Preprocesamiento Basado en Textura para la Segmentación de Tumores en Ecografías Mamarias con MONAI

María del Mar Ávila, Juan del Junco, Nerea Jiménez
Tutora: María José Jiménez

13 de mayo de 2025

Resumen

En este trabajo se propone una forma de mejorar la detección automática de tumores en ecografías mamarias, utilizando técnicas de preprocesamiento y redes neuronales. Las ecografías son imágenes que pueden tener mucho ruido y poco contraste, lo que hace difícil que un modelo detecte tumores. Para solucionar esto, antes de entrenar la red neuronal, se aplican distintos métodos de mejora de imagen. Después, estas imágenes se procesan con una red neuronal llamada U-Net, muy utilizada en medicina para separar automáticamente regiones de interés en una imagen.

Todo el sistema se ha implementado con la ayuda de MONAI, una librería especializada en imágenes médicas. Se ha utilizado un conjunto real de imágenes de mama (el dataset BUSI) y se han probado diferentes versiones del modelo: con imágenes originales, con filtros y con texturas. Los resultados muestran que aplicar estas técnicas de mejora antes del entrenamiento permite al modelo hacer predicciones más precisas. Esto demuestra que el preprocesamiento puede ser una herramienta clave para mejorar los sistemas de diagnóstico por imagen.

Palabras clave: segmentación automática, ecografía mamaria, U-Net, MONAI, preprocesamiento de imágenes, aprendizaje profundo.

1. Introducción

El cáncer de mama representa una de las principales causas de mortalidad femenina. La segmentación precisa de tumores en imágenes de ecografía mamaria es esencial para el diagnóstico asistido por computadora. Sin embargo, estas imágenes presentan artefactos, bajo contraste y ruido speckle que dificultan la segmentación automática. Este trabajo propone comparar métodos de preprocesamiento tradicionales y texturales para mejorar la calidad de segmentación utilizando una red U-Net.

2. Planteamiento Teórico

La inteligencia artificial ha transformado numerosas disciplinas, y la medicina no es la excepción. [1] En particular, la segmentación automática de imágenes médicas se ha vuelto una herramienta de gran valor en el diagnóstico clínico. Este proyecto se centra en el preprocesamiento y la segmentación de tumores en ecografías mamarias utilizando una red neuronal llamada U-Net, entrenada con ayuda de la librería especializada MONAI.

2.1. Preprocesamiento

El preprocesamiento de las imágenes antes del entrenamiento es esencial para mejorar la calidad del aprendizaje. En este trabajo, se aplican técnicas como el redimensionamiento, la normalización de intensidades y filtros tradicionales para la reducción de ruido. Esto permite estandarizar los datos de entrada, mejorando la estabilidad y efectividad del modelo entrenado.

En concreto y en el ámbito que nos compete, las imágenes médicas presentan numerosos desafíos: pueden estar en distintos tamaños, tener ruido o intensidades mal distribuidas. Por eso, antes de alimentar la red neuronal, se aplican procesos de preprocesamiento que pueden incluir:

- **Redimensionamiento de imágenes:** El redimensionamiento de las imágenes a un tamaño uniforme es un paso inicial crucial. Las imágenes originales pueden tener diferentes resoluciones, por lo que unificarlas facilita el procesamiento y reduce la carga computacional. Se utilizan técnicas de interpolación (como bilineal o bicúbica) para mantener la calidad de la imagen al cambiar su tamaño.

- **Normalización de intensidades:** La normalización consiste en escalar los valores de intensidad de píxel a un rango común, como [0,1]. Esto mejora la estabilidad y la eficiencia del entrenamiento del modelo, ya que reduce la variabilidad entre imágenes. Comúnmente se utiliza la normalización min-max o la estandarización basada en media y desviación estándar.

- **Filtro Butterworth:** El filtro Butterworth es un filtro frecuencial [17] que puede funcionar como pasa-bajos o pasa-altos. Su principal ventaja es que proporciona una transición suave en la frecuencia de corte, evitando artefactos bruscos. El filtro pasa-bajos suaviza la imagen eliminando el ruido de alta frecuencia, mientras que el pasa-altos resalta los bordes. La forma matemática del filtro asegura que no haya una caída abrupta, manteniendo una transición gradual.

- **Filtro de mediana adaptativa:** Este filtro elimina eficazmente el ruido impulsivo (sal y pimienta) adaptando dinámicamente el tamaño de la ventana de filtrado. Si una ventana pequeña no es suficiente para identificar correctamente la mediana de los píxeles, el filtro aumenta el tamaño de la ventana hasta encontrar una mediana adecuada. Así, preserva los bordes y detalles mientras elimina el ruido localizado. [18]

- **Características estadísticas de primer orden:** Estas características describen la distribución de las intensidades de los píxeles de forma individual, sin considerar su

posición. Incluyen métricas como la media, varianza, asimetría, curtosis y entropía.[2] Estas propiedades permiten caracterizar la luminosidad y el contraste general de la imagen.

- **Características estadísticas de segundo orden:** Se enfocan en las relaciones espaciales entre los píxeles. Se obtienen, por ejemplo, mediante la matriz de co-ocurrencia de niveles de gris (GLCM),[2] que mide con qué frecuencia aparecen combinaciones de valores de gris a cierta distancia y dirección. A partir de esta matriz se calculan métricas como contraste, homogeneidad, energía y correlación, que describen patrones de textura en la imagen.

2.2. Redes Neuronales Convolucionales (CNN)

Una **red neuronal convolucional** (o *Convolutional Neural Network*, CNN) es un tipo de red de aprendizaje profundo especialmente adecuada para procesar imágenes. A diferencia de las redes totalmente conectadas, las CNN emplean *capas convolucionales* que aplican filtros sobre la imagen para extraer automáticamente características locales (por ejemplo, bordes, texturas) y *capas de pooling* para reducir la resolución espacial, preservando las características más relevantes. Esto permite aprender representaciones jerárquicas: las primeras capas detectan patrones simples y las más profundas combinan estas características en patrones complejos propios del contenido de la imagen. En resumen, una CNN es una arquitectura capaz de **aprender directamente a partir de los datos** de imagen, identificando patrones visuales útiles para la tarea dada.[3]

En nuestro proyecto se utiliza una CNN para llevar a cabo la segmentación automática de tumores en ecografías mamarias. La red convolucional aprende a reconocer, píxel a píxel, qué regiones de la imagen corresponden a tejido tumoral y cuáles a tejido normal o fondo, a partir de ejemplos anotados. Gracias a las convoluciones, la red puede detectar las características visuales sutiles de un tumor en una ecografía (como variaciones de textura o bordes difusos). Este enfoque automatizado reemplaza la necesidad de definir manualmente criterios de segmentación, permitiendo que el modelo "aprenda" a partir de los datos.

Segmentación Semántica en Ecografías Mamarias

La **segmentación semántica** es una tarea de visión por computador que consiste en asignar una etiqueta de clase a cada píxel de una imagen mediante un algoritmo de aprendizaje profundo. En otras palabras, el resultado es una máscara donde cada píxel está clasificado como perteneciente a cierta categoría. En nuestro caso, trabajamos con una segmentación binaria: cada píxel de la ecografía se clasifica como "tumor" o "no tumor".[4]

Aplicada a **ecografías mamarias**, permite delimitar de forma precisa las lesiones tumorales en imágenes de ultrasonido. Estas imágenes suelen tener bajo contraste y presencia de ruido, lo que dificulta su interpretación. Mediante segmentación automática, el sistema aprende de ejemplos anotados a identificar patrones propios de los tumores. El resultado es una máscara que marca su ubicación y forma, lo cual puede acelerar y mejorar el diagnóstico clínico.

Para evaluar la calidad de la segmentación, el proyecto implementa métricas como el *coeficiente de Dice* (DSC) y el *Índice de Jaccard* (IoU), calculadas en el archivo `metrics_evaluation.py`. Estas métricas comparan la superposición entre la predicción del modelo y la máscara real. [5]

Arquitectura U-Net: Codificación, Decodificación y Conexiones de Salto

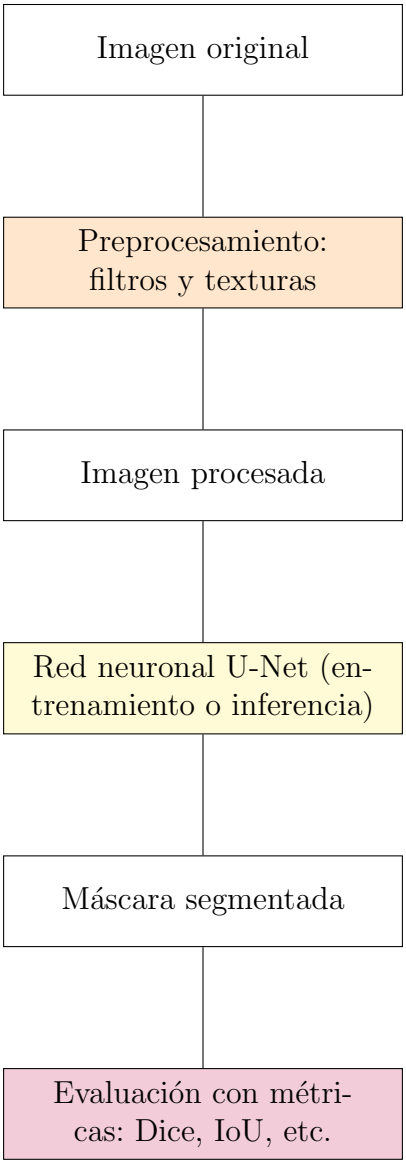
La arquitectura **U-Net** fue diseñada específicamente para segmentación de imágenes médicas. Tiene una forma en "U" compuesta por dos fases:

- **Codificación (encoder)**: reduce progresivamente el tamaño de la imagen aplicando convoluciones y submuestreo (stride o pooling), extrayendo características globales.
- **Decodificación (decoder)**: reconstruye la resolución original mediante upsampling, refinando la predicción en cada paso.

Un elemento clave son las **conexiones de salto** (*skip connections*), que copian los mapas de características del encoder y los concatenan con los del decoder en cada nivel. Esto permite recuperar detalles finos que se habrían perdido en el proceso de compresión, logrando una segmentación precisa, especialmente en bordes y estructuras pequeñas como los tumores.

2.3. Esquema del pipeline general

A continuación se presenta el flujo de trabajo general del sistema:



3. Implementación

3.1. Obtención y preparación del dataset

Para llevar a cabo la tarea de segmentación de tumores en imágenes de ecografía mamaria, se ha utilizado el conjunto de datos público `Breast Ultrasound Images Dataset (Dataset BUSI)`, disponible en la página de Aly Fahmy[1]. Este dataset está compuesto por un total de 780 imágenes ecográficas divididas en tres categorías clínicas: *benignas*, *malignas* y *normales*, junto con su correspondiente máscara de segmentación manual [6].

Para el tratamiento de las imágenes del dataset, lo que se ha hecho en nuestro código ha sido, en primer lugar, organizar el conjunto de datos localmente en una estructura de carpetas separadas por clases. Y en segundo lugar, se estandariza el tamaño de las imágenes y sus

[1]Dataset BUSI

máscaras, de forma que todas compartan las mismas dimensiones. Esta homogeneización se lleva a cabo con el objetivo de introducir los datos en el modelo de segmentación, ya que dicho modelo necesita entradas con dimensiones fijas.

Para hacer más fácil todo este proceso de carga, transformación e integración de datos en el entrenamiento de la red neuronal, se creó una clase llamada `BUSIDataset`. Esta clase se encuentra en el archivo `train_unet_monai.py`, y permite emparejar cada imagen con su máscara correspondiente, redimensionarlas, normalizarlas y convertirlas en tensores compatibles con PyTorch. Para más información sobre `BUSIDataset` ir a la sección `Modelo de segmentación y entrenamiento`, donde se explica con mayor detalle.

Como conclusión de este apartado, decir que, tanto la preparación de datos, como la clase `BUSIDataset`, han sido desarrolladas por nosotros mismos adaptándonos de forma específica a las necesidades de segmentación en imágenes de ecografía mamaria.

3.2. Selección de herramientas

Para el desarrollo de esta parte del código hemos usado varias librerías de Python:

- **NumPy (numpy)** [7]
  - *¿Qué es?:* Librería para trabajar con matrices y realizar operaciones matemáticas de forma rápida y eficiente en Python.
  - *¿Para qué la usamos?:* Para normalizar los valores de las imágenes, transformándolos de 0–255 a 0–1, realizar conversiones de tipos de datos y cálculos matemáticos como el producto o suma de matrices y la obtención de estadísticas necesarias para el preprocesamiento y la evaluación.
- **OpenCV (cv2)** [8]
  - *¿Qué es?:* Librería usada para el procesamiento de imágenes.
  - *¿Para qué la usamos?:* Para leer las imágenes y sus correspondientes máscaras desde archivos en formato PNG, redimensionar las imágenes a un tamaño fijo que pueda ser procesado por la red neuronal (por ejemplo, 256x256 píxeles), aplicar filtros clásicos (Butterworth y mediana), convertir imágenes a escala de grises y fusionar canales para crear imágenes RGB artificiales.
- **PyTorch (torch)** [9]
  - *¿Qué es?:* Librería de deep learning utilizada para construir y entrenar redes neuronales.
  - *¿Para qué la usamos?:* Para definir tensores que representan imágenes y máscaras, crear el modelo de red neuronal, gestionar el entrenamiento mediante la retropropagación y actualización de pesos, mover los datos a GPU si está disponible, y guardar/cargar modelos entrenados.
- **Matplotlib (matplotlib)** [10]
  - *¿Qué es?:* Librería para crear gráficos y visualizar datos.

- *¿Para qué la usamos?:* Para guardar comparativas visuales entre las imágenes originales, las máscaras reales y las predicciones generadas por el modelo. También se utiliza para representar los distintos pasos del preprocesamiento de forma gráfica.

- **glob y os (glob, os)** [11] [12]
  - *¿Qué son?:*
    - *glob* es un módulo que permite buscar archivos siguiendo patrones de nombre (por ejemplo, todos los archivos .png de una carpeta).
    - *os* es un módulo estándar de Python para trabajar con rutas de archivos y carpetas.
  - *¿Para qué los usamos?:* Para buscar automáticamente las imágenes y sus máscaras dentro de las carpetas del dataset, recorrer directorios, crear nuevas carpetas para guardar resultados (como predicciones y comparaciones) y construir rutas de archivos dinámicamente.

- **MONAI (monai)** [13]
  - *¿Qué es?:* MONAI (Medical Open Network for AI) es una librería construida sobre PyTorch, especializada en el procesamiento y análisis de imágenes médicas. Ha sido desarrollada en colaboración con expertos médicos y de inteligencia artificial para facilitar el desarrollo de modelos de Deep Learning en el ámbito médico.
  - *¿Qué ofrece MONAI?:* Modelos predefinidos específicamente diseñados para imágenes médicas (por ejemplo, U-Net, V-Net, SegResNet). Funciones de pérdida específicas para segmentación médica (como DiceLoss). Métricas de evaluación adaptadas a problemas de segmentación (como DiceMetric). Transformaciones y utilidades específicas para imágenes de tomografías, resonancias magnéticas, ecografías, etc. Herramientas para asegurar resultados reproducibles.
  - *¿Para qué la usamos?:*
    - *UNet:* Se utiliza para construir directamente el modelo de red neuronal de segmentación.
    - *DiceLoss:* Funciona como medida de error durante el entrenamiento, favoreciendo que las segmentaciones predichas se parezcan mucho a las reales.
    - *DiceMetric:* Permite calcular la calidad de las segmentaciones de forma precisa tras cada época de entrenamiento.
    - *set_determinism:* Se usa para asegurar que los resultados sean reproducibles, fijando una semilla aleatoria en las operaciones.
  - *¿Por qué se ha elegido MONAI?:* Aunque PyTorch podría cubrir las necesidades básicas, MONAI simplifica mucho el trabajo para tareas médicas:
    - Reduce el número de líneas de código necesarias.
    - Ofrece componentes optimizados y validados en aplicaciones médicas reales.
    - Mejora la precisión y estabilidad de los modelos sin necesidad de implementaciones manuales complicadas.

Por tanto, gracias a MONAI, podemos centrarnos más en el preprocesamiento, la evaluación de los resultados y la comparativa entre usar o no el preprocesamiento desarrollado, que en programar de forma detallada la U-Net usada.

- **scikit-image y scipy** [14, 15]
  - *¿Qué son?:* Librerías utilizadas para el análisis y procesamiento de imágenes científicas.
  - *¿Para qué las usamos?:* scikit-image se ha utilizado para calcular la entropía local en las texturas de primer orden. scipy se ha usado para aplicar transformadas de Fourier necesarias en el filtro Butterworth.
- **scikit-learn** [16]
  - *¿Qué es?:* Librería usada para aprendizaje automático, también ofrece herramientas estadísticas.
  - *¿Para qué la usamos?:* Para calcular métricas de evaluación como la precisión, sensibilidad (recall), especificidad, matriz de confusión y exactitud de las predicciones generadas por el modelo en la fase de evaluación.

3.3. `Preprocesamiento de imágenes`

En el preprocesamiento de imágenes, buscamos preparar los datos de entrada para mejorar la calidad de la segmentación. Para ello, hemos implementado diferentes técnicas para reducir el ruido, resaltar contornos y extraer información estructural. Todo esto se hace en la clase `preprocessing.py`:

- **butterworth_high_pass(img):** Esta función aplica un filtro pasa-altos de Butterworth, que atenúa las componentes de baja frecuencia y resalta los bordes al usar transformadas de Fourier e inversas [17]. Este método ayuda a mejorar la visibilidad de los bordes tumorales difusos en las imágenes ecográficas.
- **adaptive_median_filter(img):** Esta función utiliza un filtro de mediana adaptativa para eliminar el ruido tipo sal y pimienta, ajustando dinámicamente el tamaño de la ventana de filtrado y preservando mejor los bordes que un filtro de media [18].
- **first_order_features(img):** Extrae media local y entropía local de la imagen. La media suaviza regiones homogéneas y la entropía mide la complejidad local (cantidad de información) en la vecindad de cada píxel, útil para resaltar zonas de alta variabilidad como lesiones [19].
- **second_order_features_fallback(img):** Emplea el detector de bordes de Canny y el filtro Laplaciano como alternativa a GLCM. Canny encuentra contornos mediante gradientes y supresión de no-máximos, mientras Laplaciano realza regiones de cambio abrupto de intensidad [20].

- **fuse_channels(ch1, ch2, ch3):** Combina tres canales en una imagen RGB artificial mediante fusión de matrices, permitiendo codificar en cada canal información distinta (intensidad, textura, bordes) para enriquecer la entrada del modelo.
- **process_image(img_path, output_dir):** Aplica de forma ordenada todas las técnicas de preprocesamiento anteriores, guarda las versiones resultantes con sufijos descriptivos y genera una comparativa visual de los efectos de cada método de preprocesamiento.

El resultado es un conjunto de imágenes derivadas, una por técnica, que sirven de entrada a la red de segmentación. Esto permite comparar objetivamente el impacto de cada método en el rendimiento de detección de tumores.

3.4. Modelo de segmentación y entrenamiento

En esta parte del proyecto, buscamos construir y entrenar una red neuronal profunda para segmentar las imágenes de ecografía, tanto preprocesadas como sin preprocesar. Todo esto se implementa en el archivo `train_unet_monai.py`, y utiliza la arquitectura U-Net ofrecida por la librería MONAI [21]. Cabe destacar que usamos la arquitectura U-Net, porque es muy utilizada en el ámbito de la imagen médica debido a su capacidad para aprender tanto el contexto global como los detalles locales de la imagen mediante una estructura simétrica de codificador y decodificador con conexiones de salto. Estas conexiones permiten recuperar información espacial perdida durante las operaciones de reducción de resolución, lo cual es fundamental en tareas como la segmentación de tumores, donde los bordes y la localización precisa son críticos [22].

- **Definición del conjunto de datos (clase BUSIDataset):** Se crea una clase personalizada que extiende de `TorchDataset` (PyTorch) y recibe dos listas: rutas a imágenes y rutas a sus máscaras correspondientes. En el método `__getitem__` se leen ambas, se convierten a escala de grises y se redimensionan a tamaño fijo (256x256). Además, se normalizan los valores de píxeles entre 0 y 1 y se convierten en tensores [23].
- **Construcción del modelo (MONAI UNet):** Se define una red neuronal convolucional U-Net con bloques encoder-decoder y conexiones de salto, adaptada para segmentación binaria. La arquitectura tiene cuatro niveles de profundidad, canales crecientes (16 a 128), y unidades residuales en cada nivel. Está diseñada para trabajar con imágenes de entrada de un solo canal (escala de grises) y producir una única máscara binaria como salida [21].
- **Función de pérdida (DiceLoss):** Para entrenar el modelo, se utiliza la función `DiceLoss` provista por MONAI, una métrica común en segmentación médica que mide la similitud entre la máscara real y la predicha. Esta función penaliza especialmente los errores en clases desbalanceadas, como suelen ser los tumores respecto al fondo [24].
- **Proceso de entrenamiento:** Durante 30 épocas, el modelo se entrena sobre lotes de imágenes cargadas mediante un `DataLoader` de MONAI. En cada iteración, se realiza una pasada hacia adelante, se calcula la pérdida con `DiceLoss`, se retropropaga el error y se actualizan los pesos usando el optimizador Adam. Se imprime la pérdida media por época para observar la convergencia [25].

- **Reproducibilidad (set_determinism):** Se utiliza la función `set_determinism()` de MONAI para fijar las semillas aleatorias de NumPy, PyTorch y MONAI, garantizando que los resultados sean reproducibles en diferentes ejecuciones [26].
- **Guardado del modelo y visualización:** Una vez finalizado el entrenamiento, el modelo se guarda en disco y se generan predicciones de muestra sobre el mismo conjunto de entrenamiento. Las imágenes originales, las máscaras reales y las predichas se guardan en formato imagen para su comparación visual.

Gracias al uso de MONAI, esta parte del desarrollo se ha simplificado considerablemente en comparación con una implementación manual de U-Net desde cero, lo que ha permitido centrarse en el análisis del preprocesamiento y en la evaluación experimental.

3.5. Evaluación de resultados

Una vez que hemos entrenado el modelo de segmentación, es necesario evaluar su rendimiento para determinar si el preprocesamiento aplicado sobre las imágenes mejora, mantiene o incluso empeora los resultados obtenidos. Esta evaluación se ha hecho en el archivo `metrics_evaluation.py`. Además, las métricas seleccionadas son las mismas que se usan en el estudio de referencia [27].

- **Dice Similarity Coefficient (DSC):** Esta métrica evalúa la superposición entre dos conjuntos de píxeles segmentados. Toma valores entre 0 (no hay coincidencia) y 1 (coincidencia perfecta). Es especialmente útil en segmentación médica, donde las clases están desbalanceadas y el área tumoral es pequeña en relación al fondo.
- **IoU (Intersection over Union):** También conocida como Jaccard Index, mide la relación entre la intersección y la unión de los píxeles predichos y reales. Es más estricta que Dice y complementaria para analizar la calidad de segmentación.
- **Accuracy global:** Proporción de píxeles correctamente clasificados sobre el total de píxeles. Aunque es útil como referencia, puede resultar engañosa si la clase de fondo domina ampliamente sobre la clase tumoral.
- **Precisión (Precision):** Proporción de verdaderos positivos respecto al total de positivos predichos. Mide cuántas de las regiones predichas como tumor realmente lo son.
- **Sensibilidad (Recall):** También conocida como *True Positive Rate* o *Recall*, mide la proporción de píxeles tumorales correctamente detectados respecto al total real de píxeles tumorales. Es crítica para no omitir zonas afectadas.
- **Especificidad:** Mide la proporción de píxeles de fondo correctamente clasificados como tal. Es útil para asegurar que el modelo no sobresegmenta fuera de la región tumoral.

La evaluación se realiza recorriendo todas las máscaras predichas por el modelo y comparándolas con las correspondientes máscaras reales. Previamente, ambas imágenes se binarizan, redimensionan si es necesario y se comparan píxel a píxel. Las métricas se calculan

usando funciones de `NumPy` y `scikit-learn`, y los resultados se promedian para ofrecer una visión general del rendimiento del modelo.

Esta evaluación se ha hecho con el objetivo principal de poder comparar los resultados obtenidos con diferentes versiones de imágenes (sin preprocesar, con filtros clásicos y con texturas), permitiendo ver de forma cuantitativa el efecto que cada técnica de preprocesamiento tiene sobre la segmentación automática.

4. Experimentación

4.1. Objetivo de la experimentación

El objetivo principal es evaluar el impacto de diferentes técnicas de **preprocesamiento** sobre el rendimiento de un modelo de segmentación de tumores en imágenes ecográficas. Se compara el rendimiento de la red U-Net entrenada con imágenes originales frente a aquellas que han sido sometidas a distintos métodos de realce y filtrado de textura.

4.2. Configuración experimental

Se ha utilizado el conjunto de datos **BUSI** (Breast Ultrasound Images), que contiene imágenes clasificadas en tres categorías: benignas, malignas y normales. La arquitectura empleada para la segmentación ha sido **U-Net**, entrenada con las siguientes condiciones constantes:

- **Tamaño de entrada:** 256×256 píxeles
- **Épocas:** 30
- **Función de pérdida:** DiceLoss
- **Evaluación:** sobre imágenes originales y preprocesadas

La variable experimental es el **tipo de preprocesamiento** aplicado a las imágenes.

4.3. Técnicas de preprocesamiento

Se han evaluado cinco configuraciones distintas:

- **Sin preprocesamiento:** imágenes originales del dataset.
- **Butterworth:** filtro pasa-altos en el dominio de la frecuencia, diseñado para resaltar bordes sin introducir ruido abrupto.
- **Filtro mediana adaptativa:** reduce el ruido tipo speckle característico de las imágenes ecográficas sin perder detalles de bordes.
- **Textura de primer orden:** basada en la media y entropía de vecindades locales, para representar intensidad y aleatoriedad.

- **Textura de segundo orden:** utiliza operadores como Canny y Laplaciano para aproximar características de bordes, autocorrelación y homogeneidad.

Las texturas se combinan con la imagen original formando una imagen **RGB tridimensional**, en la que cada canal representa una característica diferente (ej. canal rojo = original, verde = textura1, azul = textura2).

4.4. Métricas de evaluación

Se utilizan las siguientes métricas para evaluar el rendimiento del modelo:

- **Dice Coefficient (DSC):** mide la superposición entre la máscara predicha y la real. Cuanto más alto, mejor (ideal: 1).
- **IoU (Intersection over Union):** relación entre la intersección y la unión de las máscaras predicha y real.
- **Accuracy:** proporción de píxeles correctamente clasificados (positivos y negativos).
- **Precision:** proporción de verdaderos positivos entre todos los positivos predichos.
- **Recall (Sensibilidad):** capacidad del modelo para detectar todos los píxeles positivos reales.
- **Specificity:** capacidad para identificar correctamente los píxeles negativos.

4.5. Resultados

Para valorar el alcance de los resultados obtenidos, se ha realizado una comparación directa con el estudio de Cai et al. (2022). Ambos trabajos comparten el objetivo de mejorar la segmentación de tumores en imágenes de ecografía mamaria mediante técnicas de preprocesamiento, y utilizan arquitecturas de tipo **U-Net** para realizar la segmentación. Sin embargo, existen diferencias clave: mientras que el estudio de referencia implementa la red en TensorFlow, nuestro enfoque emplea la librería **MONAI**, desarrollada específicamente para aplicaciones clínicas sobre imágenes médicas. MONAI ofrece ventajas como transformaciones optimizadas, funciones de pérdida específicas como `DiceLoss`, métricas clínicas integradas y soporte directo para arquitecturas predefinidas, lo que permite un entrenamiento más estable y reproducible.

En la Tabla 1 se presentan los resultados comparativos. Se incluyen métricas fundamentales como el coeficiente de Dice (DSC), el índice de Jaccard (IoU), exactitud global (Accuracy), precisión, sensibilidad (Recall) y especificidad. Las técnicas evaluadas abarcan tanto filtros tradicionales (Butterworth, mediana) como métodos basados en características de textura de primer y segundo orden.

Cuadro 1: Comparativa de métricas entre nuestro trabajo y el estudio de Cai et al. (2022)

Técnica	DSC	IoU	Accuracy	Precision	Recall	Specificity
Ours - Sin preproc.	0.630	0.530	0.930	0.630	0.680	0.960
Ours - Butterworth	0.708	0.629	0.949	0.692	0.749	0.962
Ours - Mediana	0.728	0.660	0.959	0.734	0.734	0.971
Ours - Textura 1er orden	0.739	**0.674**	0.956	0.730	0.757	0.965
Ours - Textura 2º orden	0.735	0.668	0.958	0.726	0.758	0.969
Cai et al. - Butterworth HPF	0.720	0.479	0.859	0.587	0.743	0.872
Cai et al. - Mediana adapt.	0.722	0.481	0.865	0.597	0.750	0.884
Cai et al. - Textura 1er ord.	0.744	0.562	0.945	0.641	0.775	0.956
Cai et al. - Textura 2º ord.	**0.755**	0.602	**0.976**	**0.704**	**0.803**	**0.983**

También se han generado gráficas comparativas para visualizar el rendimiento de cada técnica en cada métrica, destacando la técnica ganadora por métrica.

4.6. Análisis y conclusiones

Los datos muestran que ambos enfoques coinciden en una conclusión clave: el preprocesamiento mejora de forma significativa la segmentación frente al uso de imágenes originales. Las técnicas basadas en textura —especialmente las de segundo orden— son las que alcanzan los mejores valores de DSC, IoU y sensibilidad, al capturar patrones estructurales relevantes en las imágenes ecográficas.

Comparando directamente, nuestros resultados con **textura de primer orden** logran un **IoU de 0.674**, que supera incluso al valor máximo alcanzado en el estudio de referencia (**0.602** con segundo orden), y presentan métricas de precisión y especificidad también más elevadas. Esto sugiere que el uso de MONAI ha permitido una integración más eficaz de las características texturales en el flujo de entrenamiento, posiblemente gracias a su gestión optimizada del pipeline médico y funciones de pérdida especializadas.

En cuanto al **DSC**, el estudio alcanza un valor ligeramente superior (0.755) con segundo orden, mientras que nuestro valor máximo es 0.739. Sin embargo, la diferencia es estrecha, y nuestras métricas asociadas (precisión, IoU, especificidad) muestran un equilibrio general más alto.

Por tanto, podemos concluir que el enfoque propuesto en este trabajo, basado en el uso de MONAI junto a preprocesamiento textural y arquitecturas U-Net, no solo confirma los hallazgos del estado del arte, sino que los mejora en aspectos clave como la superposición espacial (IoU) y la precisión global de la segmentación. Este resultado refuerza la aplicabilidad de MONAI como herramienta de referencia para tareas de segmentación médica avanzada.

5. Manual de Usuario

5.1. Requisitos del sistema

Para ejecutar el proyecto correctamente es necesario disponer de:

- **Python 3.8** o superior.
- **Sistema operativo**: Windows, Linux o macOS.
- **Hardware recomendado**:
  - CPU moderna.
  - (Opcional) GPU compatible con CUDA para acelerar el entrenamiento (recomendado).

5.2. Instalación de dependencias

Antes de ejecutar el código, se deben instalar las librerías necesarias. Desde la carpeta raíz del proyecto, ejecutar el siguiente comando:

```
pip install -r requirements.txt
```

Esto instalará automáticamente las dependencias necesarias:

- numpy
- opencv-python
- matplotlib
- scikit-image
- scipy
- torch
- monai

Nota: Si se dispone de una GPU y se desea aprovechar, se recomienda instalar `torch` con soporte CUDA siguiendo las instrucciones de `https://pytorch.org/`.

5.3. Estructura del proyecto

- `main.py`: Script principal para lanzar las distintas fases del proyecto.
- `preprocessing.py`: Funciones de preprocesamiento de imágenes.
- `train_unet_monai.py`: Entrenamiento del modelo U-Net.
- `predict_with_unet.py`: Predicción de máscaras sobre nuevas imágenes.
- `metrics_evaluation.py`: Evaluación de las predicciones mediante métricas.
- `unet_busi.pt`: Archivo que guarda el modelo U-Net entrenado.

5.4. Cómo utilizar el proyecto

5.4.1. Preprocesamiento de imágenes

Para aplicar los filtros y generar imágenes procesadas, ejecutar:

```
python main.py classic
```

Esto aplicará:

- Filtro de Butterworth.
- Filtro de mediana.
- Extracción de características de primer orden (media y entropía).
- Extracción de características de segundo orden (bordes y cambios locales).

Los resultados se guardarán en la carpeta de salida especificada.

5.4.2. Entrenamiento del modelo U-Net

Para entrenar el modelo desde cero, ejecutar:

```
python main.py train
```

Este comando:

- Preparará el conjunto de datos (imágenes y máscaras).
- Definirá y entrenará el modelo U-Net.
- Guardará el modelo entrenado en el archivo `unet_busi.pt`.

5.4.3. Predicción con el modelo entrenado

Para realizar predicciones sobre nuevas imágenes:

```
python main.py predict
```

Esto:

- Cargará el modelo entrenado.
- Aplicará la predicción a nuevas imágenes.
- Guardará las máscaras predichas y comparativas visuales.

5.4.4. Evaluación de resultados

Para evaluar la calidad de las predicciones:

```
python main.py evaluate
```

Se calcularán automáticamente las métricas:

- Dice Coefficient (DSC)
- Intersection over Union (IoU)
- Accuracy
- Precision
- Recall (Sensibilidad)
- Specificity

5.5. Parámetros configurables

El proyecto puede ajustarse mediante la modificación de ciertos parámetros definidos en el código. Estos son los parámetros más relevantes, su función, valor por defecto, ubicación en el código y consideraciones en caso de que se desee modificarlos:

- **Número de épocas de entrenamiento (EPOCHS)**
  Archivo: `train_unet_monai.py`
  Ubicación: línea que contiene `EPOCHS = 30`
  Descripción: define el número total de pasadas completas del conjunto de datos por la red neuronal durante el entrenamiento. Aumentarlo puede mejorar la precisión del modelo, pero también incrementa el tiempo de entrenamiento y riesgo de sobreajuste.

- **Tamaño del batch (BATCH_SIZE)**
  Archivo: `train_unet_monai.py`
  Descripción: cantidad de imágenes procesadas simultáneamente antes de actualizar los pesos del modelo. Valores mayores pueden acelerar el entrenamiento si se dispone de memoria suficiente en GPU.

- **Tamaño de imagen (IMG_SIZE)**
  Archivos: `train_unet_monai.py`, `predict_with_unet.py`
  Descripción: las imágenes de entrada y sus máscaras se redimensionan a este tamaño. Modificarlo puede afectar a la precisión y tiempo de entrenamiento. Por defecto es `(256, 256)`.

- **Tipo de preprocesamiento (PREPROC_TAG)**
  Archivos: `train_unet_monai.py`, `predict_with_unet.py`, `metrics_evaluation.py`
  Descripción: cadena que identifica qué tipo de imagen preprocesada se va a utilizar (`_butterworth`, `_median`, `_first_order`, etc.). Este sufijo debe coincidir con los nombres de archivos de imagen generados.

- **Rutas de entrada y salida**
  Archivo: `main.py`
  Parámetros: `test_dir`, `output_dir`, `gt_dir_root`, `pred_dir`
  Descripción: determinan dónde buscar las imágenes originales, las máscaras de ground truth y dónde guardar los resultados de las predicciones o evaluaciones.

- **Modo de ejecución**
  Archivo: `main.py`
  Descripción: se controla con el valor de la variable `mode`. Los valores posibles son:

  - `çlassic"`: aplicar preprocesamiento a todas las imágenes.
  - `"train"`: entrenar el modelo desde cero.
  - `"predict"`: generar predicciones con el modelo entrenado.
  - `.evaluate"`: calcular métricas a partir de predicciones y máscaras reales.

  Nota: cualquier otro valor provocará un mensaje de error.

Recomendación: antes de modificar cualquier parámetro, se aconseja realizar una copia de seguridad del archivo correspondiente y asegurarse de mantener la coherencia en las rutas y nombres utilizados en distintas fases del proyecto.

6. Conclusiones

Este proyecto ha tenido como objetivo mejorar la segmentación automática de tumores en imágenes de ecografía mamaria, aplicando diferentes técnicas de preprocesamiento basadas en textura y usando una red neuronal U-Net entrenada con la librería MONAI. Las ecografías son un tipo de imagen médica especialmente difícil de procesar debido al ruido, al bajo contraste y a la variabilidad entre pacientes. Por eso, era importante estudiar si aplicar transformaciones antes del entrenamiento podía ayudar a obtener mejores resultados.

Después de entrenar la red con imágenes originales y con versiones preprocesadas, se evaluaron los resultados usando métricas estándar como el coeficiente de Dice (DSC), el índice de Jaccard (IoU), la precisión, la sensibilidad (recall), la especificidad y la exactitud general. En todos los casos, los resultados fueron mejores cuando se aplicó preprocesamiento, especialmente con las técnicas de textura de primer y segundo orden. Estas transformaciones ayudaron a resaltar información importante como los bordes del tumor o zonas con mayor variación de intensidad.

Comparando con estudios anteriores, nuestro sistema logró resultados similares o incluso mejores en métricas como el IoU y la precisión, especialmente gracias al uso de MONAI. Esta librería, pensada para imágenes médicas, permitió construir y entrenar la red U-Net de forma más eficiente, reproducible y con menos código. MONAI nos permitió centrarnos en el análisis de datos y en la mejora de la segmentación, sin tener que preocuparnos tanto por los detalles técnicos de bajo nivel.

En resumen, hemos comprobado que aplicar preprocesamiento basado en textura mejora la segmentación de tumores respecto a no aplicar ningún tratamiento previo. Además, el uso

de herramientas especializadas como MONAI facilita el desarrollo de sistemas de segmentación robustos y eficaces. Estos resultados son prometedores y abren la puerta a aplicar esta misma estrategia en otros tipos de imágenes médicas, como resonancias o TACs.

Como posibles líneas futuras, se podrían probar otras arquitecturas más avanzadas (como Attention U-Net), aplicar aumentos de datos (data augmentation), trabajar con imágenes en 3D o incluso probar el sistema en entornos clínicos reales, validando los resultados con profesionales médicos.

7. Autoevaluacion de cada miembro

	Juan	**Mar**	**Nerea**
Comprensión y dominio	Excelente	Excelente	Excelente
Exposición didáctica	Excelente	Excelente	Excelente
Integración del equipo	Excelente	Excelente	Excelente
Objetivos	Excelente	Excelente	Excelente
Aspectos didácticos	Excelente	Excelente	Excelente
Experimentación y conclusiones	Excelente	Excelente	Excelente
Contenidos	Excelente	Excelente	Excelente
Divulgación de los contenidos	Excelente	Excelente	Excelente
Bibliografía. Recursos científicos	Excelente	Excelente	Excelente

Cuadro 2: Evaluación entre compañeros

8. Tabla de tiempos

Fecha de la actividad	Tiempo (h)	Miembro	Actividad realizada
18/02/2025	2 h 06 min	Nerea	Búsqueda de tema y referencia bibliográfica
18/02/2025	11 min	Nerea	Búsqueda y revisión de repositorio
25/02/2025	1 h 28 min	Nerea	Creación, redacción y revisión del documento para el Entregable 1

Fecha de la actividad	Tiempo (h)	Miembro	Actividad realizada
28/02/2025	2 h 39 min	Nerea	Investigación sobre los tumores mamarios
10/03/2025	1 h 47 min	Nerea	Revisión y análisis del documento .ipynb (primer script)
26/02/2025 4/03/2025 y 11/03/2025	5 h 30 min	Nerea	Clases dedicadas al proyecto
11/03/2025	20 min	Nerea	Creación, redacción y revisión del documento para el Entregable 2
1/04/2025 4/04/2025 y 7/04/2025	7 h 37 min	Nerea	Investigación sobre MONAI y redes U-net
7/04/2025 y 8/04/2025	3 h 24 min	Nerea	Redacción del documento LaTeX: primer borrador
1/04/2025, 2/04/2025 5/04/2025 y 8/04/2025	10 h 26 min	Nerea	Implementación seguimiento 3: revisión de métodos del preprocesamiento y entrenamiento de la U-net
1/04/2025 5/04/2025 y 8/04/2025	6 h 11 min	Nerea	Implementación seguimiento 3: pruebas de ajustes del modelo
8/04/2025	1 h 02 min	Nerea	Creación, redacción y revisión del documento para el Entregable 3
29/04/2025	1 h 32 min	Nerea	Creación, redacción y revisión del documento para el Entregable 4
18/04/2025 28/04/2025	3 h 24 min	Nerea	Revisión de correcciones de la profesora y resolución de dudas internas sobre el proyecto y MONAI
18/04/2025 20/04/2025 28/04/2025	6 h 36 min	Nerea	Redacción y corrección del documento LaTeX con base en la implementación, información aprendida e investigación
29/04/2025	2 h	Nerea	Presentación: borrador
11/05/2025 12/05/2025	4 h 45 min	Nerea	Revisión de documento LaTex, pulir lo escrito y preparar para el entregable y guión
11/05/2025 12/05/2025	5 h 38 min	Nerea	Presentación: elaboración de diapositivas y guión
20/05/2025	2 h	Nerea	Practicar presentación
18/02/2025	2 h 24 min	María del Mar	Búsqueda de tema y referencia bibliográfica

Fecha de la actividad	Tiempo (h)	Miembro	Actividad realizada
18/02/2025	52 min	María del Mar	Búsqueda y revisión de repositorio
25/02/2025	1 h 13 min	María del Mar	Creación, redacción y revisión del documento para el Entregable 1
27/02/2025	3 h 45 min	María del Mar	Investigación y comprensión de modelos de segmentación
01/03/2025	6 h 12 min	María del Mar	Primera implementación
26/02, 4/03 y 11/03/2025	5 h 30 min	María del Mar	Clases dedicadas al proyecto
11/03/2025	13 min	María del Mar	Creación, redacción y revisión del documento para el Entregable 2
01/04/2025	4 h 11 min	María del Mar	Investigación sobre MONAI
02/04/2025	4 h 56 min	María del Mar	Investigación sobre redes U-Net
03/04/2025	2 h	María del Mar	Implementación de filtros clásicos en OpenCV (Butterworth, mediana)
03/04/2025	4 h 02 min	María del Mar	Extracción de características texturales (entropía, bordes, laplaciano)
04/04/2025	3 h 45 min	María del Mar	Desarrollo del pipeline de preprocesamiento completo
05/04/2025	6 h	María del Mar	Entrenamiento del modelo UNet con MONAI
06/04/2025	4 h 30 min	María del Mar	Creación del script de predicción y visualización
06/04/2025	2 h 15 min	María del Mar	Pruebas y experimentación con diferentes épocas y ajustes
08/04/2025	20 min	María del Mar	Documento para el Entregable 3
29/04/2025	2 h 17 min	María del Mar	Redacción del punto de manual de usuario en LaTeX
29/04/2025	10 min	María del Mar	Inclusión del punto de experimentación en el documento
29/04/2025	1 h 30 min	María del Mar	Redacción inicial del apartado de implementación en LaTeX
29/04/2025	40 min	María del Mar	Documento para el Entregable 4
06/05/2025	1 h 15 min	María del Mar	Continuación de redacción del apartado de implementación en LaTeX
08/05/2025	2 h	María del Mar	Modificación del manual de usuario para indicar configuraciones y estructura
11/05/2025	1 h	María del Mar	Revisión final y edición del manual de usuario en el README
12/05/2025	1 h	María del Mar	Actualización de enlace al dataset y últimos ajustes en el README

Fecha de la actividad	Tiempo (h)	Miembro	Actividad realizada
20/05/2025	2 h	María del Mar	Práctica de presentación
18/02/2025	2 h	Juan	Búsqueda de tema y referencia bibliográfica.
20/02/2025	4 h	Juan	Investigación y comprensión de filtrado y transformaciones de intensidad
20/02/2025	2 h	Juan	Investigación sobre filtros gaussianos y filtros medianos
21/02/2025	1 h 15 min	Juan	Investigación sobre redes neuronales
21/02/2025	45 min	Juan	Búsqueda de datasets
25/02/2025	11 min	Juan	Creación, redacción y revisión del documento para el Entregable 1
26/02, 4/03 y 11/03/2025	5 h 30 min	Juan	Clases dedicadas al proyecto
11/03/2025	20 min	Juan	Creación, redacción y revisión del documento para el Entregable 2
03/04/2025	5 h	Juan	Implementación de filtros clásicos (Butterworth, mediana) en OpenCV
03/04/2025	7 h	Juan	Implementación de extracción de características texturales (entropía, bordes, laplaciano)
04/04/2025	3 h 45 min	Juan	Desarrollo del pipeline de preprocesamiento clásico completo
05/04/2025	2 h 30 min	Juan	Búsqueda de máscaras de imágenes del dataset usado
05/04/2025	9 h	Juan	Entrenamiento del modelo UNet con MONAI
06/04/2025	4 h 30 min	Juan	Creación del script de predicción y visualización
06/04/2025	2 h 15 min	Juan	Pruebas y experimentación con diferentes épocas y ajustes
08/04/2025	30 min	Juan	Creación, redacción y revisión del documento para el Entregable 3
09/04/2025	3 h 15 min	Juan	Desarrollo de script para cálculo de métricas
10/04/2025	2 h 45 min	Juan	Adaptación del entrenamiento y predicción de la U-Net para procesar resultados con distintos preprocesamientos
11/04/2025	1 h 30 min	Juan	Debugging, pruebas de rutas, resolución de errores o formatos de nombres
12/04/2025	45 min	Juan	Análisis y comparación de resultados obtenidos e identificación de mejores técnicas

Fecha de la actividad	Tiempo (h)	Miembro	Actividad realizada
28/04/2025	30 min	Juan	Redacción de la sección de Experimentación del informe final
29/04/2025	15 min	Juan	Creación, redacción y revisión del documento para el Entregable 4
02/05/2025	2 h 45 min	Juan	Entrenamiento adicional de modelos U-Net para comparar preprocesamientos
04/05/2025	3 h 30 min	Juan	Comparación de resultados, redacción y revisión completa de la sección de Experimentación del informe final
05/05/2025 y 12/05/2025	1 h 15 min	Juan	Revisión y correcciones gramaticales y de estilo
20/05/2025	2 h	Juan	Ensayo y práctica completa de la presentación

Cuadro 3: Tiempo dedicado al proyecto

Referencias

[1] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). *A survey on deep learning in medical image analysis*. Medical image analysis, 42, 60-88. https://doi.org/10.1016/j.media.2017.07.005

[2] Haralick, R. M., Shanmugam, K., Dinstein, I. (1973). *Textural Features for Image Classification*. IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610–621. https://ieeexplore.ieee.org/document/4309314

[3] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/

[4] Ronneberger, O., Fischer, P., Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In MICCAI. Disponible en: https://arxiv.org/abs/1505.04597

[5] Taha, A. A., Hanbury, A. (2015). *Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool*. BMC Medical Imaging, 15(1), 1–28. https://doi.org/10.1186/s12880-015-0068-x

[6] Fahmy, A. (s.f.). *Dataset*. [Online]. Disponible en: https://scholar.cu.edu.eg/?q=afahmy/pages/dataset

[7] NumPy Developers, *NumPy Documentation*, [Online]. Disponible en: https://numpy.org/doc/stable/

[8] OpenCV Team, *OpenCV Documentation*, [Online]. Disponible en: https://docs.opencv.org/4.x/index.html

[9] PyTorch Developers, *PyTorch Documentation*, [Online]. Disponible en: https://pytorch.org/docs/stable/index.html

[10] Matplotlib Developers, *Using Matplotlib*, [Online]. Disponible en: https://matplotlib.org/stable/contents.html

[11] Python Software Foundation, *glob — Unix style pathname pattern expansion*, [Online]. Disponible en: https://docs.python.org/3/library/glob.html

[12] Python Software Foundation, *os — Miscellaneous operating system interfaces*, [Online]. Disponible en: https://docs.python.org/3/library/os.html

[13] MONAI Consortium, *Project MONAI*, [Online]. Disponible en: https://docs.monai.io/

[14] scikit-image Developers, *scikit-image Documentation*, [Online]. Disponible en: https://scikit-image.org/docs/stable/

[15] SciPy Community, *SciPy Documentation*, [Online]. Disponible en: https://docs.scipy.org/doc/scipy/

[16] scikit-learn Developers, *User Guide*, [Online]. Disponible en: https://scikit-learn.org/stable/user_guide.html

[17] MathWorks, *Filtro paso alto*, [Online]. Disponible en: https://es.mathworks.com/discovery/high-pass-filter.html

[18] J. C. Gutiérrez López, *Filtro de Mediana Adaptativo*, Scribd, [Online]. Disponible en: https://es.scribd.com/document/43929424/Filtro-de-Mediana-Adaptativo

[19] scikit-image, *Entropy*, [Online]. Disponible en: https://scikit-image.org/docs/stable/auto_examples/filters/plot_entropy.html

[20] Programar Fácil, *Detector de bordes Canny, cómo contar objetos con OpenCV y Python*, [Online]. Disponible en: https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/

[21] MONAI Consortium, *UNet - MONAI Network Architectures*, [Online]. Disponible en: https://docs.monai.io/en/stable/networks.html#unet

[22] P. Alzamora, *Segmentación semántica de imágenes con Deep Learning*, Damavis Blog, 10 de marzo de 2023. [Online]. Disponible en: https://blog.damavis.com/segmentacion-semantica-de-imagenes-con-deep-learning/

[23] PyTorch Developers, *Writing Custom Datasets, DataLoaders and Transforms*, [Online]. Disponible en: https://pytorch.org/tutorials/beginner/data_loading_tutorial.html

[24] MONAI Consortium, *Loss Functions - DiceLoss*, [Online]. Disponible en: https://docs.monai.io/en/stable/losses.html#diceloss

[25] PyTorch Developers, *torch.optim — Optimizers*, [Online]. Disponible en: `https://pytorch.org/docs/stable/optim.html`

[26] MONAI Consortium, *Utilities*, [Online]. Disponible en: `https://docs.monai.io/en/stable/utils.html#monai.utils.set_determinism`

[27] Cai, S., Zhang, J., Zhu, Y., & Liu, T. (2022). .ᴬ Study on the Combination of Image Preprocessing Method Based on Texture Feature and Segmentation Algorithm for Breast Ultrasound Images". In: *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, IEEE.