visualiza en la Figura 17.

Tras las modificaciones del valor del *momentum* a 0.90 y 0.95, se puede observar que los resultados varían ligeramente, pero permiten extraer conclusiones relevantes. En concreto, con un valor de 0.90 se obtuvo un rendimiento ligeramente superior en todas las métricas: IoU (0.245), Precisión (0.3276), *Recall* (0.4928) y Dice (0.3936). Al aumentar el *momentum* a 0.95, estas métricas disminuyeron levemente, indicando una posible pérdida de estabilidad o sobreajuste en las actualizaciones de pesos.

Cuadro 11: Comparativa de métricas globales del modelo YOLOv11n con distinto *Momentum*

Métricas globales	**momentum = 0.90**	**momentum = 0.95**
True Positives (TP)	378372	343936
False Positives (FP)	776601	695050
False Negatives (FN)	389465	423901
True Negatives (TN)	77098762	77180313
IoU	0.2450	0.2351
Precisión	0.3276	0.3310
Recall	0.4928	0.4479
Dice coefficient	0.3936	0.3807

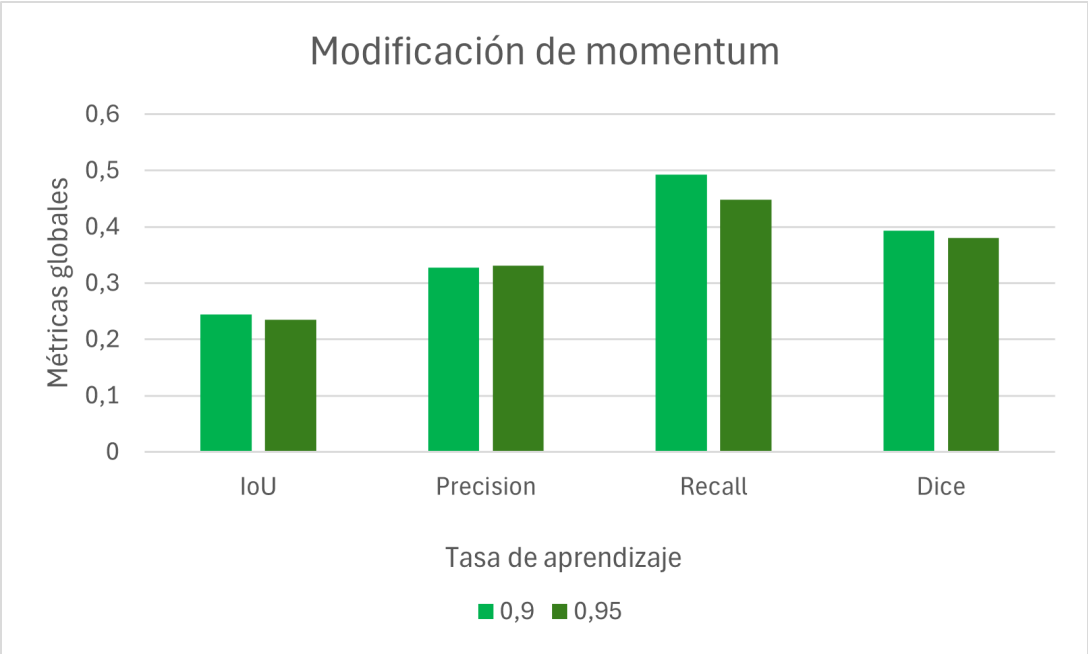

Figura 17: Gráfica de los valores de las métricas modificando el *momentum*.

Al analizar las métricas absolutas, se confirma esta tendencia: el número de TP fue mayor con *momentum* = 0.90 (378372 frente a 343936), mientras que los FN fueron menores (389465 vs. 423901), lo que explica el valor superior del *Recall* en este caso. Aunque el número de FP también fue mayor con el valor de 0.90 (776601 frente a 695050), este ligero aumento fue compensado por la mejora general del resto de métricas. En cambio, al aumentar a 0.95, el modelo

parece experimentar una ligera pérdida de capacidad de generalización, posiblemente debido a un sobreajuste o a una convergencia demasiado agresiva, lo cual se refleja en la disminución de todos los indicadores, especialmente en *Recall* y TP. **En conclusión, un valor de momentum = 0.90 es más adecuado para este caso, ya que proporciona un equilibrio más oportuno.**

4.5.3. *Weight Decay*

De igual manera que en los casos anteriores, el hiperparámetro *weight decay* (también conocido como penalización L2) se analiza con el fin de evaluar su impacto en el rendimiento del modelo. Se utilizarán distintos valores de este hiperparámetro (valor por defecto = 0.0005) ajustándolo a 0.0001 y 0.001. El objetivo es observar cómo influye en las métricas de segmentación (IoU, Precisión, Recall y Dice) y determinar el valor más adecuado para este caso. Al igual que en los análisis anteriores, los resultados obtenidos se recogen en la Tabla 12 y en la Figura 18.

Cuadro 12: Comparativa de métricas globales del modelo YOLOv11n con distinto *Weight Decay*

Métricas globales	**weight decay = 0.0001**	**weight decay = 0.001**
True Positives (TP)	303582	337781
False Positives (FP)	620357	662940
False Negatives (FN)	464255	430056
True Negatives (TN)	77255006	77212423
IoU	0.2187	0.2361
Precisión	0.3286	0.3375
Recall	0.3954	0.4399
Dice coefficient	0.3589	0.3820

Una vez modificado el hiperparámetro *weight decay*, con los valores de 0.0001 y 0.001, se observan claras diferencias en las métricas globales de rendimiento del modelo. En términos generales, el valor más alto de 0.001 ofrece un mejor comportamiento del modelo en todas las métricas: IoU (0.2361 frente a 0.2187), Precisión (0.3375 frente a 0.3286), *Recall* (0.4399 frente a 0.3954) y Dice (0.3820 frente a 0.3589). Esto indica que un mayor grado de regularización

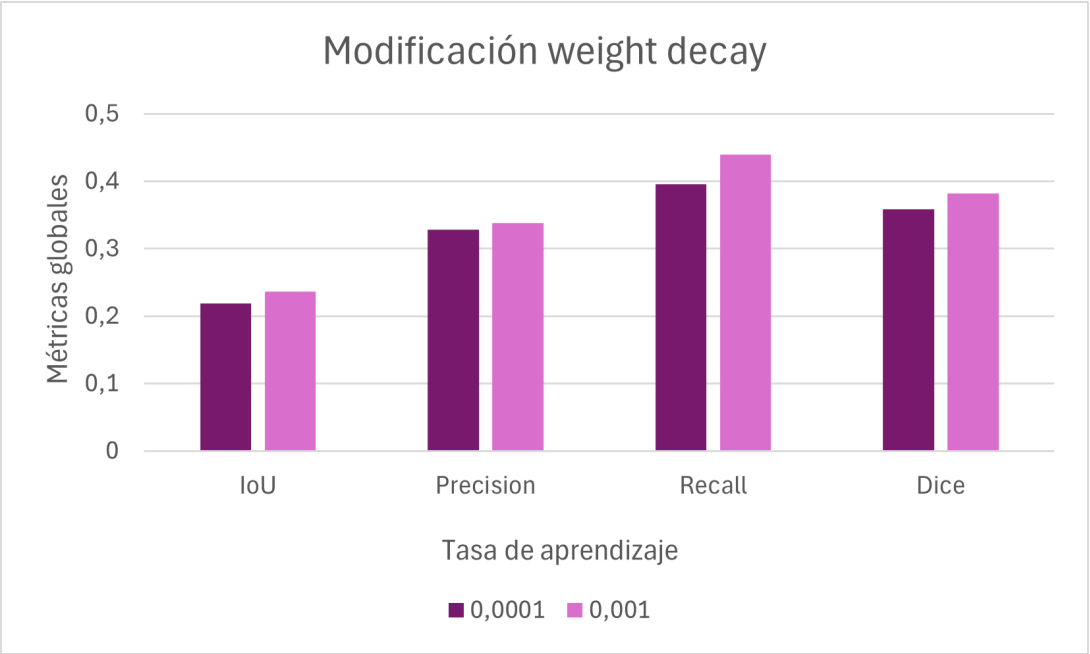

Figura 18: Gráfica de los valores de las métricas modificando el *weight decay*.

ayudó al modelo a evitar el sobreajuste y a mejorar su capacidad de generalización.

El análisis de las métricas absolutas apoya esta conclusión. Con *weight decay* = 0.001, se obtuvo un mayor número de TP (337781 vs. 303582) y una menor cantidad de FN (430056 vs. 464255), lo que repercute directamente en la mejora del *Recall*. Aunque se produjo un incremento en los FP (662940 frente a 620357), este aumento fue compensado por la mejora global en las demás métricas, especialmente en aquellas más relevantes para la segmentación, como el coeficiente de Dice. **En consecuencia, se concluye que con *weight decay* = 0.001 mejora el rendimiento del modelo. Esto sugiere que un ajuste adecuado de este hiperparámetro puede tener un impacto positivo durante el entrenamiento.**

4.5.4. Combinación hiperparámetros

Según los análisis de los apartados anteriores, se han identificado las configuraciones más relevantes que aportaron beneficios al rendimiento del modelo. En concreto, se observó que los valores lr0 = 0.005, *momentum* = 0.90 y *weight decay* = 0.001 ofrecían resultados superiores respecto a sus configuraciones por defecto, reflejándose en métricas como IoU, Precisión, *Recall* y el coeficiente Dice.

Dado que estos hiperparámetros afectan a distintos aspectos del proceso de optimización

como son la velocidad de convergencia, la estabilidad de las actualizaciones y la regularización del modelo, resulta interesante plantear una última prueba en la que se combinen simultáneamente los valores optimizados, como se puede observar en la Figura 19. El objetivo de este experimento final es evaluar si dicha combinación permite obtener un rendimiento aún mayor de forma conjunta, o si, por el contrario, al usarlos a la vez, disminuye el rendimiento. En la Tabla 13 se recogen los resultados obtenidos tras el entrenamiento.

Figura 19: Entrenamiento del modelo con la combinación de los hiperparámetros.

Cuadro 13: Resultados obtenidos tras la combinación de los hiperparámetros

Métricas globales	**weight decay = 0.0001**
True Positives (TP)	331939
False Positives (FP)	549885
False Negatives (FN)	435898
True Negatives (TN)	77325478
IoU	0.2519
Precisión	0.3764
Recall	0.4323
Dice coefficient	0.4024

Los resultados globales obtenidos muestran una mejora en varias métricas, destacando un IoU de 0.2519, Precisión de 0.3764, *Recall* de 0.4323 y un coeficiente Dice de 0.4024, superando los entrenamientos en los que los hiperparámetros fueron modificados individualmente. **Se concluye que la integración conjunta de los tres hiperparámetros afinados ha permitido mejorar el rendimiento general del modelo.**

Finalmente, como añadido a estos resultados, a continuación se presentan algunas imágenes en la Figura 20 que ilustran el resultado del proceso de segmentación con este último modelo optimizado. Estas imágenes permiten observar la calidad y precisión del modelo entrenado.

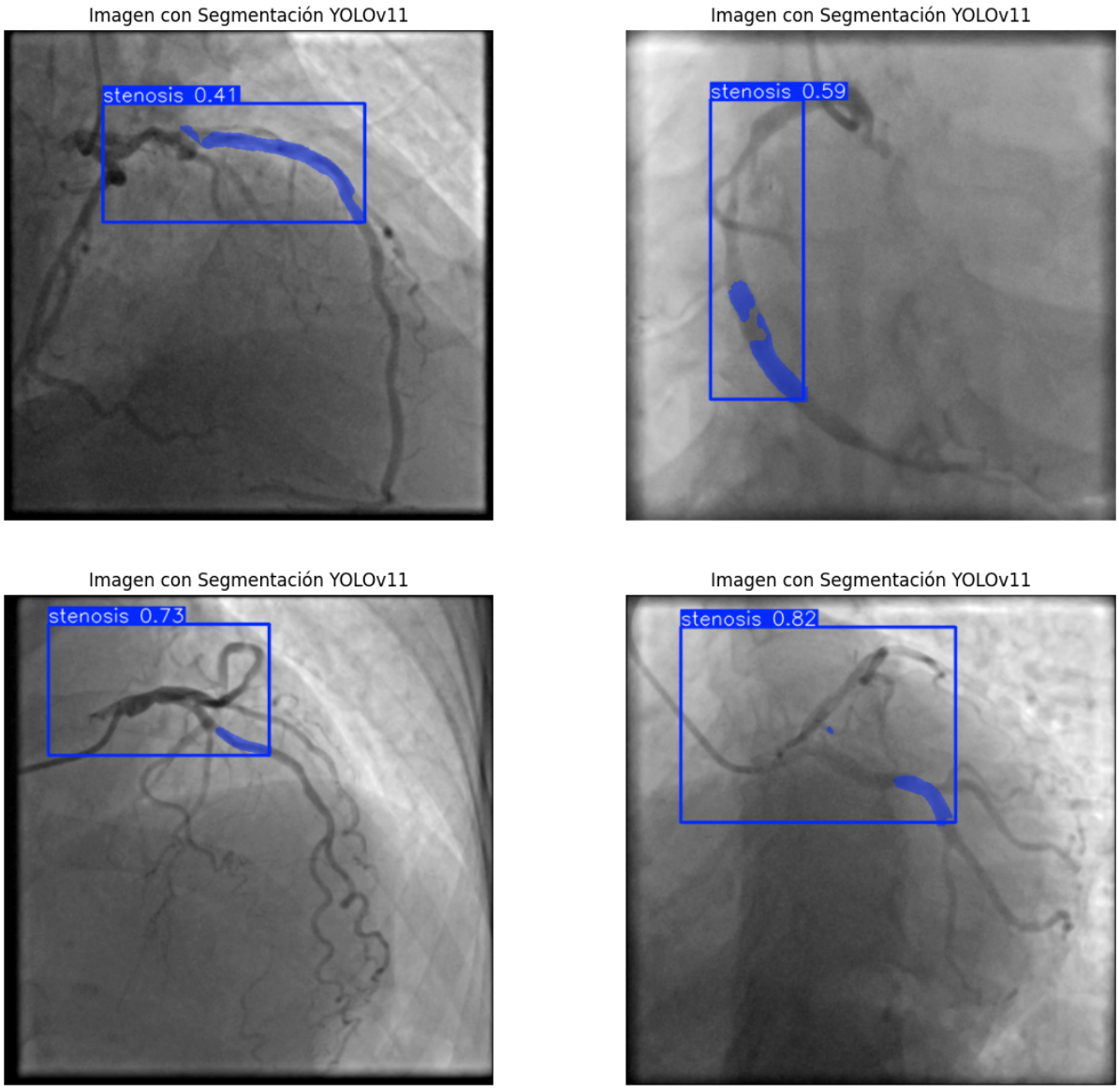

Figura 20: Imágenes segmentadas por el modelo optimizado.

5

Conclusiones y Líneas Futuras

5.1. Conclusiones

En este Trabajo Fin de Grado se ha llevado a cabo el entrenamiento de modelos de aprendizaje profundo mediante redes neuronales convolucionales utilizando la arquitectura de la familia YOLO con el objeto de segmentar estenosis en coronariografías. Se ha realizado hincapié en la importancia que tienen las enfermedades cardiovasculares, sobre todo en la enfermedad de las arterias coronarias, debido a que actualmente es una de las principales causas de mortalidad en todo el mundo. En este sentido, se ha resaltado la importancia de una detección temprana, que puede facilitar tanto el diagnóstico como el tratamiento por parte de los profesionales sanitarios.

En este contexto, se llevaron a cabo varias pruebas utilizando dos versiones distintas de la arquitectura YOLO, modificando parámetros como el número de épocas y el tamaño de lote con el objetivo principal de optimizar el rendimiento del modelo. Tras analizar y comparar los resultados, se ha llegado a la conclusión de que la configuración que ha ofrecido una mejor práctica fue YOLOv11n, con 100 épocas y un tamaño de lote de 16. Posteriormente, se llevó a cabo el entrenamiento de dicho modelo con la modificación de hiperparámetros tanto de manera individual como simultánea; ofreciendo resultados satisfactorios al unificarlos. Sin embargo, se observaron limitaciones en ciertas imágenes, probablemente debido a la baja calidad de los datos y a las restricciones de capacidad y memoria del equipo utilizado.

Este proyecto ha representado un desafío enriquecedor desde el punto de vista académico y personal. Partiendo sin conocimientos previos en Inteligencia Artificial o redes neuronales, el trabajo ha implicado un proceso exhaustivo de estudio, comprensión y práctica de conceptos

complejos. La fase inicial de búsqueda bibliográfica sobre la enfermedad de las arterias coronarias no solo proporcionó el contexto clínico necesario, sino que también me ha permitido ser consciente del gran impacto que puede llegar a tener la tecnología en el ámbito de la salud. A nivel personal, ha sido una experiencia gratificante enfrentarme a un problema real y poder aplicar lo aprendido durante la carrera para buscar soluciones. Trabajar con datos de pacientes reales y contribuir con mi ayuda a un campo con tanto potencial científico ha sido una oportunidad única. Desde pequeña, me ha interesado la combinación de salud y tecnología, y este proyecto ha reforzado mi convicción de que dicha combinación tiene un gran futuro prometedor.

5.2. Líneas Futuras

Este proyecto presenta distintas líneas futuras por las cuales se podría continuar este estudio; por tanto, algunas de las vías de investigación podrían incluir:

- Ampliación y mejora del conjunto de datos: Una limitación importante del trabajo ha sido el tamaño reducido y la calidad de las imágenes. En el futuro, se podría trabajar con conjuntos de datos más amplios o aplicar técnicas de mejora de imagen como filtros de realce de contraste, eliminación de ruido o normalización del brillo para mejorar la calidad visual.

- Cambio de arquitectura: Aunque YOLO ha resultado eficiente y relativamente sencillo de implementar, existen modelos más avanzados que podrían ofrecer un rendimiento superior. Arquitecturas como EfficientDet o incluso modelos de segmentación médica como U-Net o nnUNet son alternativas viables. Además, una línea interesante sería la combinación de modelos: utilizar YOLO para detección inicial y luego aplicar una red especializada en segmentación para delimitar con mayor precisión las áreas afectadas.

- Mejora del entorno de entrenamiento: El entrenamiento de los modelos se ha realizado en un equipo con recursos limitados (GPU de baja capacidad), lo cual ha condicionado al entrenamiento. Para superar esta limitación, podrían utilizarse plataformas en la nube como Google Colab Pro o Microsoft Azure, que permiten el entrenamiento en GPUs de alto rendimiento y con un mayor volumen de datos.

- Evaluación de distintas métricas: Aunque se han utilizado métricas estándar como precisión, *recall* y mAP, sería interesante incorporar otras métricas más relevantes en el ámbito médico, como la curva ROC, el área bajo la curva (AUC), o medidas basadas en la correlación con diagnósticos clínicos reales.

En resumen, este trabajo ha sido un primer paso hacia la integración de modelos de aprendizaje profundo en el análisis automático de imágenes médicas. Aunque aún queda un camino por recorrer en cuanto a precisión, fiabilidad y validación clínica, los resultados obtenidos y el conocimiento adquirido abren la puerta a futuras investigaciones que podrían tener un impacto significativo en la práctica clínica y en la vida de los pacientes.

Referencias

[1] MathWorks. (2024) What is a neural network? MathWorks. Accedido el 23 de mayo de 2025. [Online]. Available: https://la.mathworks.com/discovery/neural-network.html

[2] W. A. Castañeda Sánchez, "Redes neuronales artificiales: una medición de aprendizajes de pronósticos como demanda potencial," *Universidad, Ciencia y Tecnología*, vol. 27, no. 118, 2023, publicado en línea el 5 de enero de 2024. Consultado el 24 de mayo de 2025. [Online]. Available: https://ve.scielo.org/scielo.php?pid=S1316-48212023000100051&script=sci_arttext

[3] MathWorks. (2024) Red neuronal convolucional (cnn). MathWorks. Consultado el 24 de mayo de 2025. [Online]. Available: https://es.mathworks.com/discovery/convolutional-neural-network.html

[4] National Heart, Lung, and Blood Institute. (2023) Aterosclerosis. Instituto Nacional del Corazón, los Pulmones y la Sangre. Consultado el 24 de mayo de 2025. [Online]. Available: https://www.nhlbi.nih.gov/es/salud/aterosclerosis

[5] M. A. García Fernández, E. Pérez David, M. Moreno, J. A. García-Robles, and C. Macaya, "Tomografía computarizada multidetectora en el estudio cardíaco," *Clínica e Investigación en Arteriosclerosis*, vol. 19, no. 1, pp. 49–55, 2007.

[6] J. D. Domingo. (2022) Deep learning en visión artificial. CARTIF. Accedido el 30 de mayo de 2025. [Online]. Available: https://blog.cartif.es/deep-learning-vision-artificial/

[7] K. E. Amraoui, M. E. Ansari, M. Lghoul, M. E. Alaoui, and J. V. de Oliveira. (2024) Embedding a real-time strawberry detection model into a pesticide-spraying mobile robot for greenhouse operation.

[8] M. Popov, A. Amanturdieva, N. Zhaksylyk, A. Alkanov, A. Saniyazbekov, T. Aimyshev, E. Ismailov, A. Bulegenov, A. Kuzhukeyev, A. Kulanbayeva, A. Kalzhanov, N. Temenov, A. Kolesnikov, O. Sakhov, and S. Fazli, "Dataset for automatic region-based coronary artery disease diagnostics using x-ray angiography images," *Scientific Data*, vol. 11, p. 20, 2024. [Online]. Available: https://www.nature.com/articles/s41597-023-02871-z

[9] P. Libby and P. Theroux, "Pathophysiology of coronary artery disease," *Circulation*, vol. 111, no. 25, pp. 3481–3488, 2005. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/15983262/

[10] H. R. Fazlali, N. Karimi, S. M. R. Soroushmehr, S. Shirani, B. K. Nallamothu, K. R. Ward, S. Samavi, and K. Najarian, "Vessel segmentation and catheter detection in x-ray angiograms using superpixels," *Medical & Biological Engineering & Computing*, vol. 56, no. 9, pp. 1515–1530, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s11517-018-1793-4

[11] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, pp. 685–695, 2021. [Online]. Available: https://doi.org/10.1007/s12525-021-00475-2

[12] IBM. (s.f.) Tipos de aprendizaje automático. Consultado el 29 de mayo de 2025. [Online]. Available: https://www.ibm.com/es-es/think/topics/machine-learning-types

[13] MathWorks. (s.f.) Training a model from scratch. Consultado el 29 de mayo de 2025. [Online]. Available: https://es.mathworks.com/solutions/deep-learning/examples/training-a-model-from-scratch.html

[14] ——. (s.f.) Transferencia del aprendizaje. Consultado el 29 de mayo de 2025. [Online]. Available: https://es.mathworks.com/discovery/transfer-learning.html

[15] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016, consultado el 29 de mayo de 2025. [Online]. Available: https://ieeexplore.ieee.org/document/7426826

[16] MedlinePlus, "Enfermedad de la arteria coronaria," https://medlineplus.gov/spanish/coronaryarterydisease.html, accedido: 10 de mayo de 2025.

[17] I. Ferreira-González, "Epidemiología de la enfermedad coronaria," *Revista Española de Cardiología*, vol. 67, no. 2, pp. 139–144, Feb 2014. [Online]. Available: https://www.revespcardiol.org/es-epidemiologia-enfermedad-coronaria-articulo-S0300893213004855

[18] Instituto Nacional de Estadística (INE). (2024) Estadísticas de defunciones según la causa de muerte. primer semestre de 2023. Instituto Nacional de Estadística. Accedido el 28 de mayo de 2025. [Online]. Available: https://www.ine.es/dyngs/Prensa/pEDCM2023.htm

[19] A. Buendía Hernández, "Enfermedad arterial coronaria o cardiopatía isquémica," *Archivos de Cardiología de México*, vol. 79, no. 4, pp. 279–285, 2009.

[20] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. M. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841516000220

[21] M. Martínez-Sellés and J. A. García Robles, "Síndrome coronario agudo sin elevación del segmento st. diagnóstico y estratificación del riesgo en urgencias," *Emergencias*, vol. 23, no. 3, pp. 183–191, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1131358711150074

[22] H. Rodríguez-Reyes and M. A. Villarreal-Alarcón, "Enfermedad arterial coronaria o cardiopatía isquémica. concepto, fisiopatología y clasificación," *Archivos de Cardiología de México*, vol. 79, no. 3, pp. 250–258, 2009.

[23] T. Du, L. Xie, H. Zhang, X. Liu, X. Wang, D. Chen, Y. Xu, Z. Sun, W. Zhou, L. Song, C. Guan, A. J. Lansky, and B. Xu, "Training and validation of a deep learning architecture for the automatic analysis of coronary angiography," *EuroIntervention*, vol. 17, no. 1, pp. 32–40, 2021.

[24] Y. Zhang, L. Wang, H. Li, J. Chen, M. Zhao, and Q. Liu, "Automatic coronary artery segmentation and stenosis detection in x-ray angiograms using deep learning," *IEEE Transactions on Medical Imaging*, vol. 41, no. 12, pp. 3430–3440, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9937189

[25] J.-E. Hwang, S. Han, H.-J. Park, M. J. Kim, Y.-H. Jeong, I.-H. Cho, H.-J. Park, Y.-D. Kim, and H.-J. Chang, "Deep learning segmentation of major vessels in x-ray coronary angiography," *Scientific Reports*, vol. 9, no. 1, p. 16897, 2019. [Online]. Available: https://www.nature.com/articles/s41598-019-53254-7

[26] IBM Cloud Education. (2020) Segmentación de imágenes: definición y técnicas. Consultado el 25 de mayo de 2025. [Online]. Available: https://www.ibm.com/es-es/topics/image-segmentation

[27] Ultralytics. (2025) Documentación oficial de ultralytics. Recuperado el 25 de mayo de 2025, de https://docs.ultralytics.com/. [Online]. Available: https://docs.ultralytics.com/es/#where-to-start

[28] ——. (2025) Segmentación de instancias. Recuperado el 25 de mayo de 2025, de https://docs.ultralytics.com/es/tasks/segment/. [Online]. Available: https://docs.ultralytics.com/es/tasks/segment/

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA