UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Máster en Ingeniería del Software e Inteligencia Artificial

Segmentación Semántica de Lesiones de Esclerosis Múltiple en Resonancias Magnéticas mejorada por Super-Resolución

Semantic Segmentation of Multiple Sclerosis Lesions in Magnetic Resonance Images Enhanced by Super-Resolution

Realizado por
Samuel Sánchez Toca

Tutorizado por
Ezequiel López Rubio
Jorge García González

Departamento
Lenguajes y ciencias de la computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, junio de 2025

UNIVERSIDAD
DE MÁLAGA

E.T.S. INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN MÁSTER DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

Segmentación Semántica de Lesiones de Esclerosis Múltiple en Resonancias Magnéticas mejorada por Super-Resolución

Semantic Segmentation of Multiple Sclerosis Lesions in Magnetic Resonance Images Enhanced by Super-Resolution

Realizado por
Samuel Sánchez Toca

Tutorizado por
Ezequiel López Rubio
Jorge García González

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2025

Fecha defensa: julio de 2025

Abstract

Este Trabajo de Fin de Máster presenta el desarrollo de un sistema automático para la segmentación de lesiones de esclerosis múltiple (EM) en imágenes de resonancia magnética cerebral, utilizando técnicas basadas en redes neuronales profundas y comparando dos arquitecturas ampliamente utilizadas: U-Net, orientada a segmentación semántica, y YOLO, originalmente diseñada para detección, pero adaptada en este trabajo para tareas de segmentación. El estudio explora distintas estrategias de selección de cortes 2D a partir de volúmenes 3D, así como la aplicación de técnicas de super-resolución (FSRCNN) sobre las imágenes antes del entrenamiento. Los resultados cuantitativos muestran que U-Net, especialmente cuando se combina estrategias de selección de cortes y superresolución, ofrece un mejor rendimiento general. Por su parte, YOLO destaca por su rápida velocidad de inferencia, aunque presenta un desempeño inferior. El análisis cualitativo refuerza estas conclusiones, evidenciando diferencias importantes en la forma en que cada red segmenta las lesiones. Este trabajo pone de manifiesto la importancia del preprocesamiento, la selección de datos y selección de red neuronal en la segmentación médica, y plantea futuras líneas de mejora para lograr modelos más precisos y eficientes.

Keywords: **Segmentación médica, Esclerosis múltiple, Resonancia magnética, U-Net, YOLO**

Índice

1. Introducción **5**

1.1. Motivación del trabajo . . . 6

1.2. Objetivos del trabajo . . . 6

1.3. Descripción general del enfoque . . . 7

2. Estado del arte **9**

2.1. Imagen por resonancia magnética . . . 9

2.2. Métodos de segmentación . . . 10

2.2.1. Basados en intensidad . . . 11

2.2.2. Basados en regiones . . . 11

2.2.3. Aprendizaje automático tradicional . . . 11

2.2.4. Aprendizaje profundo . . . 11

2.2.5. Enfoques híbridos . . . 12

2.3. Super-resolución en imágenes MRI . . . 12

3. Fundamentos teóricos **15**

3.1. Redes neuronales convolucionales . . . 15

3.2. Segmentación semántica . . . 17

3.3. U-Net . . . 18

3.4. YOLO . . . 21

3.5. Super-resolución con FSRCNN . . . 22

4. Metodología **25**

4.1. Enfoque experimental . . . 25

4.2. Conjunto de datos y preprocesado . . . 26

4.3. División del conjunto de datos . . . 28

4.4. Flujo de preprocesado . . . 31

4.5. Estrategia de selección de imágenes . . . 32

4.6. Métricas utilizadas . . . 33

4.7. Entrenamiento 34

5. Resultados **37**

5.1. Evaluación cuantitativa 37

5.2. Evaluación cualitativa 42

6. Conclusiones y líneas futuras **47**

Apéndice A. Diseño e implementación **57**

A.1. Arquitectura software 57

A.2. Entorno de desarrollo 59

A.3. Estructura del proyecto 60

A.4. Implementación del sistema 62

A.5. FSRCNN 67

A.6. U-Net 69

A.7. YOLO 73

1

Introducción

La esclerosis múltiple (EM) es una enfermedad neurológica crónica que afecta a millones de personas en todo el mundo. Esta enfermedad causa inflamación en varias áreas del cerebro, lo que provoca daño a la mielina, el tejido graso que rodea y aísla las fibras nerviosas, fundamental para la función neurológica adecuada [1]. La resonancia magnética (RM) es una herramienta fundamental para el diagnóstico de la EM, así como para monitorizar su progresión y evaluar la respuesta a los tratamientos.

Dentro del manejo clínico de la EM, la segmentación precisa de las lesiones es esencial para la cuantificación volumétrica de la carga de lesiones. Esta cuantificación es importante para comprender y caracterizar la progresión de la enfermedad. Sin embargo, la segmentación manual de las lesiones de EM en las exploraciones de RM es una tarea muy ardua, que requiere mucho tiempo y una experiencia significativa por parte de los expertos Además, esta tarea manual es propensa a la variabilidad, tanto entre diferentes observadores (inter-observador) como para el mismo experto en momentos distintos (intra-observador). La dificultad y la subjetividad asociadas a la segmentación manual ponen de manifiesto una necesidad no satisfecha de desarrollar métodos automáticos que puedan segmentar las lesiones de EM de manera consistente y eficiente [1] . El desarrollo de técnicas de segmentación de lesiones automatizadas es un paso clave para avanzar en la gestión clínica y optimizar el tratamiento para las personas con EM . Los avances recientes en el aprendizaje profundo han demostrado un potencial significativo para automatizar tareas en el análisis de imágenes médicas [2] . Sin embargo, los métodos de aprendizaje profundo requieren la disponibilidad de grandes cantidades de datos para el entrenamiento, y existe una limitación en cuanto a la disponibilidad de grandes conjuntos de datos con segmentaciones manuales precisas realizadas por expertos [3].

1.1. Motivación del trabajo

La segmentación precisa de las lesiones de EM en imágenes de resonancia magnética es un reto fundamental en el ámbito clínico. Aunque la RM es una herramienta clave para el diagnóstico y seguimiento de esta enfermedad, la segmentación manual de las lesiones continúa siendo una tarea compleja.

En este contexto, los métodos automáticos de segmentación basados en redes neuronales profundas representan una oportunidad para mejorar significativamente la eficiencia, precisión y reproducibilidad del proceso. El desarrollo de sistemas automáticos adaptados a este problema permitiría optimizar la práctica clínica y servir como herramienta adicional a los médicos en su trabajo.

Dada la necesidad e importancia de métodos automáticos para la segmentación de las lesiones de EM, este trabajo busca explorar y desarollar un sistema automático de segmentación de lesiones usando redes neuronales.

1.2. Objetivos del trabajo

El objetivo principal de este trabajo es desarrollar un sistema automático para la segmentación de lesiones de EM a partir de imágenes de RM, utilizando técnicas de aprendizaje profundo. Este objetivo general se concreta en los siguientes subobjetivos:

- Diseñar e implementar un programa de segmentación automática que incluya implementaciones de las siguientes redes neuronales:
  - YOLO (You Only Look Once), diseñada para detección de objetos, aunque también puede realizar segmentación. Destaca por su capacidad de procesamiento rápido en tiempo real [4].
  - U-Net, diseñada específicamente para segmentación pixel a pixel en imágenes biomédicas [5].
  - FSRCNN (Fast Super-Resolution Convolutional Neural Network): esta red está diseñada para la superresolución de imágenes [6].
- Investigar el impacto del uso FSRCNN en la etapa de preprocesamiento del conjunto de datos. Se aplicará superresolución con FSRCNN a las imágenes y máscaras

originales antes del entrenamiento.

- Comparar el rendimiento del sistema con y sin mejora de resolución para determinar si esta técnica aporta beneficios reales en la segmentación de lesiones y también determinar qué red de segmentación rinde mejor. Para ello, se entrenarán y evaluarán las redes según métricas estándar.

1.3. Descripción general del enfoque

Este trabajo adopta un enfoque dual para abordar la tarea de segmentación automática de lesiones de EM en imágenes de RM. En primer lugar, se establece una línea base mediante el entrenamiento de redes YOLO y U-Net con imágenes originales extraídas de un conjunto de datos anotado. En segundo lugar, se explora una estrategia alternativa basada en el preprocesamiento de las imágenes y sus respectivas máscaras con técnicas de super-resolución.

Para ello, se emplea la red FSRCNN con el objetivo de mejorar la resolución espacial de las imágenes, lo cual podría facilitar una mejor detección y segmentación de lesiones pequeñas o poco definidas. Las imágenes mejoradas se utilizan posteriormente para entrenar las arquitecturas YOLO y U-Net, evaluando así el impacto del aumento de resolución en el rendimiento de los modelos.

Durante todo el proceso se garantiza una correcta separación entre los conjuntos de entrenamiento, validación y prueba, evitando cualquier solapamiento de datos que pueda inducir sesgos derivados de correlaciones entre imágenes del mismo paciente. Finalmente, se evalúa el rendimiento mediante métricas estándar definidas en el capítulo Metodologías.

2

Estado del arte

En esta sección se revisa el estado del arte relacionado con la segmentación de imágenes médicas, con especial atención al caso de la RM de imágenes cerebrales. Se analizan los distintos enfoques en segmentación semántica, así como las arquitecturas de redes neuronales empleadas en este trabajo: U-Net y YOLO. Asimismo, se examinan las técnicas recientes de mejora de resolución mediante redes neuronales, haciendo énfasis en el modelo FSRCNN. Esta revisión proporciona el contexto necesario para justificar las decisiones metodológicas adoptadas en el presente trabajo y permite situar la propuesta dentro del marco actual de investigación.

2.1. Imagen por resonancia magnética

La RM es una de las técnicas de obtención de imágenes médicas más utilizadas y seguras en la actualidad. A diferencia de otros métodos como la radiografía, la tomografía computarizada (CT), la tomografía por emisión de positrones (PET), la SPECT o el ultrasonido, la RM no utiliza radiación ionizante, lo que la hace adecuada para estudios repetidos sin riesgo para el paciente.

La RM se basa en el comportamiento de los protones de hidrógeno, abundantes en el cuerpo humano, cuando son expuestos a un campo magnético estático. Bajo este campo, los protones alinean su magnetización y, al aplicar una onda de radiofrecuencia, se perturban. Cuando regresan a su estado de equilibrio, emiten una señal que se registra para formar la imagen [7]. Este proceso permite generar imágenes con diferente contraste, dependiendo de tres parámetros fundamentales:

- Densidad de protones (ρ)
- Tiempo de relajación T1 (spin-rejilla): cuánto tarda la magnetización en volver a

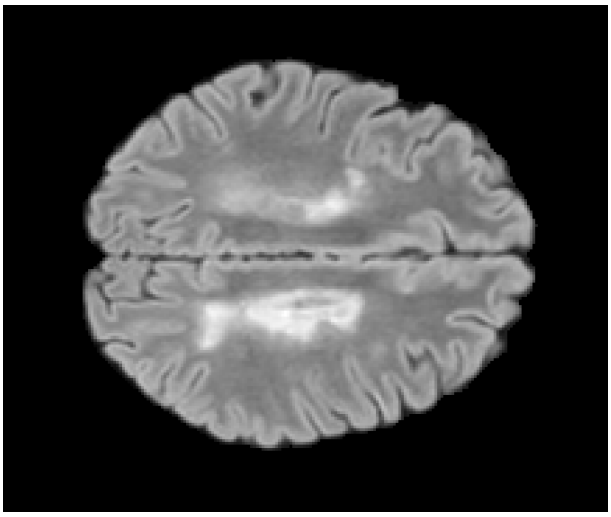


Figura 1: Imagen FLAIR de un paciente con EM (la lesión es la zona blanca)

su equilibrio longitudinal.

- Tiempo de relajación T2 (spin-spin): cuánto tarda en desaparecer la coherencia transversal de los protones.

A partir de estos parámetros, se pueden generar diferentes tipos de imágenes: T1-weighted, T2-weighted y ρ-weighted, cada una útil para resaltar distintos tipos de tejido o anomalías [8]. Además, se emplean secuencias especializadas como FLAIR (Fluid Attenuated Inversion Recovery), que suprime el líquido cefalorraquídeo para facilitar la detección de lesiones hiperintensas periventriculares —una característica típica de la esclerosis múltiple— (ver Figura 1). La secuencia FLAIR combina sensibilidad a lesiones como en T2, pero con mejor contraste en zonas cercanas a los ventrículos cerebrales [9].

2.2. Métodos de segmentación

La segmentación en imágenes médicas ha evolucionado significativamente, desde métodos clásicos basados en intensidad hasta enfoques modernos de aprendizaje profundo. Esta evolución ha estado motivada por la necesidad de mejorar la precisión, robustez y automatización en la detección de estructuras y lesiones, como en el caso del cerebro en estudios de RM.

2.2.1. Basados en intensidad

Los métodos clásicos, como el método del valor umbral (thresholding) y las técnicas basadas en regiones, utilizan directamente los valores de intensidad de los píxeles. Un ejemplo temprano es el método de Otsu [10]. Aunque simples y rápidos, estos enfoques son sensibles al ruido y a la no homogeneidad de intensidad, por lo que su desempeño es limitado en imágenes complejas.

2.2.2. Basados en regiones

Estas técnicas identifican regiones conectadas con intensidades similares, a partir de un punto inicial. Ejemplos incluyen métodos con contornos activos localizados (LRACM) [11] o análisis de simetría en 3D [12]. Aunque más robustas que el método del valor umbral, dependen de parámetros manuales y pueden ser sensibles a la inicialización.

2.2.3. Aprendizaje automático tradicional

El machine learning clásico, como k-means, SVM (*Support Vector Machine*) o *random forests*, mejora la segmentación al utilizar características extraídas (intensidad, textura, forma). Métodos como los super-vóxeles combinados con *random forest* [13], o modelos semi-supervisados con restricciones espaciales [14], resultan en mejores segmentaciónes. Sin embargo, muchos de estos enfoques siguen dependiendo de la ingeniería manual de características y pueden ser computacionalmente intensivos.

2.2.4. Aprendizaje profundo

Con la llegada del aprendizaje profundo, la segmentación alcanzó nuevos niveles de precisión. Las redes convolucionales (CNN) permiten aprender directamente las características relevantes desde los datos, reduciendo la necesidad de diseño manual:

- 2D CNNs como las de Pereira et al. [15] y Havaei et al. [16] mostraron avances notables, aunque limitados por la falta de contexto tridimensional.
- Modelos 2.5D abordan este problema procesando múltiples cortes adyacentes [17], [18].

- Redes 3D como las de Myronenko [19] o Baid et al. [20] permiten un análisis volumétrico completo, aunque requieren más recursos computacionales.

2.2.5. Enfoques híbridos

Los métodos híbridos combinan lo mejor de varios enfoques:

- Aprendizaje profundo junto con aprendizaje automático tradicional, como por ejemplo CNN con *Conditional Random Field* (CRF) [21].
- Segmentación basada en contornos junto con aprendizaje automático, como modelos que combinan *random forests* con contornos activos [22].
- Metaheurísticas junto con *clustering*, como *Fuzzy C-Means* (FCM) con algoritmos evolutivos [23].

2.3. Super-resolución en imágenes MRI

Los algoritmos de super-resolución se basan en un modelo general de adquisición que describe cómo una imagen de alta resolución ideal X se degrada durante el proceso de captura para dar lugar a una secuencia de imágenes de baja resolución $Y_k Y_k$. Este modelo, ampliamente aceptado en la literatura, se puede expresar como:

$$Y_k = D_k B_k G_k X + V_k \tag{1}$$

donde G_k representa la transformación geométrica (como el movimiento del paciente), B_k es el operador de desenfoque asociado a la adquisición de las propias imágenes RM (PSF), D_k es el operador de submuestreo, y V_k representa el ruido aditivo, generalmente gaussiano. Para simplificar, esta expresión también puede escribirse de forma compacta como $Y_k = W_k X + V_k Y_k = W_k X + V_k$, donde W_k es el operador global que resume todos los efectos degradantes. La tarea de SR consiste en invertir este modelo para estimar la imagen X, lo cual constituye un problema mal planteado, razón por la cual se recurre a técnicas de optimización [24].

A partir de este modelo, se han propuesto diversos enfoques para invertir el proceso y estimar X. Entre ellos destacan los métodos basados en *Iterative Back-Projection*,

técnicas deterministas con regularización (como mínimos cuadrados o variación total), enfoques estadísticos bayesianos como el estimador MAP, y métodos geométricos como la proyección en conjuntos convexos (POCS) [24]. Todos ellos comparten la necesidad de incorporar información a priori para compensar el carácter mal planteado del problema.

Sin embargo, estos métodos tradicionales presentan limitaciones. En primer lugar, dependen fuertemente de una estimación precisa del movimiento entre cortes o adquisiciones (G_k), lo cual resulta problemático en presencia de movimientos impredecibles o deformaciones anatómicas sutiles. Además, muchos algoritmos asumen que el perfil de desenfoque (PSF) es conocido y constante, lo cual rara vez se cumple en escenarios clínicos. Por otro lado, el ruido en RM no siempre se ajusta al modelo gaussiano simple utilizado en estas formulaciones, y la variabilidad del contraste entre sujetos también dificulta la generalización. Así pues, aunque estos métodos son usados, requieren parámetros cuya elección puede tener un mayor impacto en el resultado que el propio método seleccionado [24], lo que puede limitar su aplicabilidad práctica en entornos clínicos. Además, su capacidad para recuperar estructuras finas, especialmente en el contexto de lesiones pequeñas o bordes difusos como ocurre en la esclerosis múltiple, puede verse comprometida por la sensibilidad a errores de registración o la imposición de regularizadores demasiado restrictivos.

En vista de las limitaciones de los métodos clásicos de super-resolución en resonancia magnética, el uso de redes neuronales profundas ha emergido como una alternativa eficaz y flexible para mejorar la resolución espacial sin necesidad de suposiciones estrictas sobre el modelo de adquisición. Por un lado, técnicas como SMORE (Self Super-Resolution) han demostrado su aplicabilidad directa en imágenes clínicas reales, utilizando redes del estado del arte como EDSR para generar datos de entrenamiento directamente a partir de las propias imágenes adquiridas [25]. Esto elimina la necesidad de conjuntos de datos externos de entrenamiento y permite preservar detalles patológicos mientras se mejora la visualización en múltiples contextos clínicos, como esclerosis múltiple, remodelado cardíaco o segmentación ventricular [26]. Por otro lado, arquitecturas más complejas como GAN-CPCE y GAN-CIRCLE, han mostrado resultados prometedores al ser usadas en imágenes de resonancia magnética, alcanzando mejoras cuantificables en métricas como PSNR y SSIM [27]. Estas redes combinan estrategias de entrenamiento para producir

imágenes detalladas con bordes más definidos, lo cual es especialmente útil en tareas de diagnóstico estructural. En conjunto, estas propuestas demuestran que los enfoques basados en aprendizaje profundo no solo superan en calidad a las técnicas tradicionales o basadas en interpolación. Esto refuerza la elección de redes neuronales profundas como método para abordar la super-resolución en este trabajo.

3

Fundamentos teóricos

En esta sección se abordan la teoría necesaria para comprender el enfoque propuesto en este trabajo. El objetivo principal es presentar los conceptos clave que sustentan el uso de técnicas de aprendizaje profundo en la segmentación semántica de lesiones de esclerosis múltiple a partir de imágenes de resonancia magnética.

Se comienza con una introducción a las redes neuronales convolucionales, que constituyen la base de los modelos empleados. A continuación, se describen las arquitecturas específicas utilizadas en este trabajo: U-Net y YOLO, ambas ampliamente reconocidas por su eficacia en tareas de segmentación. Posteriormente, se describe la segmentación semántica y finalmente se detalla la técnica de super-resolución mediante la red FSRCNN, que se emplea como estrategia de mejora de calidad de las imágenes de entrada.

3.1. Redes neuronales convolucionales

Las redes neuronales convolucionales son una arquitectura especializada de redes profundas diseñada para procesar datos con estructura espacial, como las imágenes, que pueden interpretarse como una matriz bidimensional de píxeles. A diferencia de las redes densas tradicionales, las CNN explotan las propiedades locales de las imágenes mediante convoluciones, lo que las convierte en una herramienta muy eficaz para tareas como clasificación, detección y segmentación [28]. En términos generales, una CNN se define por el uso de capas convolucionales, aunque su estructura concreta puede variar según la tarea: algunas incluyen capas totalmente conectadas al final (como en clasificación), mientras que otras son completamente convolucionales (como en segmentación o detección). Sus principales ventajas incluyen:

- **Reducción de parámetros**: al reutilizar filtros a lo largo de la imagen, se disminuye significativamente el número de pesos, reduciendo el riesgo de sobreajuste y mejorando la eficiencia computacional.
- **Captura de relaciones espaciales locales**: los filtros pueden detectar patrones como bordes, texturas o formas, respetando la estructura espacial de los datos.
- **Invarianza a translaciones**: gracias a los pesos compartidos y operaciones de pooling, las CNN son robustas a pequeños desplazamientos y deformaciones en la imagen [29].

Una CNN está compuesta por una serie de capas que transforman progresivamente la imagen de entrada en una representación de alto nivel útil para la tarea objetivo. A continuación se describen sus componentes principales:

- **Capa convolucional**: aplica filtros que recorren la imagen y generan mapas de activación. Cada filtro aprende a detectar una característica visual concreta.
- **Capa de activación**: normalmente se aplica una función como ReLU para introducir no linealidad y permitir el aprendizaje de relaciones complejas.
- **Capa de agrupamiento (pooling)**: reduce la dimensión espacial de los mapas de activación aplicando filtros (por ejemplo, de tamaño 2×2) con un cierto desplazamiento o *stride* (habitualmente 2), conservando únicamente la información más relevante (como el valor máximo en cada región en el caso de *max pooling*) y disminuyendo así la carga computacional.

Las CNN aprenden representaciones jerárquicas: en las primeras capas se detectan patrones simples como bordes o texturas, mientras que en las capas más profundas se combinan esas representaciones para identificar estructuras más complejas y específicas de la tarea. Para lograr esto, las operaciones de convolución y agrupamiento son fundamentales. Como ejemplo ilustrativo, en la Figura 2 se muestra cómo una CNN puede utilizar un filtro convolucional para detectar bordes verticales en una imagen en escala de grises. Este tipo de operación permite resaltar zonas donde se producen cambios bruscos de intensidad.

Figura 2: Ejemplo de operación de convolución con un filtro de detección de bordes (derecha) aplicado a una imagen original (izquierda).

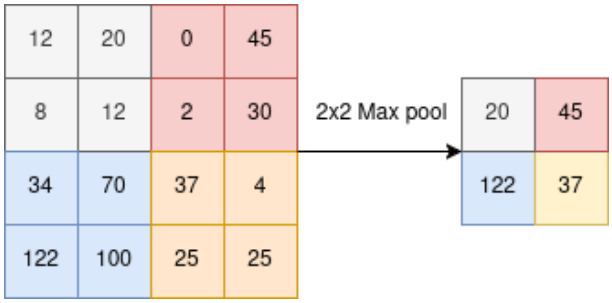


Figura 3: Aplicación de la operación de agrupamiento en el caso de *max pooling* con un filtro de tamaño 2×2 y un *stride* de 2. A cada paso el filtro se mueve 2 posiciones en horizontal y vertical, calculando el valor máximo, lo que resulta en una matriz de dimensiones 2×2.

La operación de agrupamiento, por su parte, tiene como objetivo principal disminuir el tamaño espacial (alto y ancho) de los mapas de activación (resultado obtenido tras la convolución), conservando la información más relevante. La Figura 3 muestra la aplicación de la operación de agrupamiento, en este caso aplicando *max pooling* sobre una matriz de entrada 4×4.

3.2. Segmentación semántica

El uso más habitual de las CNN es la clasificación de imágenes, donde se asigna una única etiqueta a toda la imagen de entrada. Sin embargo, en muchas aplicaciones de visión por computador, especialmente en el ámbito médico, se requiere un nivel de detalle mucho

Figura 4: Ejemplo de segmentación semántica sobre una imagen con gaviotas. De izquierda a derecha: imagen original, máscara generada a partir del contorno de las gaviotas y resultado final con superposición en color.

mayor. En estos casos, no basta con conocer la presencia de una estructura, sino que es necesario localizarla con precisión. La segmentación semántica es la tarea de asignar a cada píxel una etiqueta de clase, es decir, particionar la imagen en subconjuntos mutuamente excluyentes, donde cada subconjunto delimita una región de interés de la imagen original [30].

A diferencia de la detección de objetos —que emplea cajas delimitadoras— o la segmentación de instancias —que diferencia entre objetos individuales de una misma clase—, la segmentación semántica no distingue entre instancias, sino que clasifica cada píxel según la clase a la que pertenece. En la Figura 4 se puede observar un ejemplo de segmentación semántica.

Este tipo de segmentación resulta esencial en tareas como la identificación de órganos, tejidos o lesiones patológicas en imágenes médicas, donde la precisión espacial es crítica para el diagnóstico y seguimiento clínico. En el caso particular de la esclerosis múltiple, la segmentación semántica permite detectar y cuantificar las lesiones de sustancia blanca presentes en la resonancia magnética, lo que puede facilitar el diagnóstico temprano y la evaluación de la progresión de la enfermedad. La Figura 5 muestra un ejemplo de segmentación semántica con la red U-Net sobre la imagen MRI de un paciente con EM.

3.3. U-Net

U-Net es una arquitectura de CNN diseñada específicamente para tareas de segmentación semántica en imágenes biomédicas [31]. A diferencia de los enfoques tradicionales basados en ventanas deslizantes, U-Net permite realizar predicciones a nivel de píxel sobre imágenes completas, ofreciendo alta precisión y eficiencia incluso con un número reducido

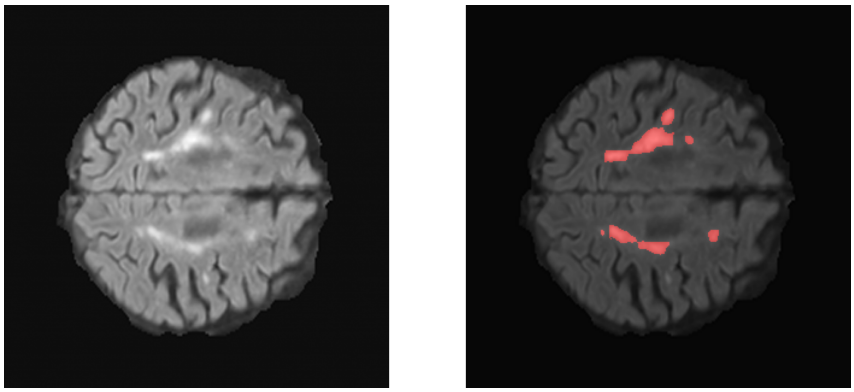


Figura 5: Ejemplo de segmentación semántica en una imagen de resonancia magnética cerebral. A la izquierda se muestra la imagen original en escala de grises. A la derecha, la misma imagen con la máscara de predicción superpuesta en color rojo, indicando las regiones identificadas automáticamente como posibles lesiones. La segmentación semántica permite clasificar cada píxel en función de su pertenencia a una categoría clínicamente relevante, facilitando el análisis estructural y la toma de decisiones diagnósticas.

de imágenes anotadas.

La arquitectura presenta una forma de *U*, compuesta por un camino de contracción (*contracting path*) y uno de expansión (*expanding path*). El primero actúa como codificador, extrayendo características a distintas escalas mediante convoluciones seguidas de operaciones de *max pooling*. El segundo realiza una decodificación progresiva mediante capas de *up-convolution*, fusionando la información de baja resolución con las características de alta resolución obtenidas durante la fase de contracción. Esta estructura simétrica permite al modelo capturar tanto el contexto global como los detalles locales, esenciales para segmentar estructuras pequeñas o con bordes imprecisos.

En la Figura 6 se muestra la arquitectura de la U-Net, compuesta por un camino de contracción (izquierda) y otro de expansión simétrico (derecha). Cada bloque azul representa un mapa de características multicanal, con el número de canales indicado en la parte superior. Las dimensiones están indicadas en la parte inferior izquierda de cada bloque. Las operaciones incluyen convoluciones 3×3 con activación ReLU (azul oscuro), max pooling 2×2 (flechas rojas), up-convoluciones 2×2 (flechas verdes), y convoluciones 1×1 al final (verde azulado). Las flechas grises horizontales indican las conexiones por copia y recorte entre el encoder y el decoder, que permiten recuperar la información

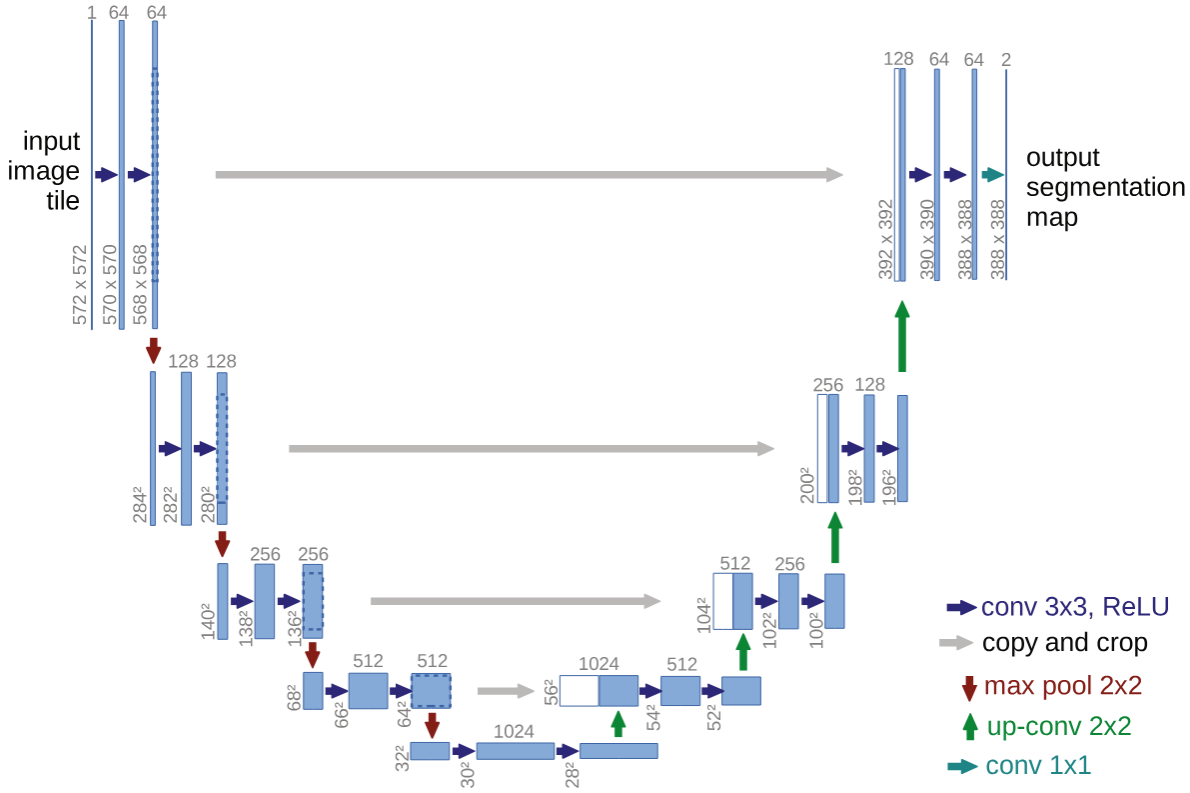


Figura 6: Arquitectura de la red U-Net [31].

espacial de alta resolución.

3.4. YOLO

YOLO es un enfoque innovador para la detección de objetos que reformula el problema como una tarea de regresión directa, eliminando la necesidad de proponer regiones o aplicar clasificadores múltiples por separado. El modelo procesa toda la imagen de entrada con una única red neuronal convolucional, que predice simultáneamente las coordenadas de las cajas delimitadoras y las probabilidades de clase asociadas [32].

La imagen se divide en una cuadrícula de tamaño fijo, donde cada celda es responsable de predecir un número determinado de cajas delimitadoras (B) para los objetos cuyo centro caiga dentro de ella. Cada una de estas cajas se acompaña de una puntuación de confianza, calculada como el producto de la probabilidad de que haya un objeto y la coincidencia con el objeto real (IoU). Además, se predicen las probabilidades de clase para cada celda. La arquitectura general de YOLO, mostrada en la Figura 7, puede dividirse en tres bloques principales:

1. **Entrada**: una imagen de tamaño fijo es introducida en la red.

2. **Extracción de características**: la imagen atraviesa varias capas convolucionales y de pooling que permiten detectar patrones visuales progresivamente más complejos, desde bordes y texturas hasta formas completas.

3. **Predicción**: las características extraídas alimentan una capa final que divide la imagen en una cuadrícula y predice, por cada celda, las cajas delimitadoras, sus puntuaciones de confianza y las probabilidades de clase.

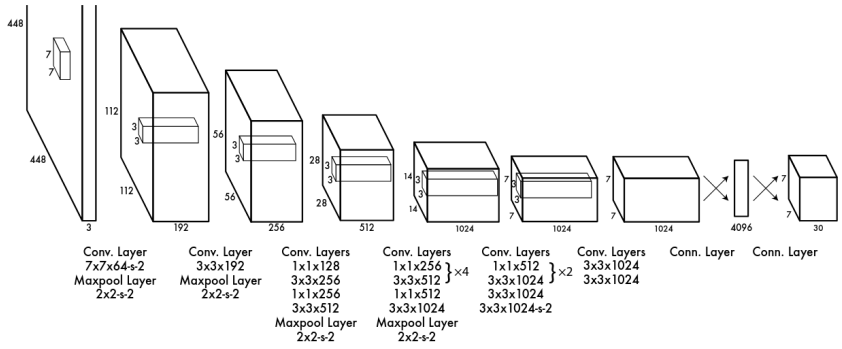


Figura 7: Arquitectura general de YOLO. El modelo divide la imagen en una cuadrícula y predice simultáneamente cajas delimitadoras, puntuaciones de confianza y clases en una sola pasada, permitiendo detección en tiempo real [4].

Este enfoque unificado permite que YOLO logre detección de objetos en tiempo real con un buen equilibrio entre velocidad y precisión, lo que lo convierte en una arquitectura popular también en tareas de segmentación por detección, especialmente en contextos donde la eficiencia computacional es crítica.

3.5. Super-resolución con FSRCNN

La super-resolución es una técnica de procesamiento de imágenes cuyo objetivo es reconstruir una imagen de alta resolución (*High Resolution* o HR, por sus siglas en inglés) a partir de una o varias imágenes de baja resolución (*Low Resolution* o LR) [24]. En el contexto de imágenes médicas, esta tarea adquiere una gran relevancia, ya que una mayor resolución espacial puede mejorar la visualización de detalles anatómicos sutiles o pequeñas lesiones que podrían pasar desapercibidas. Tradicionalmente, la super-resolución se abordaba mediante métodos cómo la interpolación u otros ya mencionados, pero en los últimos años han cobrado protagonismo obteniendo mejores resultados los enfoques basados en aprendizaje profundo [33]. Estos modelos aprenden a mapear directamente las imágenes LR a HR mediante redes CNN.

Entre los modelos más representativos de super-resolución se encuentra FSRCNN (Fast Super-Resolution Convolutional Neural Network), una arquitectura propuesta por Dong et al.[34] (véase Figura 8), que mejora significativamente tanto la precisión como la velocidad de su predecesor, SRCNN (Super-Resolution Convolutional Neural Network).

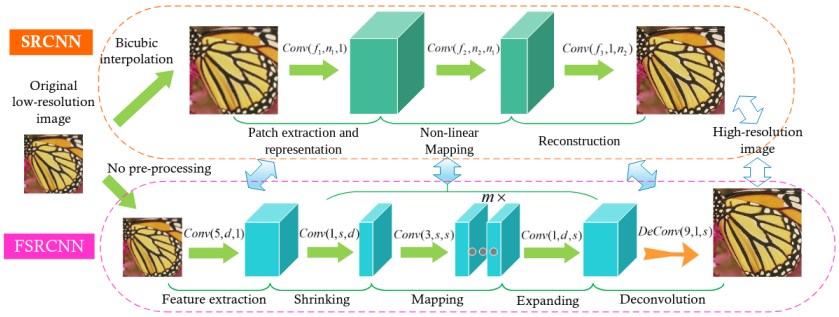


Figura 8: Comparación entre las arquitecturas de SRCNN y FSRCNN. SRCNN realiza una interpolación bicúbica previa y luego aplica tres capas convolucionales para extraer características y reconstruir la imagen. FSRCNN, por su parte, elimina la necesidad de preprocesamiento mediante interpolación y emplea una arquitectura optimizada con etapas de extracción, reducción y expansión de características, seguidas de una deconvolución final. Esta reorganización mejora tanto la precisión como la eficiencia computacional en tareas de super-resolución [34].

SRCNN realiza primero una interpolación bicúbica para escalar la imagen a la resolución deseada antes de procesarla. A continuación, aplica tres capas convolucionales que se encargan de extraer características locales, mapearlas de forma no lineal y reconstruir la imagen mejorada. Este enfoque presenta limitaciones en eficiencia y calidad debido a la interpolación inicial. FSRCNN, en cambio, introduce varias mejoras clave: elimina la necesidad de interpolación previa y trabaja directamente con la imagen en baja resolución. Su arquitectura incluye módulos para extraer características, reducir y expandir la dimensionalidad, aplicar múltiples capas de mapeo no lineal y, finalmente, realizar una deconvolución para generar la imagen en alta resolución. Este diseño permite obtener resultados más rápidos y precisos, mejorando tanto el rendimiento como la calidad final de la reconstrucción.

4

Metodología

En este capítulo se describe con detalle la metodología seguida para el desarrollo del sistema de segmentación automática de lesiones de esclerosis múltiple. Primero, se presenta el conjunto de datos utilizado y el proceso de preprocesado necesario para convertir las imágenes volumétricas originales en cortes bidimensionales adecuados para el entrenamiento de redes neuronales. A continuación, se explican los criterios de división del conjunto de datos, incluyendo la adopción de una estrategia de validación cruzada para una evaluación más robusta.

Seguidamente, se describe el flujo completo de preprocesado, desde la lectura de las imágenes hasta su adaptación al formato requerido por cada modelo. Posteriormente, se detallan las distintas estrategias de selección de imágenes evaluadas en el trabajo, diseñadas para estudiar el impacto que tiene la composición del conjunto de entrenamiento en el rendimiento del sistema.

Finalmente, se enumeran y definen las métricas utilizadas para cuantificar la calidad de las segmentaciones obtenidas, abarcando tanto medidas de solapamiento espacial como de rendimiento computacional. Esta metodología proporciona un marco experimental sólido para comparar las diferentes configuraciones propuestas a lo largo del estudio.

4.1. Enfoque experimental

El objetivo principal es desarrollar un sistema automático capaz de segmentar de forma precisa las lesiones provocadas por la esclerosis múltiple en imágenes de resonancia magnética cerebral. Este sistema busca facilitar la labor de diagnóstico y seguimiento de pacientes, reduciendo la carga de trabajo de los profesionales sanitarios y aumentando la objetividad del proceso.

Para abordar este reto, se ha planteado una metodología experimental basada en la comparación de múltiples configuraciones, que combinan diferentes arquitecturas de red, estrategias de selección de imágenes y técnicas de mejora de resolución. En concreto, se han evaluado los siguientes factores:

- **Arquitectura:** se han utilizado dos modelos de segmentación, U-Net y YOLO.
- **Selección de slices:** se han diseñado tres estrategias con el fin de comparar las redes.
- **Super-resolución:** se ha evaluado el impacto de aplicar un modelo FSRCNN para duplicar la resolución espacial de las imágenes antes del entrenamiento.

La combinación de estas variantes da lugar a un conjunto de diez experimentos distintos, que se enumeran en la Tabla 1. **Lesión** se refiere a la inclusión exclusiva de cortes con una región lesionada, **Cerebro** selecciona cortes con o sin lesion pero no vacíos, $\times 2$ indica la aplicación de super-resolución. Cada configuración ha sido evaluada utilizando un conjunto de métricas estándares para segmentación médica, con el objetivo de identificar cuál de ellas ofrece el mejor equilibrio entre precisión y robustez.

En las siguientes secciones se describen con mayor detalle el conjunto de datos utilizado, los procesos de preprocesamiento y selección de imágenes, así como las métricas empleadas para la evaluación de los modelos.

4.2. Conjunto de datos y preprocesado

El conjunto de datos utilizado es el *MSLesSeg-Dataset*, proporcionado en el contexto de la competición internacional *ICPR 2024* (*International Conference on Pattern Recognition*). Estos datos han sido diseñados específicamente para avanzar en el desarrollo de sistemas completamente automáticos de segmentación de lesiones de esclerosis múltiple, y destaca por su amplitud, calidad de anotaciones y realismo clínico. El conjunto está compuesto por imágenes de resonancia magnética de 75 pacientes reales diagnosticados con EM, obtenidas en condiciones clínicas habituales. Cada paciente cuenta con entre uno y cinco puntos temporales de seguimiento. Para cada uno de estos puntos temporales, se incluyen tres modalidades de imagen: T1-weighted (T1-w), T2-weighted (T2-w) y

Tabla 1: Resumen de las configuraciones experimentales evaluadas.

ID	Red	Selección de Slices	Super-resolución
A	U-Net	Base	Sin SR
B	YOLO	Base	x2
C	U-Net	Base	x2
D	YOLO	Base	Sin SR
E	U-Net	Lesión	Sin SR
F	U-Net	Lesión	x2
G	YOLO	Lesión	Sin SR
H	YOLO	Lesión	x2
I	U-Net	Cerebro	Sin SR
J	U-Net	Cerebro	x2
K	YOLO	Cerebro	Sin SR
L	YOLO	Cerebro	x2

FLAIR. Las máscaras de segmentación de las lesiones han sido generadas manualmente por expertos a partir de las secuencias FLAIR, que son las más sensibles para la detección de lesiones de EM. Las modalidades T1-w y T2-w también se proporcionan para favorecer una caracterización multimodal más completa de las lesiones. De los 75 pacientes, hay 22 pacientes que no disponen de máscaras de segmentación. Por lo tanto, se han descartado todas sus imágenes por completo. El resto de pacientes, 53, sí disponen de máscaras de segmentación y sus imágenes han sido usadas en este trabajo.

En cuanto al formato de las imágenes, están en formato NIfTI (*Neuroimaging Informatics Technology Initiative*), un estándar ampliamente utilizado en neuroimagen médica para almacenar datos volumétricos tridimensionales. En el caso de las imágenes de *MSLesSeg-Dataset*, presentan unas dimensiones $182 \times 218 \times 182$ en los ejes X, Y, y Z, respectivamente. Los valores de intensidad están almacenados en coma flotante (*float64*). Este formato almacena un volumen tridimensional de intensidades, habitualmente compuesto por cortes adquiridos en un plano específico (ej. axial), Figura 9.

En este trabajo se ha optado por utilizar cortes axiales (XY) como base para el pre-

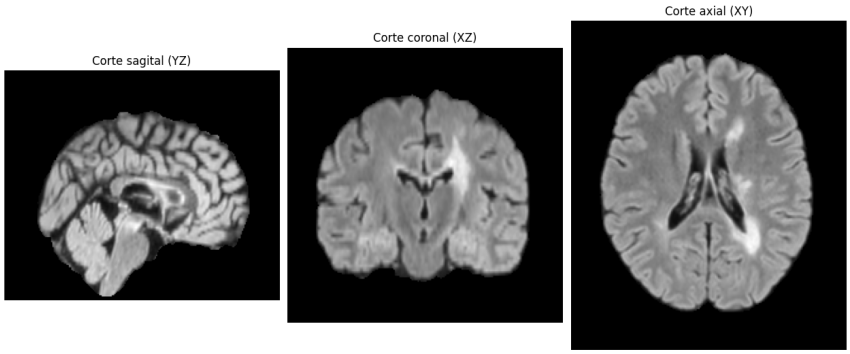


Figura 9: Visualización de los tres planos anatómicos principales a partir de un volumen FLAIR en formato NIfTI. De izquierda a derecha: corte sagital (vista lateral del cerebro), corte coronal (vista frontal) y corte axial (vista desde arriba). Cada imagen corresponde a un corte central extraído a lo largo del eje correspondiente del volumen tridimensional.

procesado y entrenamiento de los modelos. Esta elección se debe a que los volúmenes del dataset original están organizados naturalmente en ese plano, lo que significa que cada corte axial corresponde a un corte adquirido directamente por el escáner de resonancia magnética. Trabajar en este plano garantiza que los modelos trabajen sobre datos que reflejan la resolución y geometría originales del estudio, evitando interpolaciones innecesarias en los ejes reconstruidos. Además, Como el objetivo es predecir las regiones lesionadas según las máscaras de referencia, y estas han sido definidas visualmente sobre la secuencia FLAIR (donde las lesiones se ven claramente), se ha optado por entrenar el modelo únicamente con las imágenes FLAIR. Esto asegura que el modelo aprenda a segmentar a partir de la fuente de información más directa y coherente con las anotaciones manuales.

4.3. División del conjunto de datos

En una fase inicial del trabajo, se optó por una división clásica del conjunto de datos, compuesto por 53 pacientes, en tres subconjuntos independientes: entrenamiento (*train*), validación (*val*) y prueba (*test*). Esta estructura permitió entrenar los modelos con un conjunto de datos amplio, ajustar los hiperparámetros utilizando el conjunto de validación y evaluar el rendimiento final sobre un conjunto de prueba completamente separado. Esta estrategia es habitual en muchos estudios y proporciona una línea base de evaluación clara

y comprensible.

No obstante, considerando el número limitado de muestras disponibles (en especial dado que el conjunto de datos está formado por pacientes individuales, con un riesgo inherente de sobreajuste intrapaciente), se decidió posteriormente ampliar la evaluación mediante la incorporación de una estrategia de validación cruzada (`K-Fold Cross-Validation`). Esta técnica permite una estimación más robusta del rendimiento general del modelo, ya que cada muestra del conjunto de datos participa tanto en el entrenamiento como en la validación en diferentes iteraciones (*folds*).

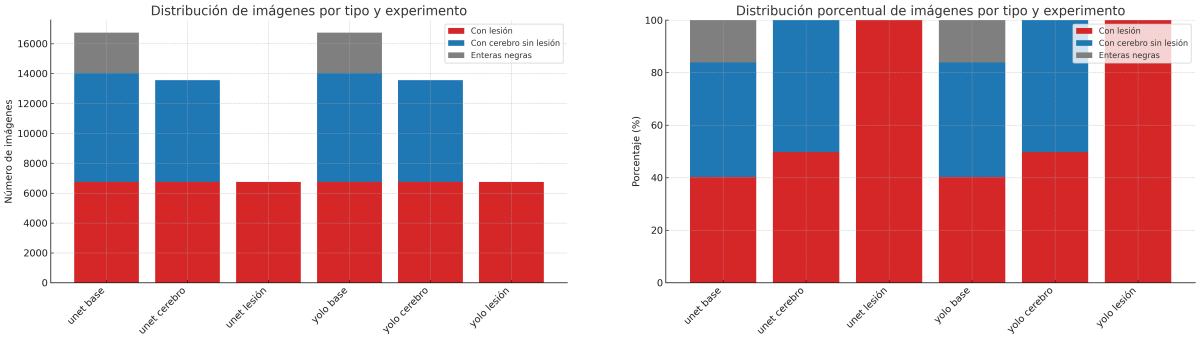
Concretamente, se empleó una validación cruzada estratificada con $K = 5$, asegurando en cada *fold* que no hubiera solapamiento de pacientes entre los conjuntos de entrenamiento y validación, lo cual es fundamental para evitar sesgos derivados de la correlación intrapaciente en las imágenes. A continuación, se resumen los tamaños de cada conjunto según la estrategia de selección de cortes utilizada:

- **Base**: Se incluyeron todos los cortes disponibles. El número total de imágenes fue de 16744 (cuya distribución es de 9994 cortes vacíos y 6750 con lesión).
- **Base con superposición (x2)**: Mismo conjunto que el anterior, pero aplicando un modelo de super-resolución FSRCNN para duplicar la resolución espacial de las imágenes FLAIR. El número total de imágenes fue de 16744.
- **Lesión**: Se seleccionaron únicamente los cortes cuya máscara presentaba lesión. Esta configuración resultó en 6750 cortes.
- **Lesión con superresolución (x2)**: Misma selección que la anterior, aplicando además un aumento de resolución mediante un modelo FSRCNN.
- **Cerebro**: Se seleccionaron aquellos cortes que contuvieran cerebro. Es decir, se descartaron los cortes que no presentaban ninguna estructura cerebral visible aunque puedan no tener ninguna lesión. Esta selección resultó en 13563 cortes.
- **Cerebro con superresolución (x2)**: Misma selección anterior con aumento de resolución, manteniendo el mismo número de imágenes.

Se pueden distinguir varios tipos de cortes en función de su contenido:

- **Cortes vacíos:** aquellos que no contienen ninguna estructura cerebral visible, ni lesión ni cerebro. Estos cortes no aportan información relevante para la segmentación y pueden introducir ruido en el entrenamiento del modelo.
- **Cortes con lesión:** aquellos que presentan al menos una región lesionada, independientemente de si también contienen otras estructuras cerebrales. Estos cortes son fundamentales para entrenar modelos de segmentación, ya que proporcionan ejemplos positivos de la clase objetivo.
- **Cortes con cerebro:** aquellos que contienen al menos una estructura cerebral visible, ya sea con o sin lesión. Estos cortes son útiles para entrenar modelos que deben aprender a segmentar el cerebro en general, no solo las lesiones.

A continuación se muestran dos figuras que ilustran la distribución de estos cortes.



(a) Distribución absoluta de imágenes por experimento

(b) Distribución porcentual de imágenes por experimento

Figura 10: Resumen de la distribución de imágenes por experimento, diferenciando entre imágenes con lesión, con cerebro sin lesión y enteras negras.

Se aplicó una **validación cruzada estratificada por paciente** utilizando la clase `GroupKFold` de la biblioteca `scikit-learn`, la cual garantiza que todos los cortes pertenecientes a un mismo paciente estén en un único subconjunto por *fold*. Esto evita fugas de información y asegura que la distribución de pacientes sea equilibrada entre los distintos pliegues. Los resultados obtenidos en cada *fold* fueron agregados mediante la media y desviación estándar de las métricas descritas al final de este capítulo, proporcionando una estimación más robusta y representativa del rendimiento de los modelos entrenados.

4.4. Flujo de preprocesado

Seleccionados el tipo de corte y la modalidad de las imágenes, el siguiente paso es describir el flujo de preprocesado, que tiene como objetivo generar conjuntos de datos 2D a partir de las imágenes originales en formato NIfTI adaptando el formato de salida según el tipo de red neuronal empleada. El flujo general se describe a continuación:

1. **Lectura de imágenes:** se inicia con la lectura de los volúmenes de imagen y sus correspondientes máscaras de segmentación.

2. **División del conjunto de datos:** los pacientes se dividen en tres subconjuntos mutuamente excluyentes: entrenamiento, validación y prueba. Esta división garantiza que no exista solapamiento de datos entre fases, evitando así correlaciones intrapaciente que puedan sesgar los resultados.

3. **Selección de slices:** de cada volumen se extraen cortes axiales bidimensionales. La estrategia de selección puede variar según las descritas anteriormente.

4. **Decisión sobre la aplicación de super-resolución (SR):** en este punto, se determina si se aplicará o no un modelo de super-resolución. Esta decisión depende del experimento configurado.

    - **Si no se aplica SR:** las imágenes y máscaras continúan directamente hacia el proceso de normalización.
    - **Si se aplica SR:** se aplica el modelo FSRCNN únicamente a las imágenes FLAIR, con el objetivo de mejorar su resolución espacial. Las máscaras, por su naturaleza binaria o categórica, se reescalan mediante interpolación para evitar artefactos y preservar las clases originales.

5. **Normalización:** las imágenes FLAIR se normalizan a un rango de intensidades estándar $[0, 255]$, y se redimensionan del tamaño original (182×218) a un tamaño uniforme (340×340 píxeles), adaptándose a los requisitos de entrada de las redes neuronales.

6. **Binarización:** en el caso de las máscaras, el rango de intensidades se convierte a 0 (fondo negro) o 255 (lesión con píxeles blancos).

7. **Guardado:** finalmente, tanto las imágenes como sus máscaras procesadas se almacenan en directorios separados según su propósito (entrenamiento, validación o prueba). En el caso de YOLO, las máscaras también se convierten a formato de anotación compatible (*bounding boxes*) en archivos de texto.

4.5. Estrategia de selección de imágenes

Para maximizar el aprovechamiento de los datos y analizar el impacto de distintas formas de selección de imágenes, se han definido tres estrategias distintas para la extracción de cortes axiales a partir de los volúmenes tridimensionales:

- **Base:** se incluyen todos los cortes axiales de cada volumen, sin aplicar ningún filtrado. Esta estrategia sirve como línea base para comparar el rendimiento de modelos entrenados con el conjunto completo de datos disponibles, incluyendo cortes sin lesión visible.

- **Lesion:** se seleccionan únicamente aquellos cortes cuya máscara contiene lesión. Esta estrategia elimina los slices vacíos que no aportan información relevante para la tarea de segmentación, reduciendo el ruido de fondo y el desbalance entre clases.

- **Cerebro:** se seleccionan cortes que contengan al menos una estructura cerebral visible, independientemente de si presentan lesión o no. Esta estrategia busca entrenar modelos que aprendan a segmentar el cerebro en general, lo cual es útil para tareas de segmentación más amplias que van más allá de las lesiones específicas.

Estas estrategias se aplican de forma programática durante el preprocesado, asegurando que cada imagen FLAIR extraída se acompaña de su correspondiente máscara binaria de segmentación. La comparación del rendimiento de los modelos entrenados con cada estrategia permite evaluar el impacto que tiene el tipo de datos suministrado sobre la capacidad del modelo para detectar y segmentar correctamente las lesiones de esclerosis múltiple.

4.6. Métricas utilizadas

Para evaluar el rendimiento de los modelos de segmentación entrenados en este trabajo, se han empleado métricas clásicas de segmentación binaria ampliamente utilizadas en el ámbito de la segmentación médica. Estas métricas permiten cuantificar tanto la precisión general del modelo como su comportamiento específico al identificar regiones lesionadas, que en este caso constituyen la clase positiva. Las métricas utilizadas son las siguientes:

- **Índice de Jaccard (IoU):** también conocido como *Intersection over Union*, mide la superposición entre la predicción y la máscara real. Se define como:

$$IoU = \frac{TP}{TP + FP + FN}$$

  donde TP son los píxeles correctamente clasificados como lesión, FP son los falsos positivos y FN los falsos negativos.

- **Coeficiente de Dice (Dice Score):** métrica similar al IoU pero más sensible a clases desbalanceadas, ampliamente utilizada en segmentación médica:

$$Dice = \frac{2TP}{2TP + FP + FN}$$

  En este trabajo, también se ha utilizado como métrica principal de referencia.

- **Precisión (Precision):** mide la proporción de verdaderos positivos sobre el total de predicciones positivas:

$$Precision = \frac{TP}{TP + FP}$$

  Es útil para valorar la exactitud del modelo evitando predicciones incorrectas.

- **Sensibilidad (Recall):** mide la capacidad del modelo para detectar correctamente las regiones lesionadas reales:

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

- **Tiempo de inferencia:** se ha medido el tiempo medio de predicción por imagen, en segundos, con el objetivo de valorar la viabilidad del modelo en entornos clínicos reales. Esta métrica no afecta directamente a la calidad de la segmentación, pero es crítica para su aplicabilidad práctica.

El conjunto de métricas seleccionadas permite una evaluación completa del sistema, considerando tanto su capacidad de detectar correctamente las lesiones como de evitar falsos positivos, además de su eficiencia computacional.

4.7. Entrenamiento

El proceso de entrenamiento de los modelos de segmentación se ha llevado a cabo adaptando la arquitectura y configuración a las características específicas del problema. En el caso de U-Net, se ha utilizado un codificador basado en ResNet34 preentrenado con el conjunto de imágenes *ImageNet*, lo que permite transferir conocimiento aprendido en tareas generales de visión por computador y acelerar la convergencia del modelo. La entrada a la red consiste en imágenes unicanal correspondientes a cortes axiales de resonancia magnética FLAIR, mientras que la salida es un mapa de segmentación con activación lineal, sobre el cual se aplica una función sigmoide en la fase de evaluación.

La función de pérdida utilizada combina dos componentes complementarios: la `Binary Cross-Entropy` (BCE), que penaliza errores píxel a píxel, y la `Dice Loss`, que favorece la superposición espacial entre predicción y máscara. La combinación se expresa como:

$$\mathcal{L}(x, y) = \text{BCEWithLogitsLoss}(x, y) + \left(1 - \frac{2\sum_i \sigma(x_i) y_i + \varepsilon}{\sum_i \sigma(x_i) + \sum_i y_i + \varepsilon}\right) \tag{2}$$

donde x representa la salida cruda del modelo (logits), $\sigma(x)$ es la función sigmoide aplicada a cada valor de salida, y es la máscara binaria de referencia, y $\varepsilon = 10^{-5}$ es un término de suavizado para evitar inestabilidades numéricas. La optimización del modelo se realiza mediante el algoritmo `Adam`, ampliamente utilizado por su eficiencia y capacidad de adaptación a diferentes escalas de gradientes. Para mejorar la capacidad generalizadora del modelo, se ha incorporado un plan de ajuste dinámico de la tasa de aprendizaje mediante la política `ReduceLROnPlateau`, que reduce el valor del *learning rate* cuando la pérdida de validación no mejora durante varias épocas consecutivas.

El entrenamiento de YOLO se realiza mediante la combinación de varias funciones de pérdida: localización (coordenadas de *bounding boxes*), clasificación de clase y segmentación. Por otra parte, se ha procurado mantener la coherencia en los principales hiperparámetros de entrenamiento entre ambos modelos con el fin de garantizar una

comparación equitativa y controlada. En particular, se ha fijado una tasa de aprendizaje inicial de 0,001, un tamaño de batch de 16 y un total de 50 épocas de entrenamiento tanto para U-Net como para YOLO. Este ajuste ha sido necesario para igualar las condiciones experimentales y observar el impacto de la arquitectura y el preprocesado, minimizando el efecto de otros factores externos. La Tabla 2 resume los principales hiperparámetros utilizados en ambos casos.

Tabla 2: Comparación de hiperparámetros entre U-Net y YOLO.

Parámetro	**U-Net**	**YOLO**
Tamaño de batch	16	16
Épocas	50	50
Tasa de aprendizaje	0.001	0.01
Optimizador	Adam	SGD
Scheduler	ReduceLROnPlateau	Cosine (opcional)
Función de pérdida	BCE + Dice	Box + Class + DFL
Precisión mixta (AMP)	Sí	Sí
Semilla aleatoria	42	42
Confianza	0.5	0.5
Aumento de datos	No	No

5

Resultados

En este capítulo se presentan los resultados obtenidos tras entrenar, validar y evaluar los modelos desarrollados para la segmentación de lesiones de esclerosis múltiple en imágenes de resonancia magnética. Se analizan tanto los aspectos cuantitativos —mediante métricas estándar de segmentación— como cualitativos, a través de la inspección visual de las predicciones generadas por los modelos.

Los resultados presentados en este capítulo permiten no solo validar el correcto funcionamiento de los modelos implementados, sino también identificar ventajas y limitaciones de cada enfoque en función de la calidad de las segmentaciones obtenidas.

5.1. Evaluación cuantitativa

La evaluación cuantitativa tiene como objetivo medir de forma objetiva el rendimiento de los modelos de segmentación utilizando métricas numéricas. Este tipo de evaluación permite comparar de manera rigurosa distintas configuraciones de modelos, arquitecturas y estrategias de preprocesamiento, proporcionando una base sólida para identificar cuáles ofrecen mejores resultados. A diferencia del análisis cualitativo, que se basa en la inspección visual de los resultados, la evaluación cuantitativa facilita una comparación sistemática mediante valores promedio y medidas de dispersión sobre múltiples particiones del conjunto de datos.

En el contexto de la segmentación médica, se han empleado métricas ampliamente aceptadas para cuantificar el grado de coincidencia entre las máscaras generadas por los modelos y las máscaras de referencia creadas por expertos. En concreto, se han utilizado el coeficiente de Dice, el índice de Jaccard (IoU), la precisión y la sensibilidad (recall), cuyas definiciones se recogen en la Sección Metodologías. Para garantizar una evaluación robusta, los resultados se han obtenido mediante validación cruzada con $K = 5$ folds,

reportando la media y la desviación estándar de cada métrica.

Tabla 3: Resultados medios y desviación estándar de los experimentos (validación cruzada con $K = 5$), se marcan en negrita los modelos con DICE más alto de cada configuración.

ID	Red	Selección	SR	Dice (%)	IoU (%)	Precisión (%)	Recall (%)
A	U-Net	Base	No	71.07 ± 4.39	67.28 ± 3.05	72.96 ± 5.76	72.09 ± 5.04
B	YOLO	Base	No	71.92 ± 1.58	68.81 ± 1.61	73.33 ± 1.41	72.18 ± 1.93
C	**U-Net**	**Base**	**x2**	**73.14 ± 1.64**	**68.05 ± 1.72**	**74.13 ± 2.74**	**75.32 ± 1.08**
D	**YOLO**	**Base**	**x2**	**74.60 ± 1.58**	**71.32 ± 1.49**	**76.84 ± 1.38**	**74.30 ± 1.87**
E	U-Net	Lesión	No	63.91 ± 3.66	51.25 ± 3.90	65.40 ± 4.18	70.20 ± 3.30
F	U-Net	Lesión	x2	64.68 ± 3.43	52.44 ± 3.78	68.73 ± 4.43	68.18 ± 2.62
G	YOLO	Lesión	No	31.11 ± 6.78	23.30 ± 5.40	34.33 ± 7.07	32.02 ± 6.83
H	YOLO	Lesión	x2	38.80 ± 7.07	30.34 ± 6.00	43.74 ± 6.90	38.62 ± 7.34
I	U-Net	Cerebro	No	67.90 ± 4.12	61.76 ± 4.49	69.06 ± 4.14	70.56 ± 3.93
J	U-Net	Cerebro	x2	72.41 ± 2.19	66.34 ± 2.40	75.01 ± 3.42	73.52 ± 1.31
K	YOLO	Cerebro	No	64.47 ± 2.20	60.78 ± 2.57	66.22 ± 1.89	64.69 ± 2.49
L	YOLO	Cerebro	x2	67.69 ± 2.58	63.72 ± 2.85	71.12 ± 1.91	66.93 ± 3.14

Los resultados presentados en la Tabla 3 muestran cómo el rendimiento de los modelos depende de la estrategia de selección de cortes y de la aplicación de superresolución.

En la estrategia **Base**, que incluye los 16.744 cortes (de los cuales 9.994 no presentan lesión y 2.703 son completamente negros), las métricas alcanzan valores altos. Esto se debe a que los cortes sin lesión, que constituyen la mayoría, se contabilizan como predicciones correctas, lo que eleva de forma notable las métricas globales. En este escenario, YOLO obtiene un mejor desempeño (B: 71.92 % Dice) que U-Net (A: 71.07 %), aunque la diferencia es reducida. La aplicación de superresolución aporta una mejora consistente en ambos casos, siendo más pronunciada en YOLO, que alcanza su mejor resultado global con **74.60 % de Dice (D)**.

La estrategia **Lesión**, que considera únicamente los 6.750 cortes con lesión, refleja un escenario mucho más exigente, ya que las métricas dependen únicamente de la correcta segmentación de regiones patológicas. Aquí los resultados caen considerablemente: U-Net

apenas alcanza 63.91 % de Dice (E), y YOLO baja hasta 31.11 % (G). Incluso aplicando superresolución, las mejoras son limitadas (F: 64.68 % en U-Net y H: 38.80 % en YOLO). Esto sugiere que, sin cortes negativos que aporten contraste, los modelos tienen más dificultades para generalizar y distinguir correctamente entre tejido sano y patológico.

La estrategia **Cerebro**, que elimina los cortes completamente negros y mantiene un equilibrio entre cortes con y sin lesión (13.563 cortes en total, de los cuales 6.815 no contienen lesión pero sí tejido cerebral visible), ofrece un compromiso intermedio. En este escenario, los modelos disponen de más información anatómica y un balance más realista. U-Net con superresolución alcanza su mejor desempeño con **72.41 % de Dice (J)**, superando claramente a su configuración Lesión y quedando muy cerca de su resultado en Base. YOLO también mejora respecto a la estrategia Lesión, alcanzando 67.69 % en su mejor configuración (L).

En resumen:

- El mejor experimento de **U-Net** corresponde a la configuración **Base con SR (C)**, con 73.14 % de Dice.
- El mejor experimento de **YOLO** es la configuración **Base con SR (D)**, con 74.60 % de Dice.

Estas diferencias se explican por la proporción de imágenes en cada estrategia. En Base, la abundancia de cortes sin lesión aumenta las métricas al contarse como correctos, mientras que en Lesión los modelos se enfrentan únicamente al reto de segmentar lesiones, obteniendo valores más bajos. La estrategia Cerebro, al mantener contexto anatómico y cierta proporción de negativos, proporciona un escenario equilibrado y clínicamente más representativo, con resultados competitivos especialmente para U-Net.

En cuanto a precisión, U-Net muestra resultados más elevados, destacando el experimento G con un 76.29 % sin necesidad de superresolución. Curiosamente, al aplicar superresolución (H), la precisión disminuye ligeramente (70.64 %), lo que podría sugerir una ligera sobresegmentación. Por otro lado, YOLO muestra valores más consistentes y moderados, con su mejor resultado en J (66.68 %). Esto sugiere que, si bien YOLO es menos preciso que U-Net en general, tiende a mantener una mayor estabilidad entre experimentos.

Para comprender mejor la correlación entre las proporciones de los cortes y las métricas obtenidas, se presentan distintas figuras que ilustran la distribución de cortes y su impacto en las métricas. La Figura 14 muestra la proporción de cortes con y sin lesión de todos los experimentos, destacando cómo la presencia de cortes negativos influye en las métricas globales.

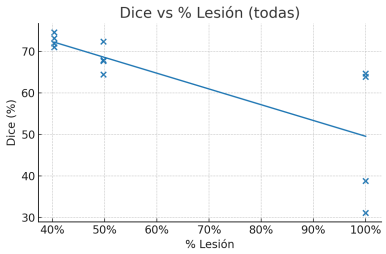


Figura 11: Relación entre Dice y los cortes con lesión.

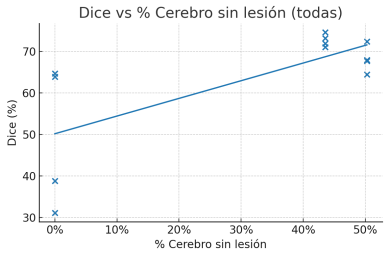


Figura 12: Relación entre Dice y los cortes sin lesión.

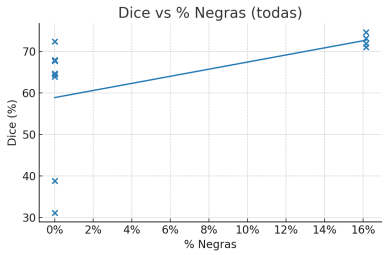
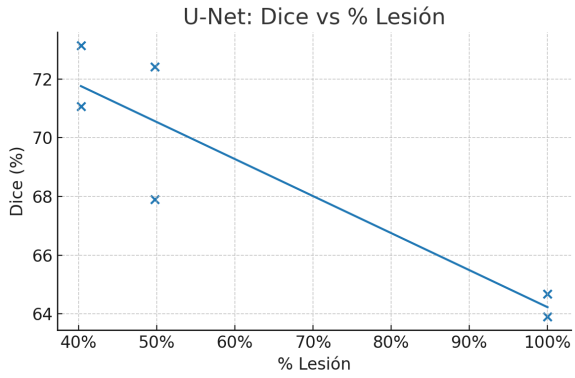


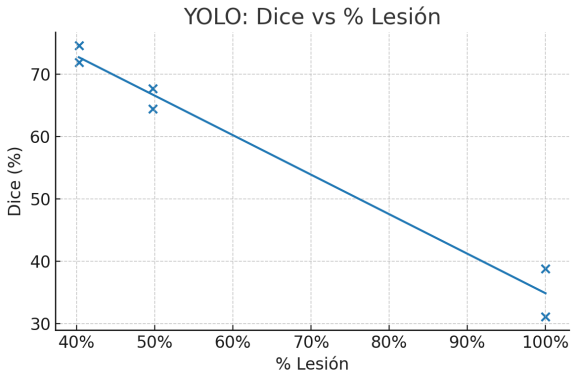
Figura 13: Relación entre Dice y los cortes negros.

Figura 14: Proporción de cortes con y sin lesión en los experimentos.

Como se observa, las redes disminuyen su rendimiento cuanto mayor proporción de lesión hay, lo que muestra que el desafío es mayor. Contrariamente, cuanta mayor proporción de imágenes negras (es decir, vacías) o sin lesión hay, aumenta el Dice, lo que significa que las redes encuentran fácil predecir cuando no hay lesión, lo que infla las métricas. Por lo tanto, la dificultad está cuanto mayor proporción de cortes con lesión hay. La Figura 15 muestra esta conclusión de forma gráfica.



(a) Correlación entre el Dice de la U-Net y la proporción de lesiones.



(b) Correlación entre el Dice de YOLO y la proporción de lesiones.

Figura 15: Resumen de la correlación entre Dice y proporción de lesiones.

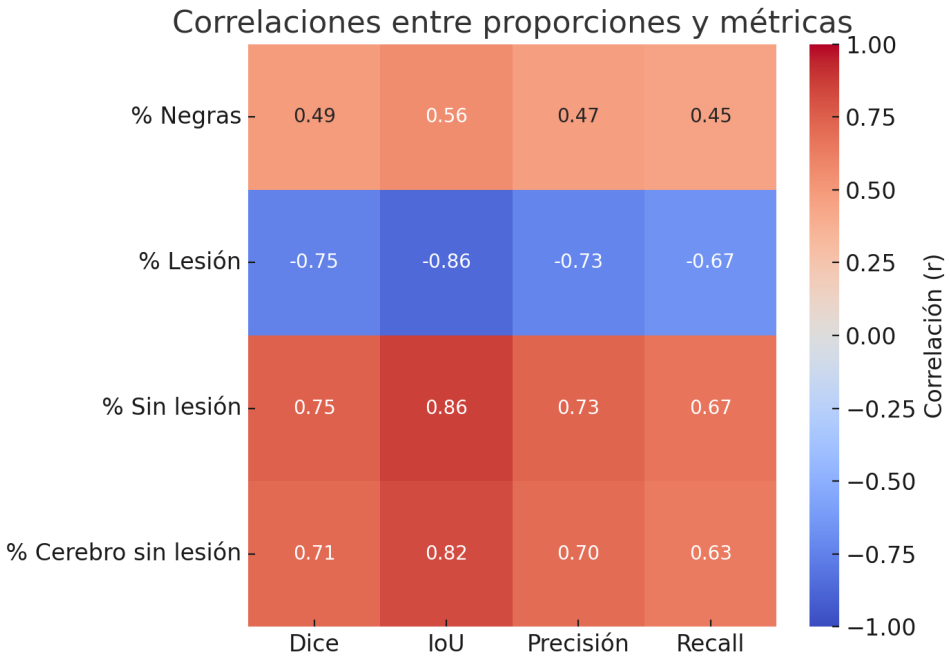


Figura 16: Mapa de calor de las correlaciones entre proporción de los cortes y métricas obtenidas.

La Figura 16 muestra la relación entre la proporción de los cortes y todas las métricas. Se observa una correlación negativa fuerte entre el porcentaje de cortes con lesión y todas las métricas (r entre –0.67 y –0.86), lo que indica que a medida que aumenta la proporción de cortes positivos el problema se vuelve más exigente y el rendimiento global disminuye. Por el contrario, tanto el porcentaje de cortes sin lesión como el de cortes con cerebro sin lesión presentan correlaciones positivas altas (r entre 0.70 y 0.86), reflejando que disponer de ejemplos negativos con información anatómica facilita al modelo aprender a discriminar y segmentar correctamente. Finalmente, el porcentaje de cortes completamente negros también muestra una correlación positiva moderada ($r \approx 0{,}45$–$0{,}56$), aunque este efecto responde en gran medida a un inflado artificial de las métricas, ya que dichos cortes son triviales de clasificar. En conjunto, el análisis confirma que el equilibrio entre cortes con y sin lesión es determinante para obtener resultados fiables, mientras que un exceso de negativos triviales puede sesgar la evaluación.

Por otra parte, aunque la aplicación de superresolución puede mejorar las métricas, es importante cuantificar su impacto computacional asociado. Tal como se muestra en la Tabla 4, el uso de superresolución incrementa los tiempos de inferencia. En el caso

de YOLO, aunque la penalización también es evidente, sus tiempos de inferencia siguen siendo considerablemente más bajos que los de U-Net, incluso cuando se aplica superresolución. Esto convierte a YOLO en una opción especialmente atractiva para escenarios donde la velocidad de procesamiento es un requisito crítico, como en entornos clínicos en tiempo real. En definitiva, los resultados muestran que existe un compromiso claro entre precisión y eficiencia, por lo que la elección de una configuración óptima dependerá del contexto de uso y las prioridades del sistema final.

Tabla 4: Tiempo medio de inferencia por imagen y desviación estándar para cada configuración calculado sobre cada fold.

ID	Red	Selección	SR	Tiempo de Inferencia (s)
A	U-Net	Base	No	$2{,}40 \times 10^{-4} \pm 7{,}84 \times 10^{-6}$
B	YOLO	Base	No	$2{,}09 \times 10^{-2} \pm 2{,}34 \times 10^{-3}$
C	U-Net	Base	x2	$2{,}41 \times 10^{-4} \pm 2{,}36 \times 10^{-6}$
D	YOLO	Base	x2	$1{,}22 \times 10^{-2} \pm 6{,}00 \times 10^{-4}$
E	U-Net	Lesión	No	$2{,}63 \times 10^{-4} \pm 1{,}23 \times 10^{-5}$
F	U-Net	Lesión	x2	$1{,}23 \times 10^{-5} \pm 2{,}68 \times 10^{-6}$
G	U-Net	Cerebro	No	$2{,}61 \times 10^{-4} \pm 1{,}24 \times 10^{-5}$
H	U-Net	Cerebro	x2	$2{,}90 \times 10^{-4} \pm 1{,}30 \times 10^{-5}$
I	YOLO	Lesión	No	$1{,}94 \times 10^{-2} \pm 1{,}70 \times 10^{-3}$
J	YOLO	Lesión	x2	$2{,}60 \times 10^{-2} \pm 1{,}85 \times 10^{-3}$
K	YOLO	Cerebro	No	$1{,}95 \times 10^{-2} \pm 2{,}10 \times 10^{-3}$
L	YOLO	Cerebro	x2	$1{,}71 \times 10^{-2} \pm 4{,}21 \times 10^{-3}$

5.2. Evaluación cualitativa

Con el objetivo de complementar la evaluación cuantitativa y analizar la calidad visual de las segmentaciones generadas por los modelos, se ha llevado a cabo un análisis cualitativo sobre una muestra representativa del conjunto de test. Esta evaluación permite observar el comportamiento de los modelos ante diferentes escenarios clínicos, como lesiones pequeñas, múltiples o con bordes difusos, aspectos que pueden no ser capturados

adecuadamente mediante métricas numéricas.

A continuación se muestran algunos ejemplos de predicción considerados clínicamente relevantes de cada modelo entrenado por experimento y por red. Empezando por la Figura 17, que muestra al paciente 1, punto temporal T1 y corte número 105. En este caso, la U-Net supera claramente a YOLO, logrando una segmentación más precisa y completa de la lesión principal, así como de las pequeñas lesiones adyacentes, aunque ha tenido un falso positivo en una pequeña lesión en la parte inferior.

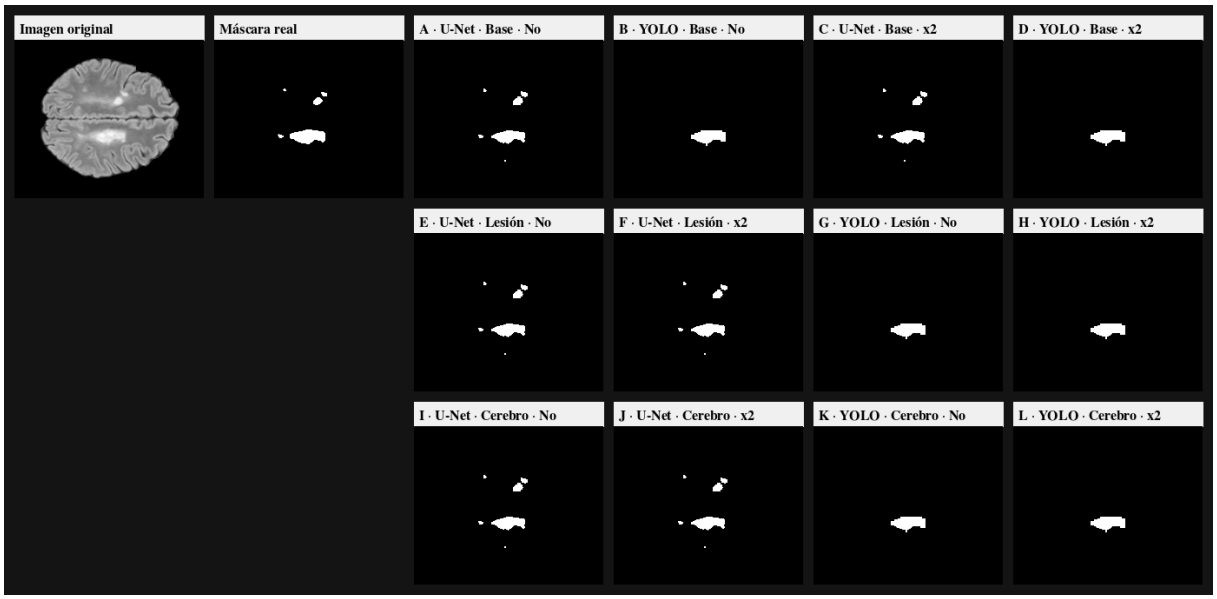


Figura 17: Lesión grande con cuatro lesiones pequeñas alrededor.

La Figura 18 corresponde a una lesión pequeña bien delimitada del paciente 12, punto temporal T4 y corte número 105. En este caso, U-Net y YOLO logran una localización precisa. Puede verse claramente la predicción por polígonos de YOLO, que capta la lesión como un área cuadrada, mientras que U-Net ofrece una segmentación más detallada y ajustada a la forma real de la lesión.

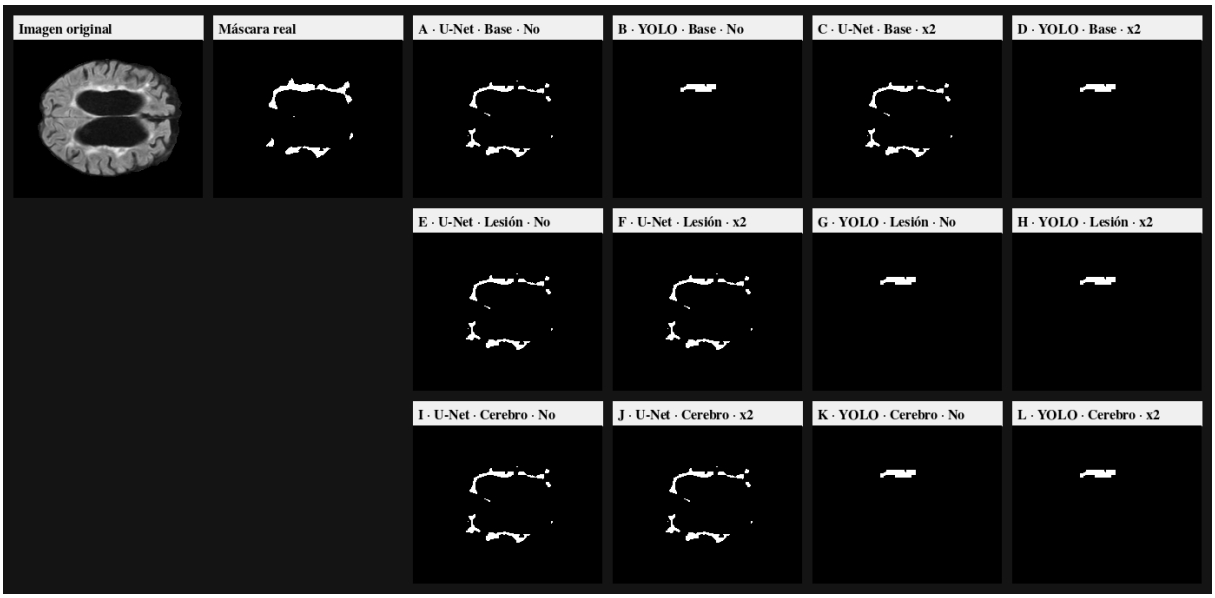


Figura 18: Lesión pequeña bien delimitada. U-Net ofrece una predicción muy precisa; YOLO comete un falso positivo en una región sin lesión.

La Figura 19 representa una lesión fragmentada con áreas pequeñas y dispersas, paciente 10, punto temporal T2 y corte número 100. En este caso, U-Net logra una segmentación ligeramente muy buena aunque no perfecto. La YOLO no logra captar ninguna lesión.

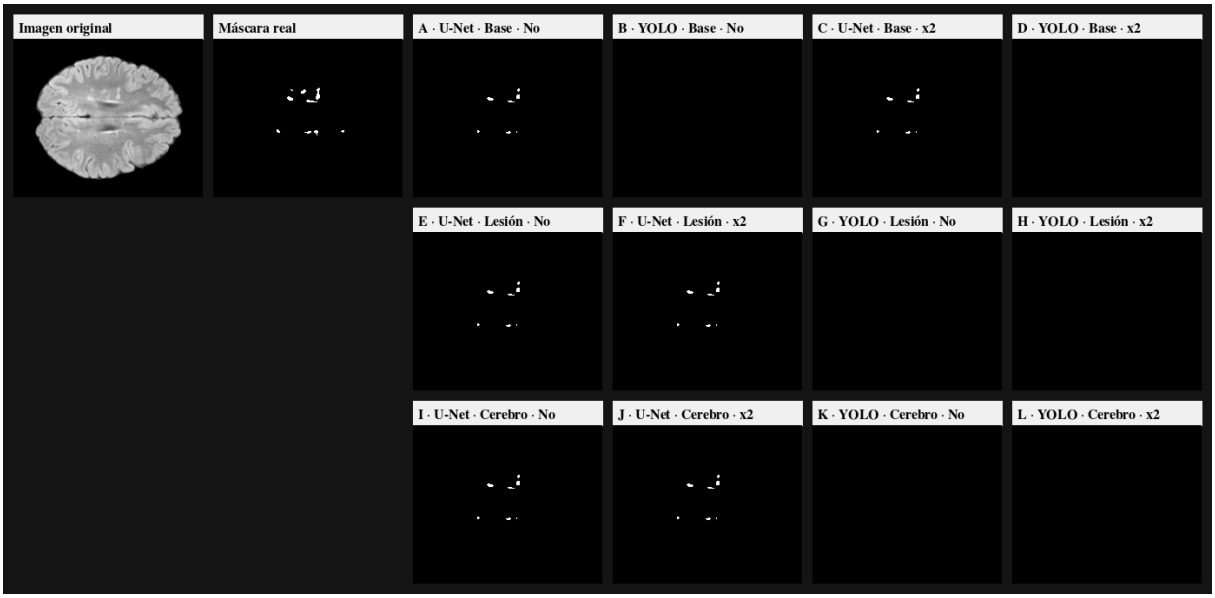


Figura 19: Lesión extensa alrededor del centro cerebral. U-Net sobresegmenta ligeramente, mientras YOLO no logra abarcar toda el área.

La Figura 20 muestra una lesión compleja repartida en áreas con regiones circulares, paciente 33, punto temporal T2 y corte número 98. La U-Net logra segmentar de manera más completa aunque con falsos positivos y ligera sobresegmentación. YOLO no es capaz de captar la lesión en su totalidad, presentando un solo área que no corresponde con la realidad.

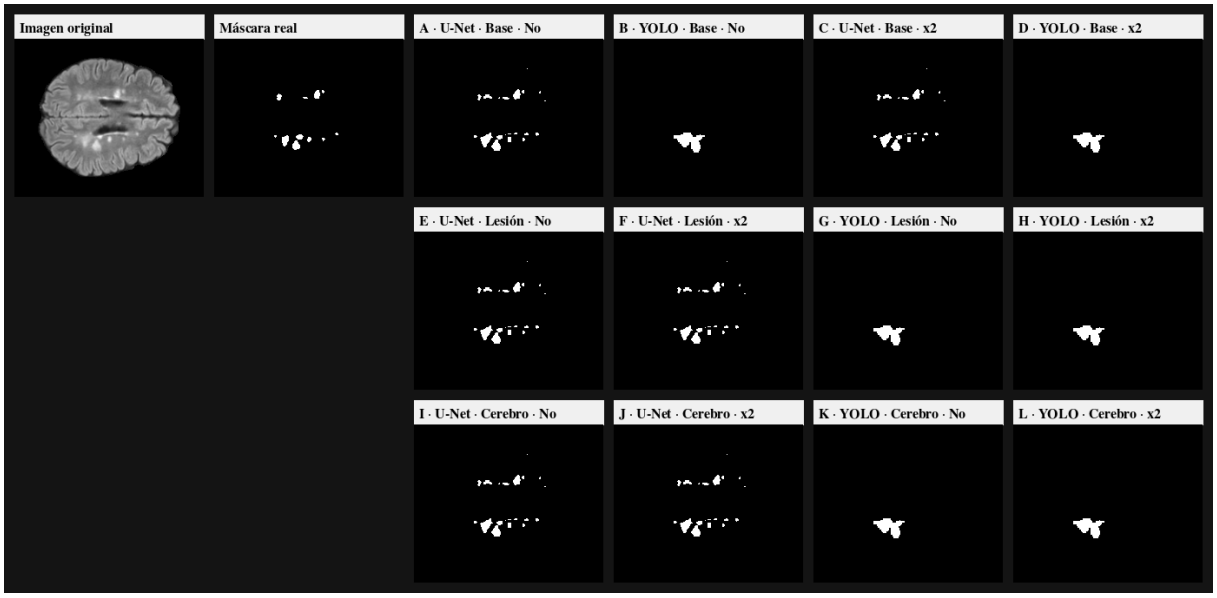


Figura 20: Lesiones fragmentadas y dispersas. YOLO solo capta un área mayor que la máscara real.

La Figura 21 representa una lesión dividida en dos partes, paciente 7, punto temporal T1 y corte número 99. En este caso, tanto U-Net como YOLO logran identificar las dos áreas lesionadas. Sin embargo, U-Net ofrece una segmentación más detallada y precisa, capturando mejor los bordes irregulares de las lesiones, mientras que YOLO presenta una segmentación más simplificada y menos ajustada.

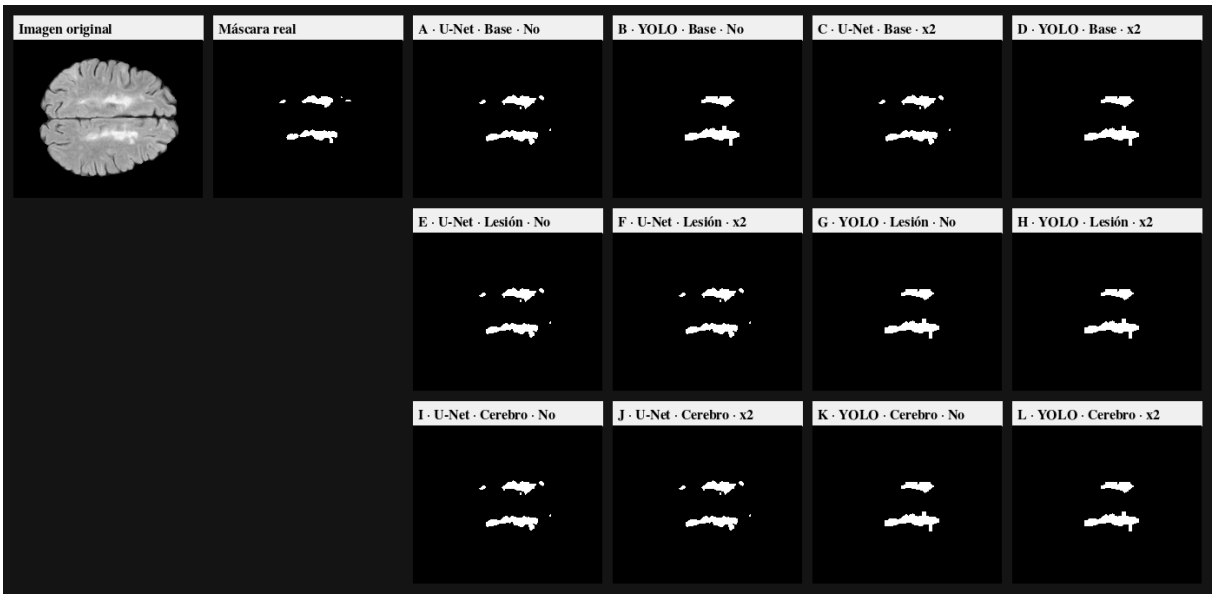


Figura 21: Lesión en dos áreas separadas. U-Net proporciona una segmentación más detallada, mientras YOLO ofrece una representación más simplificada.

6

Conclusiones y líneas futuras

En este Trabajo de Fin de Máster se ha desarrollado un sistema automático para la segmentación de lesiones de esclerosis múltiple en imágenes de resonancia magnética, explorando distintas estrategias basadas en redes neuronales profundas. El objetivo principal era comparar el rendimiento des distintas configuraciones usando las redes U-Net y YOLO en tareas de segmentación semántica, evaluando así el impacto de diferentes técnicas de preprocesamiento y selección de datos. Los resultados obtenidos permiten extraer varias conclusiones relevantes:

- La elección de los cortes a utilizar durante el entrenamiento tiene un impacto en el rendimiento de los modelos.

- La incorporación de técnicas de superresolución (FSRCNN) ha demostrado ser beneficiosa, especialmente en la arquitectura U-Net, mejorando la calidad de las segmentaciones y elevando los valores de $Dice$ e IoU sin un coste computacional prohibitivo.

- U-Net ha mostrado un mejor rendimiento global en términos de precisión espacial, alcanzando mejores resultados en métricas como Dice, IoU y precisión. Por su parte, YOLO ha destacado por su rapidez en la inferencia y su mayor sensibilidad (recall), aunque su estrategia de predicción basada en polígonos puede limitar la precisión en lesiones pequeñas o de forma irregular.

- El análisis cualitativo ha permitido observar diferencias relevantes entre ambas arquitecturas que no se pueden captar analizando exclusivamente los resultados numéricos. Mientras que U-Net proporciona segmentaciones más precisas a nivel de

píxel, YOLO tiende a agrupar regiones cercanas mediante polígonos rectangulares, lo que puede ser una limitación en contextos clínicos donde los detalles morfológicos son clave.

- En términos de eficiencia computacional, YOLO ha demostrado tiempos de inferencia menores, lo que podría ser ventajoso en escenarios donde la rapidez es prioritaria, como en sistemas de ayuda al diagnóstico en tiempo real.

A pesar de los resultados prometedores obtenidos, este trabajo presenta diversas limitaciones que conviene tener en cuenta al interpretar los hallazgos. En primer lugar, el tamaño del conjunto de datos es reducido, tanto en número de pacientes como en cortes disponibles, lo que puede afectar a la capacidad de generalización de los modelos. Aunque se ha aplicado validación cruzada para mitigar este problema, sería deseable contar con un volumen de datos mayor y más diverso. Además, el enfoque seguido se ha basado en la segmentación de cortes individuales en 2D, sin aprovechar la información contextual tridimensional que podría aportar una representación completa del volumen cerebral. Esto limita especialmente la segmentación de lesiones de forma irregular o difusa.

Otra limitación importante es la ausencia de técnicas de postprocesamiento clínico en las predicciones, como operaciones morfológicas para refinar los bordes o eliminar regiones espurias. Dichas técnicas podrían mejorar la calidad final de las máscaras generadas. Asimismo, cabe destacar que la arquitectura YOLO no está diseñada originalmente para segmentación semántica, sino para tareas de detección, por lo que su aplicación en este trabajo ha requerido una adaptación no estructural que puede limitar su rendimiento frente a modelos específicamente diseñados para segmentación.

Por otro lado, el sistema ha sido evaluado únicamente sobre un dominio clínico concreto —resonancia magnética cerebral en pacientes con esclerosis múltiple—, por lo que no puede garantizarse su aplicabilidad directa en otros contextos médicos o modalidades de imagen. Finalmente, las restricciones computacionales disponibles durante el desarrollo han condicionado el tamaño y profundidad de los modelos, así como el número de experimentos realizados, lo que sugiere que podrían alcanzarse mejores resultados con mayor capacidad de cómputo y entrenamiento más prolongado. Estas limitaciones abren múltiples líneas de mejora que se abordan en la sección de trabajo futuro.

Como líneas futuras, se propone explorar arquitecturas híbridas que combinen la eficiencia de detección de YOLO con la precisión de segmentación de U-Net, así como evaluar el sistema sobre conjuntos de datos más amplios y variados para mejorar su capacidad de generalización. Además, aunque la aplicación de superresolución mejora los resultados, solo se ha explorado la duplicación de la resolución. Con más recursos computacionales, se podría investigar si mayores superresoluciones ($\times 3$, $\times 4$, etc) pueden proporcionar mejoras mayores. También sería interesante estudiar el impacto de la aplicación de otras técnicas de preprocesado, como por ejemplo, las aumentaciones. Éstas permiten aumentar el número de muestras del conjunto de datos aplicando transformaciones como rotaciones geométricas o modificaciones de los colores entre otros, enriqueciendo el conjunto de datos que en el ámbito de las imágenes médicas suele ser escaso.

En resumen, los resultados muestran que U-Net, combinada con estrategias de selección centradas en la lesión y técnicas de superresolución, constituye una solución robusta y precisa para la segmentación de lesiones de esclerosis múltiple. Sin embargo, la elección del modelo ideal dependerá del contexto clínico y de los requisitos computacionales específicos de la aplicación final.

Referencias

[1] A. Rondinella, F. Guarnera, E. Crispino et al., *ICPR 2024 Competition on Multiple Sclerosis Lesion Segmentation – Methods and Results*, 2024. arXiv: 2410.07924 [eess.IV]. dirección: https://arxiv.org/abs/2410.07924.

[2] A. S. Lundervold y A. Lundervold, «An overview of deep learning in medical imaging focusing on MRI,» *Zeitschrift für Medizinische Physik*, vol. 29, n.º 2, págs. 102-127, 2019, Special Issue: Deep Learning in Medical Physics, ISSN: 0939-3889. DOI: https://doi.org/10.1016/j.zemedi.2018.11.002. dirección: https://www.sciencedirect.com/science/article/pii/S0939388918301181.

[3] G. Litjens, T. Kooi, B. E. Bejnordi et al., «A survey on deep learning in medical image analysis,» *Medical Image Analysis*, vol. 42, págs. 60-88, 2017, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2017.07.005. dirección: https://www.sciencedirect.com/science/article/pii/S1361841517301135.

[4] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, 2016. arXiv: 1506.02640 [cs.CV]. dirección: https://arxiv.org/abs/1506.02640.

[5] O. Ronneberger, P. Fischer y T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015. arXiv: 1505.04597 [cs.CV]. dirección: https://arxiv.org/abs/1505.04597.

[6] C. Dong, C. C. Loy y X. Tang, *Accelerating the Super-Resolution Convolutional Neural Network*, 2016. arXiv: 1608.00367 [cs.CV]. dirección: https://arxiv.org/abs/1608.00367.

[7] R. W. Brown, Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson y R. Venkatesan, *Magnetic Resonance Imaging: Physical Principles and Sequence Design*, 2nd. Hoboken, NJ: John Wiley & Sons, Incorporated, 2014.

[8] M. A. Balafar, A. R. Ramli, M. I. Saripan y S. Mashohor, «Review of brain MRI image segmentation methods,» *Artificial Intelligence Review*, vol. 33, págs. 261-274,

2010. DOI: 10.1007/s10462-010-9155-0. dirección: https://doi.org/10.1007/s10462-010-9155-0.

[9] R. Bakshi, S. Ariyaratana, R. Benedict y L. Jacobs, «Fluid-attenuated inversion recovery magnetic resonance imaging detects cortical and juxtacortical multiple sclerosis lesions,» *Archives of Neurology*, vol. 58, n.º 5, págs. 742-748, mayo de 2001. DOI: 10.1001/archneur.58.5.742.

[10] N. Otsu, «A Threshold Selection Method from Gray-Level Histograms,» *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, n.º 1, págs. 62-66, 1979. DOI: 10.1109/TSMC.1979.4310076.

[11] E. Ilunga-Mbuyamba, J. G. Avina-Cervantes, J. Cepeda-Negrete, M. A. Ibarra-Manzano y C. Chalopin, «Automatic selection of localized region-based active contour models using image content analysis applied to brain tumor segmentation,» *Computers in Biology and Medicine*, vol. 91, págs. 69-79, dic. de 2017, Epub 2017 Oct 7, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2017.10.003.

[12] A. Kermi, K. Andjouh y F. Zidane, «Fully automated brain tumour segmentation system in 3D-MRI using symmetry analysis of brain and level sets,» *IET Image Processing*, vol. 12, n.º 11, págs. 1964-1971, 2018.

[13] M. Soltaninejad, G. Yang, T. Lambrou et al., «Supervised learning based multimodal MRI brain tumour segmentation using texture features from supervoxels,» *Computer Methods and Programs in Biomedicine*, vol. 157, págs. 69-84, 2018, ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2018.01.003. dirección: https://www.sciencedirect.com/science/article/pii/S016926071731355X.

[14] T. Zhan, F. Shen, X. Hong et al., «A Glioma Segmentation Method Using CoTraining and Superpixel-Based Spatial and Clinical Constraints,» *IEEE Access*, vol. 6, págs. 57 113-57 122, 2018. DOI: 10.1109/ACCESS.2018.2873674.

[15] S. Pereira, A. Pinto, V. Alves y C. A. Silva, «Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images,» *IEEE Transactions on Medical Imaging*, vol. 35, n.º 5, págs. 1240-1251, 2016. DOI: 10.1109/TMI.2016.2538465.

[16] M. Havaei, A. Davy, D. Warde-Farley et al., «Brain tumor segmentation with Deep Neural Networks,» *Medical Image Analysis*, vol. 35, págs. 18-31, 2017, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2016.05.004. dirección: https://www.sciencedirect.com/science/article/pii/S1361841516300330.

[17] F. Milletari, S.-A. Ahmadi, C. Kroll et al., «Hough-CNN: Deep learning for segmentation of deep brain regions in MRI and ultrasound,» *Computer Vision and Image Understanding*, vol. 164, págs. 92-102, 2017, Deep Learning for Computer Vision, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2017.04.002. dirección: https://www.sciencedirect.com/science/article/pii/S1077314217300620.

[18] G. Wang, W. Li, S. Ourselin y T. Vercauteren, «Automatic brain tumor segmentation based on cascaded convolutional neural networks with uncertainty estimation,» *Frontiers in Computational Neuroscience*, vol. 13, pág. 56, 2019. DOI: 10.3389/fncom.2019.00056.

[19] A. Myronenko, «3D MRI brain tumor segmentation using autoencoder regularization,» en *International MICCAI Brainlesion Workshop*, ép. Lecture Notes in Computer Science, vol. 11384, Cham: Springer International Publishing, 2018, págs. 311-320. DOI: 10.1007/978-3-030-11726-9_28.

[20] U. Baid, S. Talbar, S. Bileddula et al., «A novel approach for fully automatic intra-tumor segmentation with 3D U-Net architecture for gliomas,» *Frontiers in Computational Neuroscience*, vol. 14, pág. 10, 2020. DOI: 10.3389/fncom.2020.00010.

[21] K. Kamnitsas, C. Ledig, V. F. Newcombe et al., «Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation,» *Medical image analysis*, vol. 36, págs. 61-78, 2017.

[22] C. Ma, G. Luo y K. Wang, «Concatenated and connected random forests with multiscale patch driven active contour model for automated brain tumor segmentation of MR images,» *IEEE Transactions on Medical Imaging*, vol. 37, n.º 8, págs. 1943-1954, 2018.

[23] T. X. Pham, P. Siarry y H. Oulhadj, «A multi-objective optimization approach for brain MRI segmentation using fuzzy entropy clustering and region-based active contour methods,» *Magnetic Resonance Imaging*, vol. 61, págs. 41-65, 2019.

[24] E. Van Reeth, I. W. K. Tham, C. H. Tan y C. L. Poh, «Super-resolution in magnetic resonance imaging: A review,» *Concepts in Magnetic Resonance Part A*, vol. 40A, n.º 6, págs. 306-325, 2012. DOI: https://doi.org/10.1002/cmr.a.21249. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cmr.a.21249. dirección: https://onlinelibrary.wiley.com/doi/abs/10.1002/cmr.a.21249.

[25] C. Zhao, B. E. Dewey, D. L. Pham, P. A. Calabresi, D. S. Reich y J. L. Prince, «SMORE: A Self-Supervised Anti-Aliasing and Super-Resolution Algorithm for MRI Using Deep Learning,» *IEEE Transactions on Medical Imaging*, vol. 40, n.º 3, págs. 805-817, 2021. DOI: 10.1109/TMI.2020.3037187.

[26] C. Zhao, M. Shao, A. Carass et al., «Applications of a deep learning method for anti-aliasing and super-resolution in MRI,» *Magnetic Resonance Imaging*, vol. 64, págs. 132-141, 2019, Artificial Intelligence in MRI, ISSN: 0730-725X. DOI: https://doi.org/10.1016/j.mri.2019.05.038. dirección: https://www.sciencedirect.com/science/article/pii/S0730725X18306507.

[27] Q. Lyu, C. You, H. Shan y G. Wang, «Super-resolution MRI through deep learning,» *arXiv preprint arXiv:1810.06776*, 2018.

[28] L. Alzubaidi, J. Zhang, A. J. Humaidi et al., «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,» *Journal of Big Data*, vol. 8, n.º 1, pág. 53, 2021, ISSN: 2196-1115. DOI: 10.1186/s40537-021-00444-8. dirección: https://doi.org/10.1186/s40537-021-00444-8.

[29] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. MIT Press, 2016. dirección: https://www.deeplearningbook.org.

[30] S. Hao, Y. Zhou e Y. Guo, «A Brief Survey on Semantic Segmentation with Deep Learning,» *Neurocomputing*, vol. 406, págs. 302-321, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.11.118. dirección: https://www.sciencedirect.com/science/article/pii/S0925231220305476.

[31] O. Ronneberger, P. Fischer y T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» en *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells y A. F. Frangi,

eds., Cham: Springer International Publishing, 2015, págs. 234-241, ISBN: 978-3-319-24574-4.

[32] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, 2016. arXiv: 1506.02640 [cs.CV]. dirección: https://arxiv.org/abs/1506.02640.

[33] S. Anwar, S. Khan y N. Barnes, «A Deep Journey into Super-resolution: A Survey,» *ACM Comput. Surv.*, vol. 53, n.º 3, mayo de 2020, ISSN: 0360-0300. DOI: 10.1145/3390462. dirección: https://doi.org/10.1145/3390462.

[34] C. Dong, C. C. Loy y X. Tang, *Accelerating the Super-Resolution Convolutional Neural Network*, 2016. arXiv: 1608.00367 [cs.CV]. dirección: https://arxiv.org/abs/1608.00367.

[35] Y. Noh, *FSRCNN-pytorch*, https://github.com/yjn870/FSRCNN-pytorch, Accessed: 2025-05-24, 2019.

Apéndice A
Diseño e implementación

En este apéndice se describen los aspectos técnicos relacionados con el diseño del sistema desarrollado y su implementación práctica. En primer lugar, se justifica la arquitectura software elegida, detallando su estructura modular y los principios que la sustentan. A continuación, se presenta el entorno de desarrollo y ejecución utilizado para implementar el código y entrenar los modelos de segmentación.

Seguidamente, se analiza la organización del proyecto a nivel de directorios y módulos, explicando la función de cada uno dentro del flujo general. Posteriormente, se profundiza en la implementación, tanto del sistema de configuración de experimentos como de los modelos utilizados. Esta sección incluye la definición formal de los experimentos mediante archivos JSON, su validación mediante Pydantic, y la relación entre las clases mediante un diagrama UML.

Finalmente, se detallan las implementaciones específicas de los modelos empleados: FSRCNN para super-resolución, U-Net para segmentación semántica y YOLO para segmentación basada en polígonos. Se concluye con una sección dedicada a los hiperparámetros utilizados durante el entrenamiento de los modelos, y una tabla comparativa entre ambas arquitecturas.

A.1. Arquitectura software

La estructura del proyecto se ha diseñado siguiendo un enfoque modular, con el objetivo de facilitar su implementación, mantenimiento, escalabilidad y reutilización a lo largo del tiempo. Cada componente del sistema —desde el preprocesamiento de imágenes hasta la evaluación de los resultados— está encapsulado en módulos independientes, con responsabilidades claramente definidas. Esta separación lógica no solo favorece una organización más limpia y comprensible del código, sino que también permite desarrollar,

probar y modificar cada parte del sistema de forma aislada, minimizando el riesgo de errores colaterales al introducir cambios.

Una de las principales ventajas de este enfoque es la ejecución desacoplada de los módulos. Por ejemplo, es posible ejecutar únicamente el preprocesamiento de las imágenes y almacenar los resultados intermedios para su uso posterior en la fase de entrenamiento, sin necesidad de repetir todo el flujo. De igual modo, se puede cambiar el modelo de segmentación (por ejemplo, de U-Net a YOLO) sin alterar el resto de etapas, lo que favorece la experimentación con distintas arquitecturas y técnicas de forma eficiente.

Además, esta estructura modular permite ejecutar tareas en paralelo o distribuir la carga de trabajo si se dispone de los recursos necesarios. Por ejemplo, se puede realizar el preprocesamiento de nuevas imágenes mientras se entrena un modelo previamente configurado, lo cual mejora significativamente el aprovechamiento del tiempo y de los recursos computacionales disponibles. Esta capacidad de paralelización sería mucho más difícil de implementar en un sistema monolítico, en el que las distintas fases del proceso estarían fuertemente acopladas y dependientes entre sí.

Desde el punto de vista del mantenimiento, el diseño modular también permite detectar y aislar errores de forma más eficiente, así como incorporar nuevas funcionalidades sin necesidad de reescribir grandes partes del código existente. Por ejemplo, se podrían incluir fácilmente nuevas técnicas de preprocesamiento, arquitecturas de red o métricas de evaluación simplemente añadiendo nuevos módulos o extendiendo los existentes.

En cuanto a la escalabilidad, este sistema es también ventajoso. Si se desea introducir, por ejemplo, una nueva red o ampliar el preprocesado o añadir nuevas métricas, solo hay que modificar el módulo correspondiente y si es preciso las conexiones con el resto de módulos.

En resumen, el uso de una arquitectura modular ha sido clave para dotar al sistema de flexibilidad, claridad y capacidad de adaptación, aspectos fundamentales en un proyecto experimental como este, donde la comparación de diferentes enfoques y configuraciones es una parte esencial del trabajo.

La Figura 22 ilustra el flujo general del sistema a alto nivel. El proceso comienza con la lectura de las imágenes de entrada, que posteriormente se someten a una fase de preprocesamiento, procesando las transformaciones de forma secuencial. A continuación,

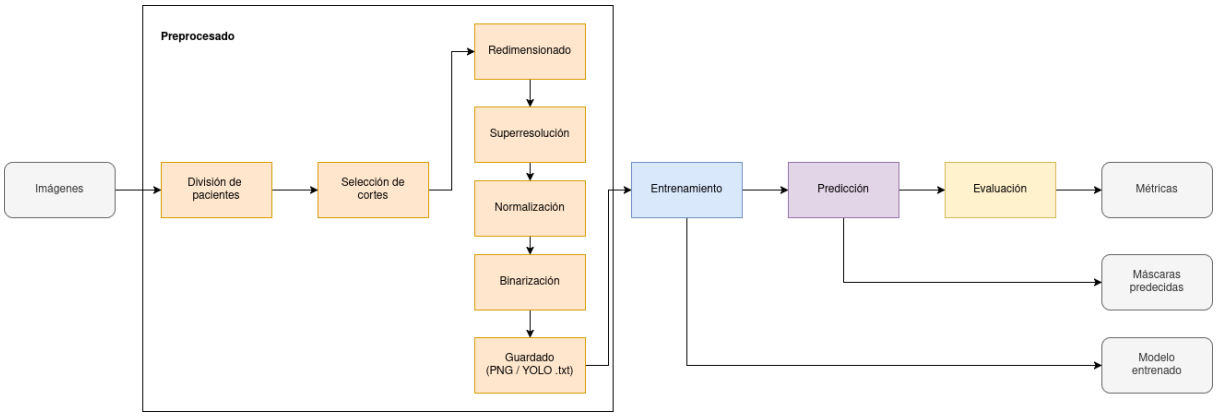


Figura 22: Diagrama del sistema implementado.

los datos procesados se utilizan en las etapas de entrenamiento, predicción y evaluación. Estas tres etapas genera tres tipos de salida respectivamente: el modelo entrenado (en formato `.pth` para U-Net y `.pt` para YOLO), las métricas de rendimiento calculadas (almacenadas en formato JSON) y las máscaras de segmentación predichas (guardadas como archivos PNG).

A.2. Entorno de desarrollo

El desarrollo del proyecto se ha llevado a cabo en dos entornos principales, ambos con sistema operativo Linux: un equipo personal y la plataforma Google Cloud, utilizada para el entrenamiento de modelos en GPU.

- **Equipo local**: portátil con procesador Intel Core i9-13900H de 13ª generación, 32 GB de RAM y tarjeta gráfica NVIDIA RTX 4060 con 8 GB de memoria dedicada. Este entorno se utilizó tanto para la implementación del código como para la validación y pruebas preliminares.

- **Equipo remoto**: se utilizó una máquina remota para los entrenamientos de los modelos conectándo a través de SSH, con dos GPUs NVIDIA RTX A6000, cada una con 48 GB de memoria dedicada y una memoria RAM de 128GB.

El proyecto está escrito enteramente en **Python**, gestionado mediante un entorno virtual creado con **venv**, donde se instalaron todas las dependencias necesarias para

garantizar la portabilidad y reproducibilidad de los experimentos. A continuación, se detallan las principales bibliotecas utilizadas:

- `segmentation-models-pytorch`: empleada para implementar la arquitectura U-Net con codificador `ResNet34`.
- `ultralytics`: utilizada para la integración de modelos YOLO en tareas de segmentación.
- `SimpleITK` y `nibabel`: para la lectura y manipulación de imágenes médicas en formato NIfTI.
- `scikit-learn`: proporciona las funciones para la validación cruzada.
- `Pillow` y `matplotlib`: para la visualización y el guardado de resultados.
- `loguru`: para el registro de logs y mensajes de depuración.
- `pydantic`: para la validación de los parámetros necesarios para instanciar las distintas clases.
- `google-cloud-storage`: para guardar los resultados de los modelos en la nube de Google en caso de trabajar con Vertex AI.

El control de versiones del proyecto se ha realizado mediante un repositorio privado en GitHub, lo cual ha permitido mantener un historial estructurado del código, facilitar el trabajo modular y garantizar la trazabilidad de los cambios a lo largo del desarrollo del trabajo.

A.3. Estructura del proyecto

El programa desarrollado es una herramienta de línea de comandos (*Command Line Interface*, CLI) que requiere un archivo de configuración en formato JSON, donde se define la estructura y los parámetros de los experimentos a ejecutar. Se ha optado por utilizar un archivo JSON para centralizar la configuración, ya que el número de parámetros necesarios haría muy complejo su especificación completa a través de la línea de comandos. A continuación, se describe la organización general de carpetas y archivos del proyecto:

- `main.py`: punto de entrada del programa.
- `cli/`: este módulo tiene las funciones encargadas de parsear los argumentos del programa y el archivo de configuración JSON.
- `config/`: contiene la clase centralizada encargada de cargar y guardar las variables de entorno para funcionalidades opcionales.
- `interfaces/`: el flujo de preprocesado está diseñado para que cada parte implemente funcionalidad común definida en este módulo.
- `datasets/`: carpeta dedicada al almacenamiento y organización de los datos de entrada, incluyendo imágenes de resonancia y sus máscaras asociadas.
- `logs/`: contiene los archivos de registro generados durante la ejecución del entrenamiento y evaluación, permitiendo el seguimiento de resultados y depuración.
- `results/`: almacena las salidas generadas por los distintos modelos, incluyendo segmentaciones predichas, métricas de evaluación y visualizaciones.
- `schemas/`: define los esquemas del flujo para asegurar la validez de los parámetros de configuración. Por ejemplo, el archivo `pipeline_schemas.py` especifica las clases y estructuras necesarias para la correcta configuración del proceso.
- `net/`: contiene las implementaciones de los modelos de red neuronal utilizados:
  - `FSRCNN/`: implementación de la red de super-resolución basada en FSRCNN, incluyendo modelos preentrenados para escalado x2, x3 y x8.
  - `unet/`: contiene la implementación de la arquitectura U-Net empleada para la segmentación de imágenes médicas.
  - `yolo/`: incluye la configuración y modelo de YOLO adaptado para la tarea de segmentación, con su archivo de configuración `yolo_config.yaml`.
- `steps/`: estructura clave del proyecto que organiza el flujo de trabajo completo mediante submódulos claramente diferenciados:

- `preprocessing/`: implementación de la preparación de los datos, división del conjunto, aplicación de la super-resolución y guardado de imágenes resultantes.
- `training/`: módulo encargado del entrenamiento de los modelos.
- `prediction/`: implementación de la lógica de inferencia y generación de predicciones a partir de los modelos entrenados.
- `evaluation/`: scripts para la evaluación cuantitativa y cualitativa de los resultados.

A.4. Implementación del sistema

A alto nivel, el programa tiene dos componentes: el experimento y el código que lo ejecuta. Los experimentos se definen de forma completamente declarativa a través de archivos en formato JSON. Cada uno de estos archivos representa un experimento individual e incluye los pasos del flujo a ejecutar (preprocesamiento, entrenamiento, predicción y evaluación), junto con los parámetros específicos requeridos por cada módulo. Estos archivos son parseados al inicio del proceso y transformados en instancias de clases de configuración definidas en el módulo `schemas/pipeline_schemas.py`. Gracias al uso de la librería Pydantic, estas configuraciones son validadas y tipadas.

Listing 1: Plantilla de archivo de configuración JSON.

```
{
    "id": "<experimento_id>",
    "preprocess": {
        "net": "<'unet' o 'yolo'>",
        "src_path": "<ruta a al dataset>",
        "dst_path": "<ruta donde escribir los resultados>",
        "resize": [320, 320],
        "split": 0.7,
        "seed": 42,
        "strategy": "<'all_slices', 'lesion_slices',
    ↪ 'brain_slices'>",
        "super_scale": <1, 2, 3, 8>
```

```
    },
    "train": {
        "net": "<igual que preprocess.net>",
        "src_path": "<ruta al dataset preprocesado>",
        "dst_path": "<donde escribir el modelo entrenados>",
        "batch_size": 16,
        "use_kfold": false,
        "epochs": 25,
        "learning_rate": 0.001
    },
    "predict": {
        "net": "<igual train.net>",
        "model_path": "<ruta/al/modelo/entrenado>",
        "src_path": "<ruta a las mascaras>",
        "dst_path": "<ruta donde escribir las predicciones>"
    },
    "evaluate": {
        "net": "<igual que train.net>",
        "model_path": "<ruta/al/modelo/entrenado>",
        "src_path": "<ruta al dataset resultado del paso predict>",
        "pred_path": "<ruta donde guardar metricas>",
        "gt_path": "<directorio donde estan las mascaras
    ↪ ground-truth>"
    }
}
```

Listing 1: Plantilla de archivo de configuración JSON.

Por otra parte está el código que ejecuta el experimento. Este código se organiza en clases, véase la Figura 23. Todas las clases de configuración heredan de `BaseModel` (Pydantic), lo que permite la validación automática y la gestión estructurada de los parámetros definidos en archivos JSON. La clase central, `PipelineConfig`, encapsula de forma opcional una de las configuraciones posibles del sistema: `PreprocessConfig`, `TrainConfig`, `PredictConfig` o `EvaluateConfig`. Cada una de estas clases define los pa-

rámetros específicos necesarios para ejecutar su correspondiente etapa del pipeline. La clase `TrainConfig` tiene parámetros para activar y configurar la validación cruzada usando Kfolds. Se emplean enumeraciones para restringir valores válidos de atributos como la red utilizada (`Net`), el método de escalado (`SuperScale`), la estrategia de selección de cortes (`Strategy`) o el método de redimensionado (`ResizeMethod`). Finalmente, la clase `SegmentationMetrics` encapsula las métricas de evaluación empleadas para valorar el rendimiento de los modelos. Finalmente, la clase `EnvConfig` se encarga de cargar variables de entorno necesarias para usar funcionalidades opcionales.

Dentro del módulo de preprocesamiento, se ha optado por un diseño basado en clases abstractas e interfaces que permiten una alta flexibilidad, reutilización y extensibilidad de los distintos componentes del flujo. La idea es facilitar el añadir nuevos pasos de preprocesado implementando la clase abstracta y añadiendo otra opción al archivo de configuración. El módulo se estructura en cuatro componentes principales (obsérvese la Figura 24), cada uno definido por una clase abstracta :

- **ImageSaver**: define la interfaz para guardar imágenes FLAIR y máscaras segmentadas. Cuenta con dos implementaciones: `PNGSaver`, que guarda los datos en formato PNG, y `YOLOSegSaver`, que además convierte las máscaras a formato compatible con segmentación en YOLO (polígonos normalizados en archivos `.txt`).

- **SliceSelector**: abstrae la lógica de selección de cortes 2D a partir de volúmenes 3D. Entre sus implementaciones se encuentran:

  - `AllSliceSelector`: selecciona todos los cortes disponibles.
  - `LesionOnlySelector`: selecciona únicamente los cortes con presencia de lesión.
  - `BrainSelector`: selecciona los cortes que contienen tejido cerebral, con o sin lesión.

- **PatientSplitter**: define el criterio de división de pacientes entre los subconjuntos de entrenamiento, validación y test. `RandomPatientSplitter` realiza esta partición de forma aleatoria pero reproducible, asegurando que no haya solapamiento de pacientes entre subconjuntos.

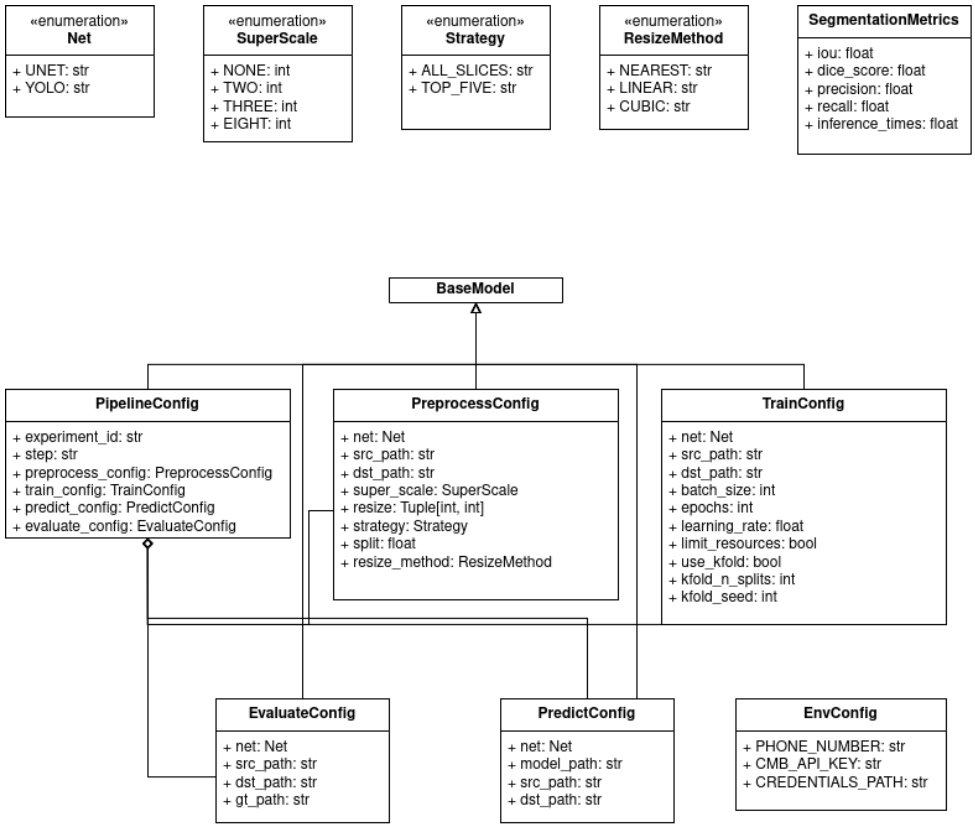


Figura 23: Diagrama de clases del sistema de configuración del flujo. Se muestra la relación entre las distintas clases que definen los parámetros de ejecución para cada etapa del procesamiento (preprocesado, entrenamiento, predicción y evaluación), así como las enumeraciones utilizadas para acotar los valores válidos. Todas las clases heredan de `BaseModel` para beneficiarse de la validación automática de Pydantic. La clase `PipelineConfig` actúa como contenedor central y se relaciona opcionalmente con una única configuración activa.

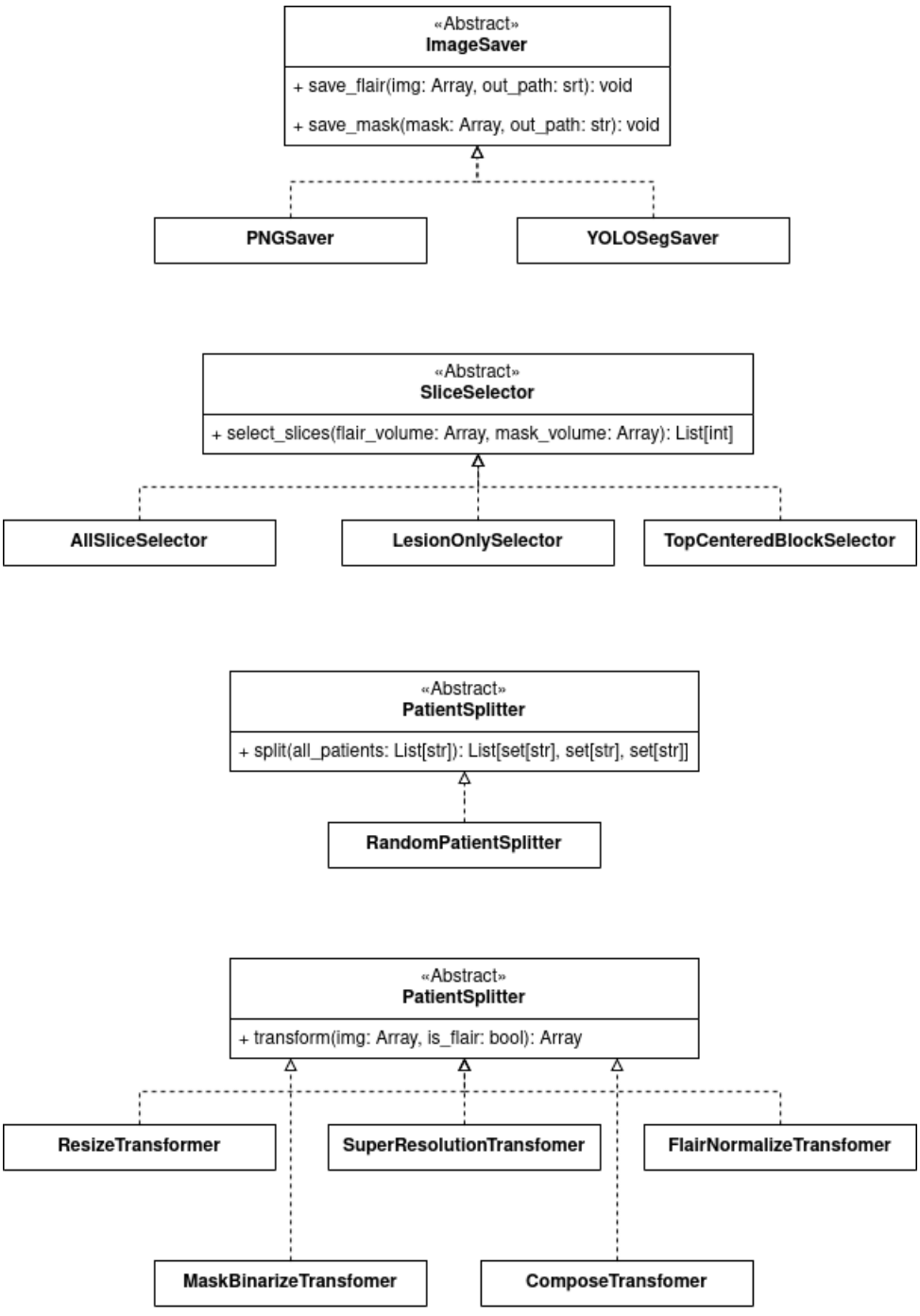


Figura 24: Diagrama de clases del sistema de preprocesamiento.

- **Transformer**: representa una operación de transformación aplicada a una imagen 2D, como redimensionado, normalización o super-resolución. Esta interfaz permite componer múltiples transformaciones mediante la clase abstracta `ComposeTransformer`. Entre las transformaciones disponibles se incluyen:

  - `ResizeTransformer`: redimensiona la imagen utilizando un método configurable (e.g. `INTER_NEAREST` para máscaras).
  - `SuperResolutionTransformer`: aplica un modelo FSRCNN únicamente sobre imágenes FLAIR para mejorar su resolución.
  - `FlairNormalizeTransformer`: normaliza las intensidades de imágenes FLAIR al rango $[0, 255]$.
  - `MaskBinarizeTransformer`: asegura que las máscaras sean estrictamente binarias (valores 0 o 255).

Este diseño desacopla completamente la lógica de procesamiento de imágenes del guardado y de la selección de cortes, permitiendo extender o modificar cualquier parte del flujo (por ejemplo, añadir aumentos de datos o nuevos formatos de salida) sin afectar al resto del sistema.

A.5. FSRCNN

La implementación utilizada se ha tomado del repositorio `FSRCNN-pytorch`[1] [35] y modificada ligeramente. Se proporcionan modelos preentrenados y adaptados para PyTorch. Estos modelos han sido entrenados originalmente con el conjunto clásico de 91 imágenes naturales, conocido como *91-image dataset*, ampliamente utilizado en la literatura para tareas de super-resolución. Para la evaluación del modelo, se emplea también el dataset *Set5*, un conjunto de imágenes de alta resolución usado habitualmente como benchmark. Ambos conjuntos se degradan artificialmente mediante interpolación bicúbica para generar versiones con escalado $\times 2$, $\times 3$ y $\times 8$, que permiten evaluar la capacidad del modelo para reconstruir detalles a partir de imágenes de baja resolución. Esta estrategia permite

[1] https://github.com/yjn870/FSRCNN-pytorch

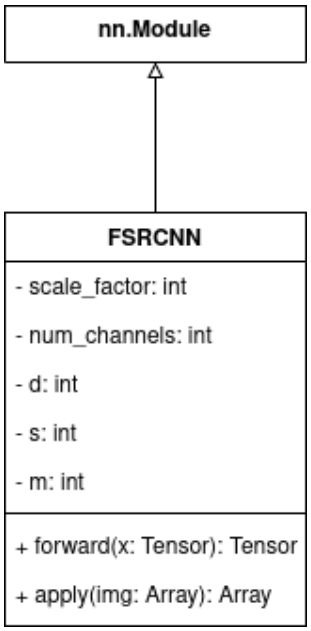


Figura 25: Diagrama de clases de la red FSRCNN.

incorporar super-resolución en el flujo de segmentación sin necesidad de realizar un entrenamiento adicional, reutilizando pesos ya optimizados para mejorar la calidad visual de las imágenes de entrada.

La clase `FSRCNN` (Figura 25), que hereda de `torch.nn.Module`, encapsula la arquitectura del modelo. El primer bloque (`first_part`) realiza la extracción de características mediante una convolución con kernel 5×5 seguida de una activación PReLU. El segundo bloque (`mid_part`) lleva a cabo una compresión dimensional con convoluciones 1×1, seguida de múltiples capas de convoluciones 3×3 y activaciones PReLU que actúan como un mapa de representación interna. Finalmente, el bloque de reconstrucción (`last_part`) utiliza una convolución transpuesta con kernel 9×9 para generar una imagen de salida con mayor resolución, escalada por un factor configurable (×2, ×3, ×8).

Entre sus atributos destacan los parámetros estructurales de la red, como el factor de escalado (`scale_factor`), el número de canales de entrada (`num_channels`) y los hiperparámetros internos (`d`, `s`, `m`) que controlan el número de filtros y bloques internos de convolución. La clase implementa los siguientes métodos principales:

- `forward(x)`: define el paso hacia adelante del modelo, aplicando las capas secuenciales sobre un tensor de entrada.

- `apply`: aplica super-resolución sobre una imagen de entrada y devuelve la imagen con el nuevo tamaño.

Esta implementación ha sido adaptada en el módulo `net/FSRCNN/`, y se compone de tres bloques principales: una capa de extracción de características, un bloque de mapeo no lineal y una etapa de reconstrucción (véase Figura 26).

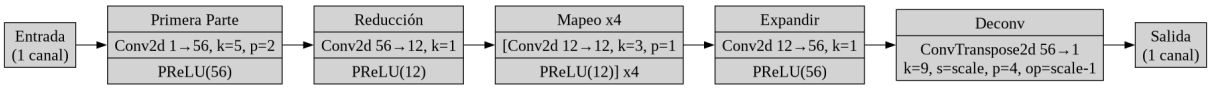


Figura 26: Arquitectura de la red FSRCNN (Fast Super-Resolution Convolutional Neural Network). El modelo toma como entrada una imagen en baja resolución de un solo canal y pasa por una serie de bloques secuenciales: una etapa de expansión inicial (First Part), reducción de canales (Shrink), mapeo no lineal profundo (Mapping), reexpansión de canales (Expand) y una deconvolución final (Deconv) para generar la imagen superresuelta. Cada capa convolucional está seguida por una activación PReLU que mejora la capacidad de aprendizaje del modelo. El parámetro `scale_factor` define el grado de super-resolución.

A.6. U-Net

La implementación de la U-Net tiene un codificador preentrenado ResNet34, adaptada para trabajar con imágenes de resonancia magnética en escala de grises (un canal). Esta elección se fundamenta en la robustez de U-Net para tareas de segmentación médica y en la capacidad de ResNet34 para extraer características profundas de forma eficiente. En La Figura 27 se muestra el diagrama de clases correspondiente a la implementación de la arquitectura U-Net dentro del sistema desarrollado. Este modelo ha sido encapsulado en una clase principal llamada `Unet`, la cual centraliza la lógica de entrenamiento, predicción y evaluación del modelo.

Esta clase contiene los atributos principales necesarios para definir el modelo: el dispositivo de cómputo (`device`), el nombre del codificador (`encoder_name`) y una instancia de configuración (`config`) que puede ser de tipo `TrainConfig`, `EvaluateConfig` o `PredictConfig`, dependiendo del contexto de ejecución. Entre sus métodos públicos más relevantes se encuentran:

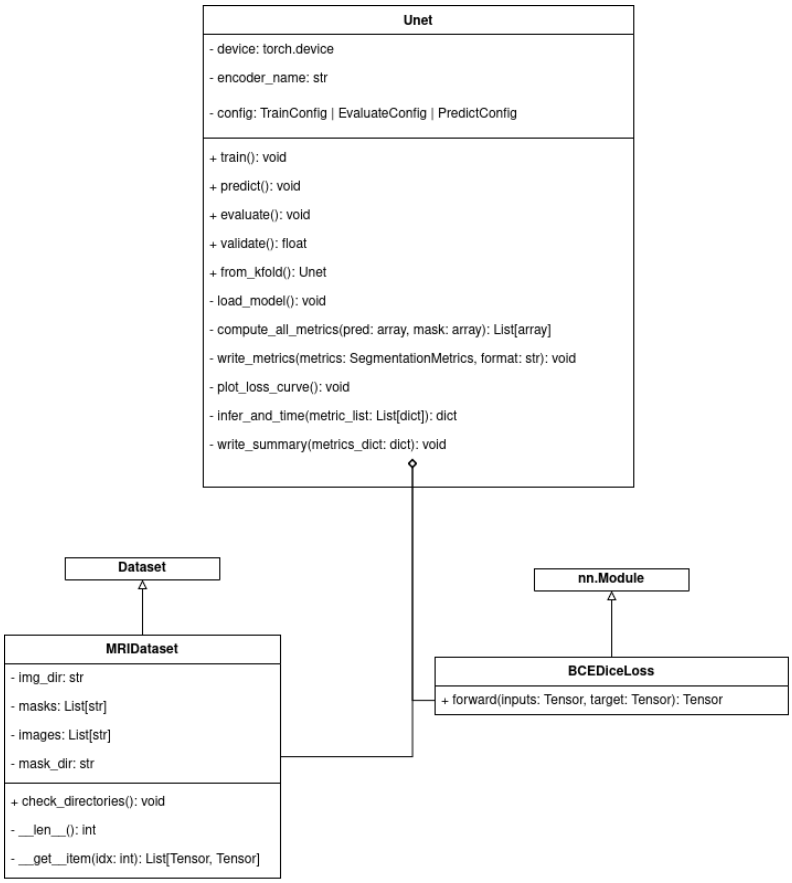


Figura 27: Diagrama de clases de la red U-Net.

- `train()`, `predict()` y `evaluate()`, que ejecutan las etapas principales del flujo de entrenamiento.
- `validate()`, que devuelve una métrica de validación para monitorizar el rendimiento durante el entrenamiento.
- `from_kfold()`, que permite instanciar el modelo en el contexto de validación cruzada.
- `infer_and_time()` y `write_summary()`, dedicados al análisis del rendimiento y al registro de resultados.

Además, la clase implementa métodos privados para la carga del modelo (`load_model()`), la evaluación de métricas (`compute_all_metrics()`), el guardado de resultados (`write_metrics()`), y la visualización de la curva de pérdida (`plot_loss_curve()`). La clase `Unet` depende de dos componentes externos clave:

- `MRIDataset`, que hereda de `Dataset` de PyTorch y se encarga de gestionar el conjunto de datos. Define atributos como el directorio de imágenes, listas de rutas de imágenes y máscaras, y métodos para validar la estructura de carpetas, calcular el tamaño del conjunto (`__len__`) y acceder a los elementos de forma indexada (`__getitem__`).
- `BCEDiceLoss`, una clase que hereda de `nn.Module` y representa una función de pérdida personalizada compuesta por la combinación de `Binary Cross Entropy` y `Dice Loss`, diseñada para mejorar la segmentación en contextos con clases desbalanceadas.

El flujo de trabajo de la red se inicia con la carga de datos mediante `DataLoader`, que gestiona tanto las imágenes como sus máscaras correspondientes, organizadas en conjuntos de entrenamiento, validación y prueba. Las imágenes se procesan por una red U-Net con las siguientes características:

- Codificador: ResNet34 preentrenado (transfer learning).
- Entrada: imágenes de un solo canal.

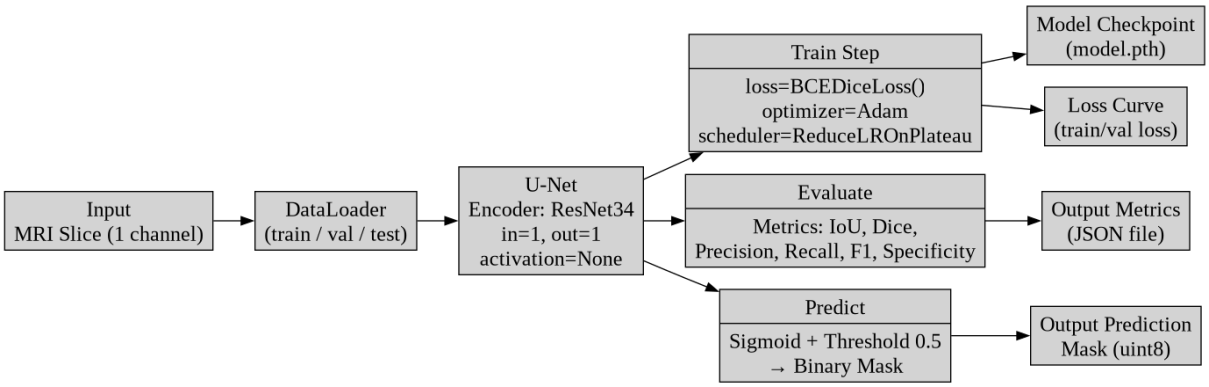


Figura 28: Diagrama del sistema de segmentación basado en U-Net.

- Salida: mapa de segmentación con activación lineal (sin activación final).

Durante el entrenamiento, la salida del modelo se evalúa con una función de pérdida combinada `BCEDiceLoss`, que suma la `Binary Cross-Entropy` con el `Dice Loss` suavizado:

$$\mathcal{L}(x, y) = \underbrace{\text{BCEWithLogitsLoss}(x, y)}_{\text{Pérdida de entropía binaria}} + \underbrace{1 - \frac{2\sum_i \sigma(x_i) y_i + \varepsilon}{\sum_i \sigma(x_i) + \sum_i y_i + \varepsilon}}_{\text{Dice Loss}} \tag{3}$$

- x es la salida cruda del modelo (logits),
- $\sigma(x)$ es la función sigmoide aplicada a x,
- y es la máscara de verdad (ground truth),
- $\epsilon = 10e - 5$ es un término de suavizado para evitar divisiones por cero.

Esto permite balancear regiones lesionadas (menores en tamaño) y no lesionadas (mayores) dentro de la imagen. La optimización se realiza mediante Adam, y el aprendizaje se regula con una política de reducción de tasa de aprendizaje (`ReduceLROnPlateau`) basada en el valor de pérdida de validación. En la fase de evaluación, se carga el modelo entrenado y se calcula el conjunto de métricas descritas en la sección Metodología. En la fase de predicción, el modelo produce máscaras binarias aplicando una función sigmoid seguida de un umbral de 0,5. Estas máscaras se guardan como imágenes binarizadas en formato `uint8`.

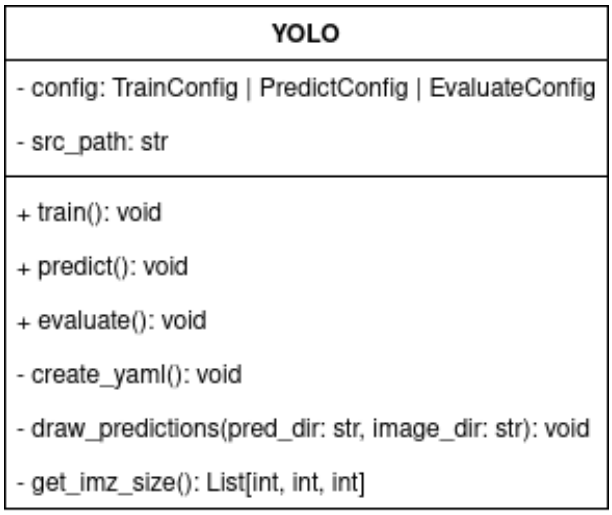


Figura 29: Diagrama de clases de la red YOLO.

En la Figura 28, se muestra el flujo de datos desde la entrada de imágenes en escala de grises, el procesamiento mediante la red U-Net con codificador ResNet34, y las tres fases principales: entrenamiento, evaluación y predicción. En el entrenamiento, se emplea la función de pérdida BCEDiceLoss y se genera un modelo entrenado (.pth) junto con curvas de pérdida. En la evaluación, se calculan métricas estándar de segmentación. En la predicción, se exportan máscaras binarias para su análisis posterior.

A.7. YOLO

Para la YOLO, se ha utilizado el modelo en su versión de segmentación (`YOLOv11-seg.pt`), disponible a través de la librería Ultralytics. A diferencia de U-Net y FSRCNN, donde se ha usado la librería `Torch`, Ultralytics proporciona la implementación de la YOLO junto con todas las herramientas necesarias para trabajar con la red. Por tanto, solo ha sido necesario implementar métodos para: leer y escribir imágenes, calcular y guardar métricas. Además de los tres modos de ejecución: entrenamiento, predicción y evaluación, definidos a través de clases de configuración específicas (TrainConfig, PredictConfig, EvaluateConfig). La Figura 29, representa la clase encargada de encapsular la lógica de entrenamiento, predicción y evaluación la red YOLO.

Esta clase está diseñada como una interfaz simplificada para interactuar con los modelos provistos por la librería `ultralytics`. Contiene los siguientes atributos principales:

- `config`: una instancia de `TrainConfig`, `PredictConfig` o `EvaluateConfig`, en función del modo de ejecución.
- `src_path`: ruta al conjunto de datos de entrada sobre el cual se realiza el entrenamiento, la inferencia o la evaluación.

Entre sus métodos públicos destacan:

- `train()`, `predict()` y `evaluate()`: que implementan las tres etapas fundamentales del ciclo de vida del modelo.
- `create_yaml()`: encargado de generar automáticamente el archivo de configuración `data.yaml` requerido por YOLO, en función de las rutas definidas en el experimento y del número de clases.
- `draw_predictions(pred_dir, image_dir)`: que reconstruye las máscaras de segmentación a partir de los archivos de predicción en formato texto (.txt), generando salidas visuales en formato imagen (.png) alineadas con las originales.
- `get_imz_size()`: función auxiliar que devuelve las dimensiones de una muestra del conjunto de datos.

En los modos de entrenamiento y predicción, se genera automáticamente un archivo `data.yaml` (requerido para entrenar el modelo usado) con los siguientes campos:

- `path`: ruta relativa al conjunto de datos.
- `train` y `val`: rutas internas a imágenes de entrenamiento y validación.
- `nc = 1`: número de clases (solo una clase: "lesion").
- `names = ["lesion"]`: nombre de la clase.
- `task = "segment"`: tipo de tarea (segmentación semántica).

Una peculiaridad de la YOLO es que, a diferencia de las arquitecturas como U-Net, que operan directamente sobre máscaras en formato de imagen (.png), el modelo YOLO en su modalidad de segmentación requiere un formato de entrada diferente: archivos de

texto (.txt) que contienen las coordenadas de los polígonos de cada objeto segmentado. Cada archivo de texto describe una imagen del conjunto de entrenamiento o validación y contiene una o más líneas, donde cada línea representa una instancia segmentada mediante un polígono cerrado. La estructura de cada línea es:

```
<class_id> <x1> <y1> <x2> <y2> ... <xn> <yn>
```

- `class_id` índice entero que representa la clase del objeto (en este caso, siempre 0, ya que solo se segmentan lesiones).
- `x_i`, `y_i`: coordenadas normalizadas (valores entre 0 y 1) de los puntos del contorno del polígono, en orden secuencial.

Este formato es obligatorio tanto para entrenamiento como para predicción y está alineado con la filosofía de YOLO de representar objetos como formas geométricas mínimas (cajas, puntos o polígonos) y trabajar sobre texto plano para velocidad y flexibilidad. Dado que el dataset original del proyecto contiene máscaras en formato de imagen binaria (.png), ha sido necesario implementar un paso de preprocesamiento previo que:

1. Detecta los contornos de cada máscara.
2. Extrae los puntos de los polígonos.
3. Normaliza las coordenadas respecto a la dimensión de la imagen.
4. Escribe los resultados en archivos .txt compatibles con el formato YOLO segment.

Este paso es fundamental para que el modelo pueda ser entrenado correctamente. Del mismo modo, tras la fase de predicción, se implementa una función (`draw_predictions`) que:

1. Lee los archivos .txt generados por el modelo.
2. Reconstruye los polígonos a escala original.
3. Los convierte en máscaras binarias rellenadas para poder evaluar con las métricas.

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