Joaquín Luna Barco

Desarrollo de un Modelo de Segmentación Automática para la Detección de Lesiones de Cáncer de Mama en Imágenes de Resonancia Magnética

Trabajo de Fin de Grado

Dirigido por Alfonso José Romero Nevado

Grado de Ingeniería Biomédica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2024

Índice

1 Introducción 1

1.1 Motivación 1

1.2 Cáncer de mama 2

1.3 Segmentación de Imágenes 3

1.4 Objetivos 3

1.5 Organización del documento 4

2 Marco Teórico 5

2.1 Características Anatómicas del Seno y su Impacto en la Interpretación de Imágenes RM 5

2.2 Redes Neuronales 7

2.2.1 *¿Qué son las redes neuronales?* 7

2.2.2 *Componentes y funcionamiento* 8

2.2.3 *Aprendizaje supervisado y no supervisado* 10

2.2.4 *Redes Convolucionales (CNN)* 11

2.2.5 *Modelo U-Net* 17

2.2.6 *Entrenamiento de Redes Neuronales* 19

3 Estado del Arte 26

4 Metodología 28

4.1 Tecnologías y Herramientas 28

4.2 Conjunto de datos 30

4.2.1 *Recopilación de Datos* 30

4.2.2 *Preprocesamiento* 31

4.3 Diseño del Modelo 38

4.4 Entrenamiento 40

4.4.1 *Optimizador* 40

4.4.2 *Función de Pérdida* 41

4.4.3 *Callbacks* 41

4.4.4 *Implementación de Técnicas de Regularización* 42

4.4.5 *Modelos entrenados* 42

4.4.6 *Métricas de Evaluación* 42

5 Resultados 44
5.1 Métricas y configuración 44
5.1.1 *Modelo 1* 44
5.1.2 *Modelo 2* 45
5.1.3 *Modelo 3* 47
5.1.4 *Modelo 4* 48
5.1.5 *Modelo 5* 49
5.1.6 *Modelo 6* 51
5.2 Predicciones 53
6 Conclusiones 60

1 Introducción

1.1 Motivación

Mi camino hacia el desarrollo de un modelo de segmentación automática de lesiones de cáncer de mama en imágenes de resonancia magnética (RM) fue una combinación única de experiencia profesional y ambición personal.

Comencé mi carrera médica como técnico de rayos, lo que me ayudó a apreciar la importancia de las imágenes médicas en el diagnóstico y tratamiento de enfermedades, especialmente el cáncer de mama.

Originalmente quería ser radiólogo, pero las circunstancias me llevaron a buscar otras formas de hacer una contribución significativa en el campo médico.

La ingeniería biomédica se presentó como una alternativa que combina mi pasión por la tecnología y mi deseo de impactar positivamente la salud.

Este TFG simboliza la conexión entre mis experiencias pasadas y mi ambición de apoyar la detección y el tratamiento del cáncer de mama.

Al desarrollar un modelo que utiliza inteligencia artificial para realizar las funciones de un radiólogo, espero no solo avanzar en mi carrera, sino también brindar herramientas que mejoren las tasas de detección temprana y el tratamiento efectivo del cáncer de mama.

Este proyecto resume mi transición de técnico de rayos a ingeniero biomédico y representa un paso adelante para contribuir a la salud y el bienestar del paciente a través de aplicaciones innovadoras de la tecnología en la atención médica.

1.2 Cáncer de mama

El cáncer de mama se define como un tipo de cáncer que se origina en el tejido mamario, afectando principalmente a las mujeres, aunque también puede afectar a los hombres en casos más raros. Se caracteriza por el crecimiento descontrolado de células en el seno, las cuales pueden formar un tumor que se siente como un bulto. Estos tumores cancerosos pueden crecer y eventualmente propagarse a los ganglios linfáticos cercanos y a otras partes del cuerpo a través del proceso de metástasis, lo que puede comprometer la vida del paciente si no se detecta y trata a tiempo.[1]

A nivel global, el cáncer de mama es una preocupación mayor de salud pública, con aproximadamente 2,3 millones de casos diagnosticados y 670.000 defunciones en mujeres en 2022. Esta enfermedad es un desafío en todos los países, independientemente de su nivel de desarrollo, aunque se observan grandes variaciones en las tasas de incidencia y mortalidad. En países con un alto índice de desarrollo humano (IDH), el cáncer de mama se diagnostica con mayor frecuencia en comparación con aquellos de bajo IDH, reflejando diferencias significativas en la carga de la enfermedad a nivel mundial. [1]

El cáncer de mama puede afectar a cualquier mujer a partir de la pubertad, pero su incidencia aumenta con la edad. Factores como el envejecimiento, la obesidad, el consumo de alcohol, antecedentes familiares de cáncer de mama, y ciertas mutaciones genéticas, como las de los genes BRCA1 y BRCA2, aumentan el riesgo de desarrollar esta enfermedad. Es fundamental la detección temprana para mejorar los resultados del tratamiento, ya que muchos casos no presentan síntomas en las etapas iniciales [1].

El tratamiento se personaliza basándose en las características del paciente, el tipo y estadio del cáncer, y puede incluir cirugía, radioterapia, y medicación, como hormonoterapia, quimioterapia, y tratamientos biológicos específicos. La eficacia del tratamiento mejora significativamente con una detección temprana [2].

1.3 Segmentación de Imágenes

La segmentación de imágenes médicas se establece como un proceso crítico dentro del análisis de imágenes diagnósticas, donde su principal función es dividir la imagen en partes o regiones significativas para facilitar su posterior análisis. Este procedimiento es de especial relevancia en el ámbito de la detección y evaluación de enfermedades, como es el caso del cáncer de mama, donde la precisión en la identificación y delimitación de lesiones puede ser determinante en las decisiones de tratamiento y pronóstico del paciente. [3]

La segmentación es el primer paso hacia la extracción de características relevantes de las imágenes médicas, desde las más sencillas como pueden ser la forma, el tamaño, y la ubicación de una lesión hasta más complejas a partir de algoritmos computacionales o la extracción masiva de características por medio de modelos de inteligencia artificial. Este proceso es fundamental para aplicaciones que van desde el diagnóstico asistido hasta la planificación quirúrgica y el monitoreo de la evolución de la enfermedad. La segmentación, por tanto, juega un papel crucial en el apoyo a los especialistas para realizar diagnósticos más precisos y rápidos, mejorando significativamente la atención al paciente. [3]

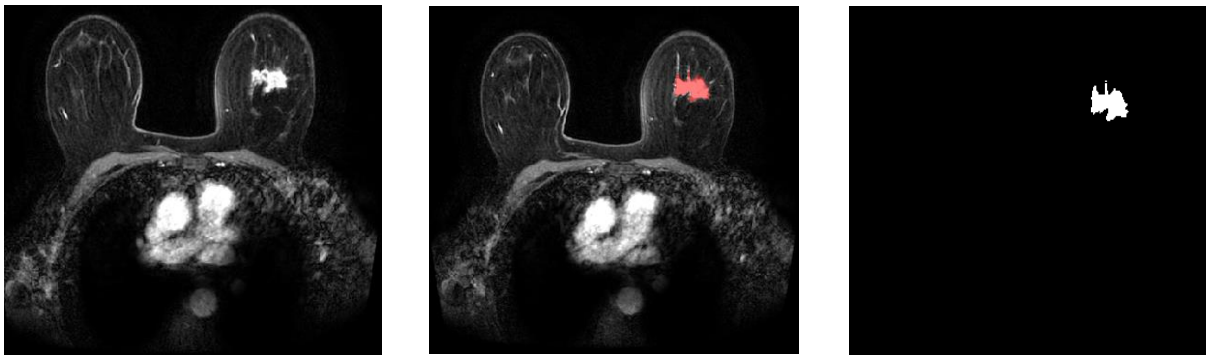


Figura 1. Imagen de lesión de mama con segmentación solapada y máscara

1.4 Objetivos

Con el desarrollo de este modelo en primer lugar conseguiremos agilizar el diagnóstico ya que tiene la capacidad de segmentar más imágenes en menos tiempo, aunque este modelo no sea 100% autónomo, será una gran herramienta para los radiólogos, ya que les permitirá segmentar con más rapidez, lo que llevará a un diagnóstico más precoz.

En el futuro cuando este tipo de modelos sean 100% autónomos y no necesiten supervisión cambiarán el diagnostico drásticamente, ya que estarán las 24 horas trabajando sin descanso, este modelo se unirá a otros para segmentar y clasificar llevando a cabo un diagnóstico casi instantáneo al momento de realizarse las pruebas diagnósticas.

1.5 Organización del documento

Después de una breve introducción para poner en contexto, el documento se encuentra organizado en 5 bloques principales. En el primero se explicarán los componentes teóricos necesarios para entender las redes neuronales, su estructura, funcionamiento, tipos y optimizaciones principales, así como las herramientas y arquitecturas necesarias para lograr nuestro objetivo.

En el siguiente punto, una vez tenemos el contexto sobre el tema de la investigación y una idea de los fundamentos teóricos de las redes neuronales dedicadas a la segmentación de imágenes médicas, se presentan diferentes investigaciones que abordan diferentes técnicas con el fin de conseguir el mismo objetivo.

En el próximo punto se explica la metodología que se lleva a cabo a la hora de realizar el modelo, desde cómo se recopilaron los datos y procesaron hasta la arquitectura y diferentes parámetros de entrenamiento que se usaron a lo largo de toda la investigación.

Para llegar al bloque cinco, donde se describen cómo se eligieron cada uno de los parámetros anteriormente explicados para la creación de cada uno de los modelos, respaldándonos en sus métricas con el fin de ir buscando una evolución. Acabando con una tabla del rendimiento para nuevos casos de cada uno de los modelos, junto a 3 predicciones de cada modelo para poder observar su desempeño en casos nunca antes vistos.

El trabajo acaba con las conclusiones de esta investigación proporcionando el contexto necesario y las limitaciones para entender los resultados obtenidos y las dificultades que conllevan estos casos en la segmentación de lesiones de mama y la gran variedad y complejidad de estos casos

2 Marco Teórico

2.1 Características Anatómicas del Seno y su Impacto en la Interpretación de Imágenes RM

La interpretación de las imágenes de resonancia magnética (RM) del seno se ve significativamente influenciada por la estructura anatómica y fisiológica del seno. Comprender estas características es crucial para mejorar la precisión en la detección y evaluación de lesiones cancerosas. [3]

Estructura Anatómica del Seno

1. **Tejido Adiposo y Glandular**:

El seno está compuesto principalmente por tejido adiposo (graso) y tejido glandular. El tejido adiposo generalmente aparece con una señal de alta intensidad en las imágenes T1-pesadas, mientras que el tejido glandular puede variar en intensidad dependiendo de la secuencia de RM utilizada. Esta diferencia en señal es esencial para distinguir las lesiones, ya que los tumores suelen tener características de señal distintas en comparación con el tejido circundante. [3]

2. **Pezón y Areola**:

El pezón y la areola presentan una morfología y densidad distintas que pueden influir en la interpretación de las imágenes. La areola, al ser más densa y contener glándulas, puede mostrar contrastes que deben ser diferenciados de posibles masas patológicas. [3]

3. **Sistema Linfático**:

Incluye ganglios linfáticos, vasos y tejido linfático. Los ganglios linfáticos pueden aparecer agrandados o con alteraciones en la señal en casos de metástasis. La visualización y evaluación de estos cambios es crucial para determinar la propagación del cáncer. [3]

4. **Vasos Sanguíneos**:

Los vasos sanguíneos proporcionan nutrición y oxígeno a los tejidos del seno. En las imágenes de RM, estos vasos pueden aparecer como estructuras lineales de alta intensidad en ciertas secuencias, y su identificación correcta ayuda a diferenciar entre vasos normales y anomalías vasculares asociadas con tumores. [3]

Impacto en la Interpretación de Imágenes de RM

1. **Diferenciación de Tejidos**:

La habilidad de diferenciar entre tejido adiposo y glandular es fundamental para la identificación de lesiones. Las secuencias de RM deben ser seleccionadas cuidadosamente para maximizar este contraste y facilitar la detección de tumores. [4]

2. **Variabilidad Anatómica**:

Factores como la edad, el ciclo menstrual y la densidad mamaria pueden afectar la apariencia de los tejidos en las imágenes de RM. Por ejemplo, en mujeres premenopáusicas, el tejido glandular puede ser más prominente y variable, lo que requiere ajustes en los parámetros de imagen para asegurar una evaluación precisa. [4]

3. **Uso de Contraste**:

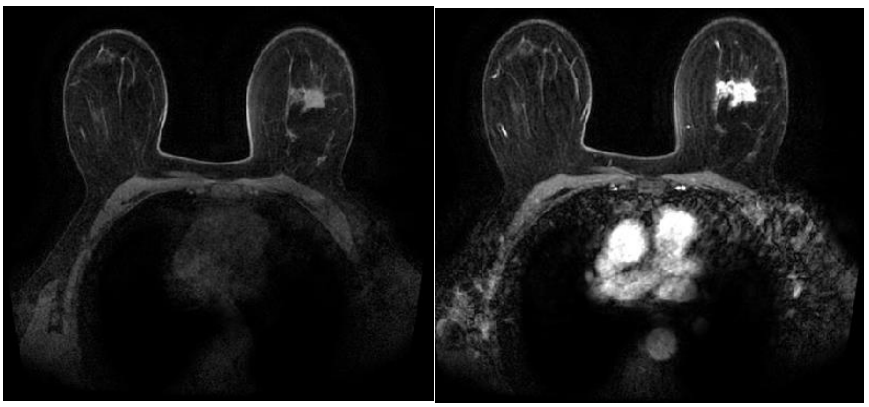


Figura 2. Imagen RM sin contraste y con contraste

Como se puede observar en la figura 2, la administración de agentes de contraste como el gadolinio es esencial para resaltar áreas con alta vascularización y permeabilidad, características comunes en tumores malignos. El contraste mejora la visibilidad de las lesiones en relación con el tejido mamario normal, permitiendo una mejor evaluación de la extensión y características de la lesión. [4]

4. **Identificación de Lesiones y Anomalías**:

Las lesiones y masas en el seno, tanto benignas como malignas, presentan características específicas en las imágenes de RM. Los tumores malignos, por ejemplo, pueden mostrar bordes irregulares y un realce heterogéneo tras la administración de contraste. La identificación precisa de estas características es crucial para el diagnóstico y la planificación del tratamiento.[4]

2.2 Redes Neuronales

En este apartado se incluyen los conocimientos necesarios para entender el funcionamiento de las redes neuronales y su uso para problemas de segmentación.

2.2.1 *¿Qué son las redes neuronales?*

Para profundizar en la explicación de una red neuronal como un sistema de procesamiento de información, vamos a repasar la funcionalidad de cada componente y su utilidad dentro del sistema, así como su aplicación en problemas específicos:

Como se observa en la figura 3, una red neuronal artificial se inspira en la red neuronal biológica del cerebro humano, diseñada para simular su capacidad de aprendizaje y reconocimiento de patrones. Este sistema computacional está compuesto por unidades de procesamiento llamadas nodos o neuronas artificiales, las cuales están organizadas en capas y conectadas entre sí por enlaces que representan sinapsis en un cerebro biológico. [5]

Después de la entrada de los datos a la red neuronal, se procesan y se genera una salida, que permite resolver problemas de regresión y/o clasificación, es decir, predice un valor numérico (regresión) o una probabilidad de pertenencia a una clase (clasificación). [6]

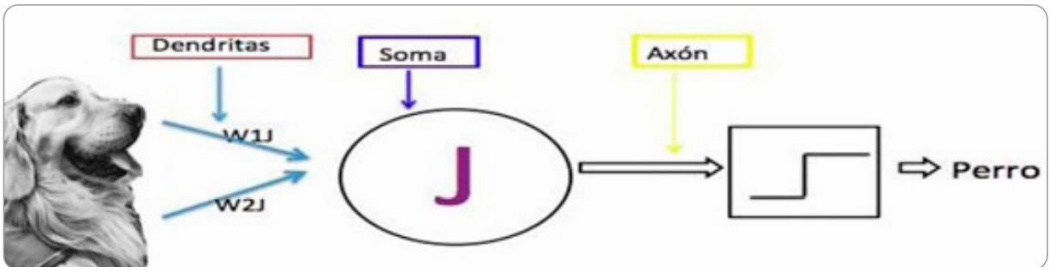


Figura 3. Esquema neuronal artificial inspirada en neurona biológica [5]

2.2.2 *Componentes y funcionamiento*

Una red neuronal se basa en tres pilares fundamentales:

Arquitectura: Es el diseño o esquema que sigue la red, compuesto por varias capas de neuronas organizadas de cierta manera. Esta configuración incluye capas de inicio, donde se ingresan los datos, que se suelen identificar como características o factores de entrada **(Xn)**, capas escondidas, que analizan la información internamente, y capas finales, donde se obtienen los resultados **(a)**. La estructura es clave para determinar cómo se manejan y se transforman los datos a lo largo de la red. [7]

Pesos y conexiones **(Wn)**: Las neuronas están interconectadas y cada enlace tiene un valor específico o peso, que refleja cuánto influencia una neurona sobre otra en el proceso de decisión. Durante el entrenamiento de la red, estos pesos se van modificando para que la red se vuelva experta en la tarea que tiene que realizar, aprendiendo de los ejemplos que recibe. [7]

Funciones de Activación **(σ)**: Cada neurona procesa las señales que recibe utilizando una regla o función específica de activación, que calcula la salida de la neurona en función de sus entradas. Estas reglas son cruciales porque deciden cómo y cuándo una neurona debe activarse, contribuyendo así al procesamiento global de la información dentro de la red. [7]

En la siguiente figura se observa esquemáticamente como se organiza una red identificando cada componente explicado.

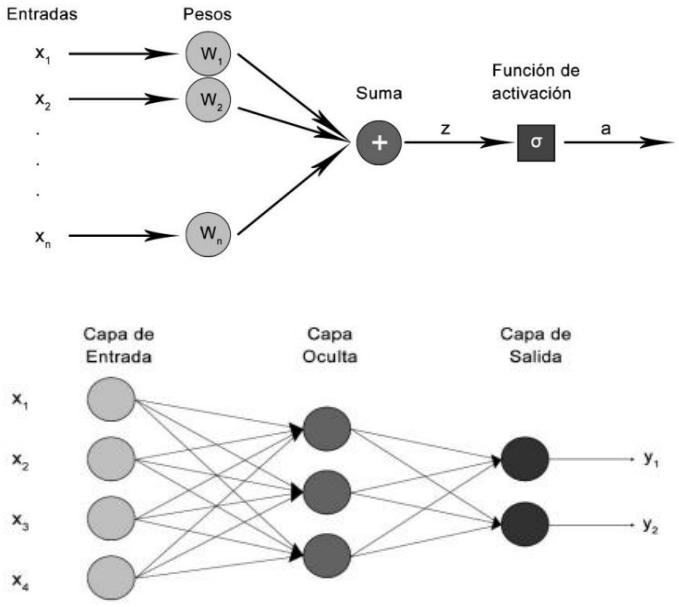


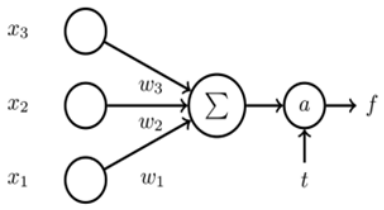
Figura 4. Arquitectura multicapa con dos salidas.[8]

Funcionamiento de una red neuronal

El aprendizaje en una red neuronal se refiere a la actualización de los pesos de las conexiones entre las neuronas, basándose en la diferencia entre la salida esperada y la salida real de la red. La Regla Delta es un algoritmo de aprendizaje temprano que ajusta estos pesos para minimizar el error en la salida de la red. Este método iterativo ajusta los pesos de manera que la red neuronal artificial aprende de manera gradual, mejorando su precisión en la tarea asignada *(Cuevas-Tello, 2018)* [8]

Este aprendizaje se asemeja al proceso biológico de fortalecimiento de conexiones entre las neuronas en el cerebro. A medida que practicamos una habilidad, las neuronas involucradas refuerzan sus conexiones, lo que mejora nuestra capacidad para realizar esa habilidad. En las redes neuronales artificiales, este proceso se emula ajustando los pesos de las conexiones entre los nodos, basándose en los datos de entrada y los errores en las predicciones o clasificaciones.

Por ejemplo, se presenta un escenario en el que se decide si ir de viaje o no, basándose en tres factores: si se tiene suficiente dinero, si la pareja quiere ir y si el clima es agradable. Cada factor tiene un peso que indica su importancia en la decisión. Este proceso ilustra cómo una red neuronal puede tomar decisiones complejas al ponderar diferentes factores de entrada.[9]



x_1	x_2	x_3	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Figura 5. Red neuronal con tabla de decisiones. *(Cuevas-Tello, 2018)* [8]

Esta imagen podría representar el caso de ejemplo, donde ***X1, X2*** y ***X3*** serían los factores de decisión y ¨***f***¨ la decisión de si ir de viaje o no. Cada factor tendría un peso que se iría modificando en base a los datos aportados anteriormente hasta llegar al óptimo, es ahí cuando la red está entrenada para recibir nuevos datos y ser capaz de tomar la decisión.

2.2.3 *Aprendizaje supervisado y no supervisado*

Aprendizaje supervisado:

El aprendizaje supervisado en redes neuronales se fundamenta en el uso de datos etiquetados para enseñar a la red cómo predecir correctamente la salida para nuevas entradas. Este proceso se desarrolla en varios pasos esenciales:

- **Inicialización de los pesos**: Los pesos de las conexiones neuronales se asignan con valores iniciales, que pueden ser aleatorios o seguir algún criterio específico.
- **Propagación hacia adelante**: La red procesa las entradas, pasándolas a través de sus capas de neuronas. Cada neurona suma las entradas ponderadas por sus respectivos pesos y aplica una función de activación para producir una salida.
- **Cálculo del error**: La diferencia entre las salidas predichas por la red y las salidas reales esperadas (etiquetas) se calcula para determinar cuánto debe ajustarse la red.
- **Back propagation**: Este error se utiliza para ajustar los pesos de la red, propagando el error de salida hacia atrás desde la capa de salida hacia las capas de entrada, modificando los pesos para reducir el error en futuras predicciones.
- **Actualización de los pesos**: Mediante algoritmos de optimización, como el descenso de gradiente, los pesos se ajustan en función del error calculado.
- **Repetición del proceso**: Los pasos del 2 al 5 se repiten a través de múltiples iteraciones o épocas, utilizando todos los ejemplos del conjunto de entrenamiento, para afinar gradualmente los pesos y mejorar la capacidad predictiva de la red.

Este enfoque busca que, tras suficiente entrenamiento, la red neural sea capaz de generalizar desde los ejemplos de entrenamiento a nuevas situaciones, produciendo salidas correctas para entradas que no se han visto previamente, aplicable en una diversidad de campos como reconocimiento de patrones, procesamiento del lenguaje y análisis predictivo [10]

Aprendizaje no supervisado:

El aprendizaje no supervisado en redes neuronales se centra en analizar y aprender de datos sin etiquetar, sin salidas específicas proporcionadas para guiar el entrenamiento. La red busca identificar patrones, estructuras ocultas o agrupaciones dentro de los datos. Este enfoque difiere del aprendizaje supervisado, ya que no se basa en comparar las salidas predichas con las correctas, sino en descubrir relaciones intrínsecas en el conjunto de datos. Los principales métodos de aprendizaje no supervisado incluyen:

- **Clustering**: Agrupación de datos en clústeres basados en similitudes para identificar patrones y segmentar los datos en grupos significativos.
- **Reducción de dimensionalidad**: Simplificación de los datos a representaciones de menor dimensión, manteniendo la información esencial, lo que ayuda a visualizar y entender mejor los datos.
- **Generación de datos**: Creación de nuevos datos que imitan los originales, usando modelos como las Redes Generativas Adversariales (GAN) para aprender y replicar la distribución de los datos existentes.

Estas técnicas se aplican en variados campos, desde la detección de anomalías hasta la recomendación de productos y la exploración de grandes conjuntos de datos, permitiendo revelar información no evidente y descubrir estructuras subyacentes sin requerir datos etiquetados previamente. El aprendizaje no supervisado es fundamentalmente exploratorio, enfrentando el desafío de identificar patrones significativos sin una guía explícita, demandando una interpretación cuidadosa y un conocimiento profundo del dominio de los datos. [10]

2.2.4 *Redes Convolucionales (CNN)*

Estas redes son las que aplicaremos en nuestro modelo para la detección de bordes y clasificación de vóxeles, consiguiendo así la segmentación de lesiones de cáncer de mama.

Las redes neuronales convolucionales (CNN, por sus siglas en inglés "Convolutional Neural Networks"), son una categoría especializada de redes neuronales artificiales inspiradas en la organización de las neuronas de la corteza visual humana. Estas "neuronas" en las CNN replican la estructura de los campos receptivos, similar a cómo funcionan las neuronas en la región visual primaria del cerebro. A diferencia de los perceptrones multicapa tradicionales, las CNN se especializan en el procesamiento de datos en forma de matrices bidimensionales, lo que las hace particularmente aptas para aplicaciones en el campo de la visión por computadora, tales como la clasificación de imágenes y la segmentación, entre otras funcionalidades [11]

Componentes de una CNN:

Una Red Neuronal Convolucional (CNN) se estructura en tres capas esenciales:

- Capa Convolucional: Funciona como el fundamento de una CNN, ejecutando los cálculos primarios. Esta capa emplea varios componentes como la entrada (por ejemplo, una imagen RGB tridimensional) y un filtro (o kernel) que recorre la imagen identificando características específicas mediante la convolución, y genera un mapa de características. A este proceso se suma la aplicación de la función ReLU para añadir no linealidad.

- Capa de Agrupación (Pooling): Se encarga de reducir la dimensión de los datos simplificando la información obtenida, mediante técnicas como la agrupación máxima o media, preservando solo los valores más significativos de los campos receptivos examinados.

- Capa Totalmente Conectada: En esta última etapa, todos los nodos de la capa están interconectados con la capa anterior, facilitando la clasificación final de la entrada en base a las características detectadas, empleando funciones de activación como la softmax para la probabilidad de clasificación.

Cada capa aporta un sistema de detección de características de la imagen , donde las primeras identifican aspectos simples de la entrada, como bordes y colores, y conforme avanzan, reconocen elementos cada vez más complejos, hasta llegar a identificar el objeto completo dentro de la imagen. Esta estructura permite que la CNN aprenda de manera eficiente desde características básicas hasta patrones complejos necesarios para la clasificación y reconocimiento de imágenes [12].

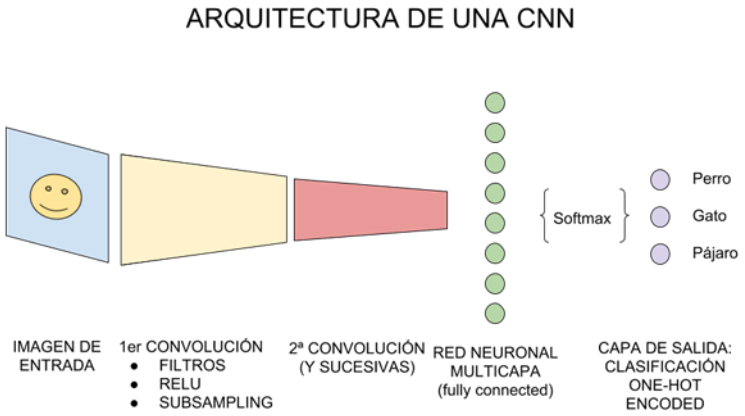


Figura 6. Estructura de una Red Neuronal Convolucional [13]

Después de ver la estructura de una CCN veamos los pasos fundamentales, desde el momento en que se introduce una imagen en la red hasta el proceso de clasificación final:

Paso 1. Píxeles como Neuronas

La entrada a la CNN son los píxeles de una imagen. Por ejemplo, una imagen de 28×28 píxeles en escala de grises corresponde a 784 neuronas en la capa de entrada. Para imágenes a color, que utilizan 3 canales (RGB), el número de neuronas de entrada sería de 2352 (28x28x3). [13]

Figura 7. Imagen que contiene una A con valores de los pixeles de 0 a 255. [13]

Paso 2. Preprocesamiento de la Imagen

Antes de procesar la imagen, se normalizan los valores de los píxeles, en este caso dividiéndolos por 255, resultando en valores entre 0 y 1. Esto asegura que la red trabaje con un rango de datos estandarizado. [13]

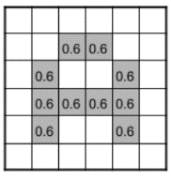


Figura 8. Imagen normalizada con valor de los pixeles de 0 a 1. [13]

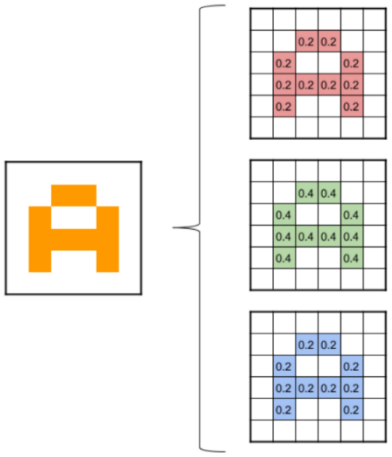


Figura 9. Imagen de 3 canales RGB normalizada (28x28x3). [13]

Paso 3. La Convolución

La operación clave de las CNN es la convolución, donde se utilizan filtros (o kernels) para detectar características específicas en la imagen, como bordes o texturas. Un kernel típico podría tener un tamaño de 3×3. A medida que este filtro se desplaza sobre la imagen, se calcula el producto escalar entre el filtro y la sección de la imagen bajo el filtro, generando un mapa de características. Este filtro tomará valores aleatorios e irá ajustándose mediante back propagation. [13]

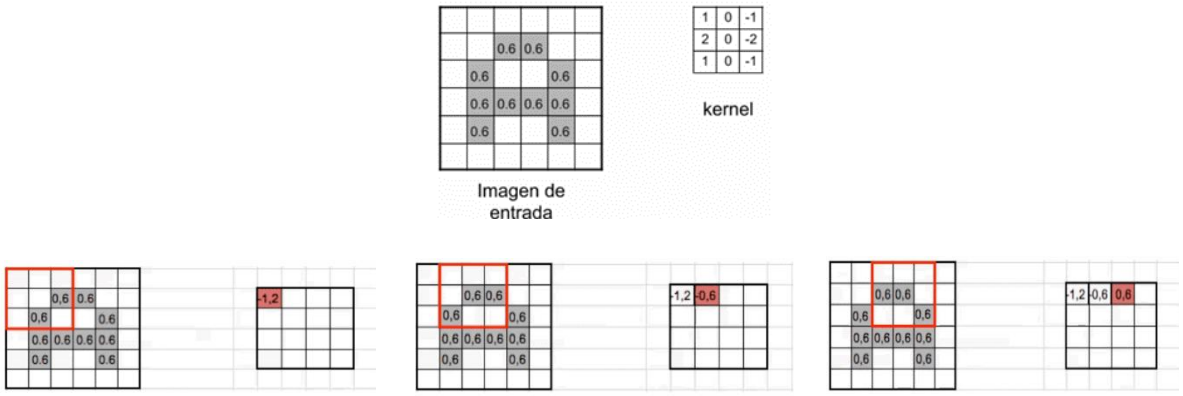


Figura 10. Muestra el resultado de los tres primeros desplazamientos del kernel durante la operación de convolución [13]

Paso 4. Múltiples Filtros para la Detección de Características

No solo se utiliza un único filtro, sino un conjunto de ellos, lo que resulta en múltiples mapas de características después de la primera convolución. Estos mapas sirven para resaltar diferentes características de la imagen.

Conforme movemos el kernel a través de la imagen y aplicamos el proceso de filtrado, generamos lo que se podría considerar como 32 "nuevas imágenes" procesadas por el kernel, (utilizando 32 filtros). Estas imágenes resultantes destacan aspectos específicos de la imagen original. Esta capacidad de resaltar ciertas características facilitará la identificación y diferenciación de un objeto respecto a otro en etapas posteriores del análisis. [13]

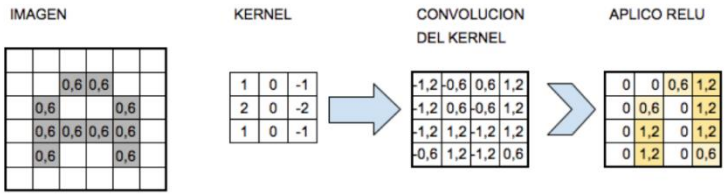


Figura 11. Obtención de mapa de características [13]

Paso 5. Función de Activación: ReLU

Después de cada operación de convolución, se aplica la función de activación, en este caso ReLU (Rectified Linear Unit) para añadir no linealidad al modelo, permitiendo que la red capture complejidades en los datos.

$$f(x)=Max\ (0,\ x).$$

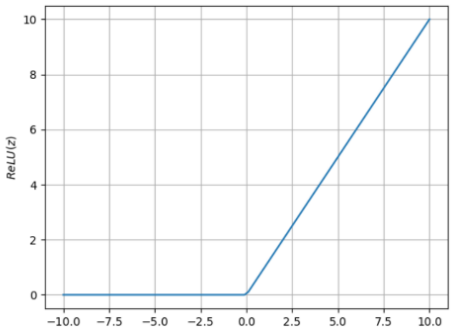


Figura 12. Gráfico función de activación ReLu [14]

Paso 6. Submuestreo o Pooling

Este paso reduce el tamaño de los mapas de características, simplificando la información mediante técnicas como Max-Pooling. De esta manera se reduce la cantidad de parámetros y el cálculo necesario para la red, manteniendo las características más importantes.[13]

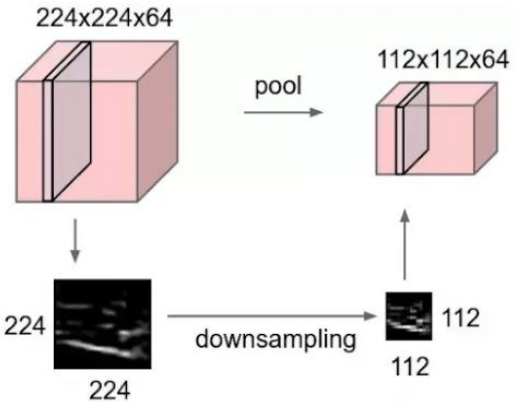


Figura 13. Ejemplo de Max-Pooling. [15]

Paso 7. Convoluciones y Pooling Adicionales

La red puede incluir varias capas de convolución y pooling sucesivas, cada una detectando características cada vez más complejas y abstractas a medida que se avanza en la red. [13]

Paso 8. Capa Totalmente Conectada

Después de las capas convolucionales y de pooling, la información procesada se aplana y se pasa a una o varias capas totalmente conectadas, que integran las características aprendidas para realizar la clasificación final. [13]

Paso 9. Clasificación Final: función de perdida Softmax

La última capa de la CNN es típicamente una capa de Softmax, que convierte las salidas de la red en probabilidades para cada clase objetivo, permitiendo la clasificación de la imagen de entrada en categorías como "gato" o "perro". [13]

Paso 10. Aprendizaje: Back propagation

A lo largo del entrenamiento, la CNN ajusta los pesos de los filtros y las conexiones en las capas totalmente conectadas mediante retro propagación, basándose en la diferencia entre las salidas predichas y las reales, mejorando su capacidad para reconocer y clasificar imágenes correctamente [16]

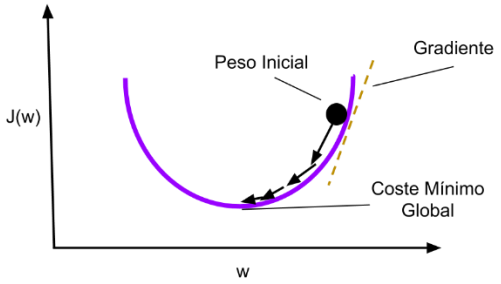


Figura 14. Proceso de optimización de pesos hasta llegar al punto mínimo. [17]

En la figura 14 se muestra el proceso de optimización de pesos que se va modificando en cada época en busca del mínimo global obteniendo así el valor ideal que debe tener el peso.

El conjunto de todos estos puntos forma un proceso iterativo que, a partir de un amplio conjunto de datos, permite a la red aprender a identificar y generalizar características principales de cada objeto, consiguiendo así una gran precisión en tareas de clasificación de imágenes como el reconocimiento de un objeto en una imagen. En nuestro caso será el reconocimiento de una lesión de cáncer de mama a través de imágenes de resonancia magnética.

2.2.5 *Modelo U-Net*

Existen diferentes arquitecturas ampliamente extendidas entre las cuales por su uso mayoritario para esta tarea se va a presentar la U-Net, este modelo es el más utilizado para la segmentación de imágenes médicas.

U-Net es una arquitectura de red neuronal convolucional desarrollada específicamente para la segmentación de imágenes biomédicas. Diseñada por Olaf Ronneberger, Philipp Fischer y Thomas Brox de la Universidad de Freiburg, la U-Net destaca por su eficacia incluso con un número limitado de imágenes anotadas para entrenamiento.[18]

La estructura de la U-Net se compone de dos partes principales: un camino de contracción y un camino de expansión, formando una estructura en forma de "U". El camino de

contracción sigue la arquitectura típica de una red convolucional, donde se aplican sucesivamente convoluciones y operaciones de max pooling para capturar el contexto de la imagen mientras se reduce su dimensión espacial. Este camino incrementa progresivamente el número de canales de características a medida que avanza en la profundidad de la red. [18]

El camino de expansión, por otro lado, realiza operaciones de upsampling ,estas operaciones sirven para aumentar la resolución de la salida. En cada paso de este camino, se combina la información espacial y de características mediante una serie de convoluciones y la concatenación con mapas de características de alta resolución del camino de contracción. Esta simetría entre los caminos de contracción y expansión permite que la red propague información de contexto a capas de mayor resolución y, en consecuencia, logre una localización precisa.[19]

Una innovación clave en la U-Net es el uso intensivo de la aumentación de datos, en particular mediante deformaciones elásticas, lo que permite que la red aprenda invarianzas a tales transformaciones sin necesidad de verlas explícitamente en el conjunto de datos de entrenamiento. Esto es crucial en la segmentación biomédica, donde las variaciones de forma son comunes. Además, la U-Net maneja el problema de separar objetos que se tocan mediante el uso de un mapa de pesos en la función de pérdida, lo que enfoca la atención de la red en los bordes de separación entre objetos .[19]

En términos de rendimiento, la U-Net ha demostrado ser muy eficaz en una variedad de aplicaciones de segmentación biomédica, superando a otros métodos en competiciones como el desafío de segmentación de estructuras neuronales en imágenes de microscopía electrónica. Su rapidez también es notable, siendo capaz de segmentar una imagen de 512x512 en menos de un segundo con una GPU moderna. [19]

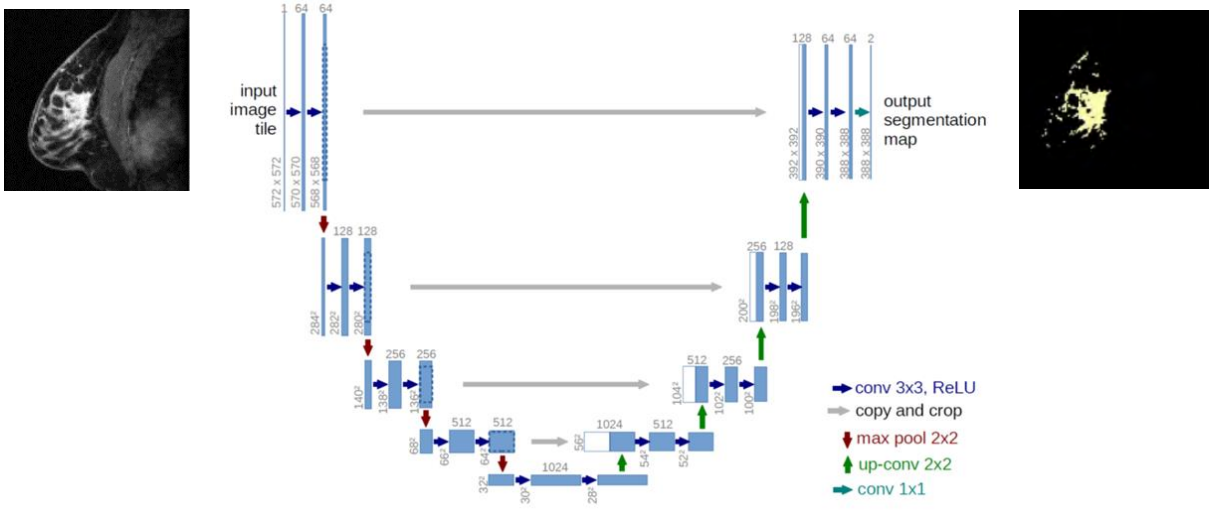


Figura 15. Arquitectura U-net. [20]

2.2.6 *Entrenamiento de Redes Neuronales*

Entrenar redes neuronales es un proceso complejo que implica varios pasos para asegurar que el modelo aprenda eficientemente a partir de los datos disponibles y pueda realizar predicciones o clasificaciones precisas. A continuación, se explicará globalmente este proceso en las secciones mencionadas, y más adelante se profundizará en características que se apliquen a nuestro modelo.

2.2.6.1 Conjunto de Datos

El conjunto de datos es el primero y uno de los elementos más importantes en el entrenamiento de redes neuronales. Consiste en los ejemplos de entrada (características) y salida (etiquetas) que la red utilizará para aprender. Los datos deben ser relevantes, suficientemente variados y, en el caso de la supervisión, correctamente etiquetados.

1. Recopilación de Datos: Este paso implica reunir un conjunto de datos amplio y representativo del problema a resolver.

2. Limpieza y Preprocesamiento: Los datos crudos a menudo requieren limpieza y transformaciones, como normalización, escalado, y manejo de valores faltantes.

3. Aumento de Datos: Para mejorar la robustez del modelo y evitar el sobreajuste, se pueden aplicar técnicas de aumento de datos, que generan datos de entrenamiento adicionales mediante la modificación de los datos existentes de formas que son invariantes a las predicciones del modelo.

4. División de Datos: Generalmente, el conjunto de datos se divide en tres partes: entrenamiento, validación y prueba, para entrenar el modelo, ajustar hiperparámetros y evaluar el rendimiento del modelo, respectivamente.

2.2.6.2 Técnicas de optimización, funciones de pérdida y activación

El objetivo del entrenamiento de una red neuronal es ajustar los pesos de la red para minimizar una función de pérdida, que mide la diferencia entre las predicciones del modelo y los valores reales.

- **Funciones de Pérdida**

Son criterios matemáticos para evaluar el rendimiento del modelo. Por ejemplo, la entropía cruzada se usa comúnmente para clasificación, mientras que el error cuadrático medio es frecuente en problemas de regresión. [6]

En el contexto de la segmentación de imágenes con redes neuronales, la elección de la función de pérdida es crucial para el éxito del modelo. [6] En este documento, se describirán cuatro funciones de pérdida ampliamente utilizadas en la segmentación de imágenes: Cross-Entropy Loss**,** Jaccard Index (IoU) Loss**,** Dice Loss y Tversky Loss**.** Se explicará el funcionamiento de cada una de estas funciones.

Cross- Entropy Loss: La Cross-Entropy Loss es una de las funciones de pérdida más comunes utilizadas en tareas de clasificación, incluidas aquellas que implican la segmentación de imágenes. Esta pérdida mide la diferencia entre la distribución de probabilidad predicha y la distribución de probabilidad real de las clases.[24]

La Jaccard Index (IoU) Loss: es una función de pérdida basada en el Índice de Jaccard o Intersection over Union (IoU), que mide la superposición entre las predicciones y las etiquetas reales. Esta métrica es particularmente útil para problemas de segmentación donde la precisión de la superposición es importante. [25]

Dice Loss: la función de perdida Dice es ampliamente utilizada en la segmentación de imágenes debido a su capacidad para manejar datos desbalanceados y medir la similitud entre conjuntos de pixeles predichos y reales. [26]

Tversky Loss: es una generalización de las funciones de pérdida de Dice y Jaccard, ajustando los pesos de penalización para falsos positivos y falsos negativos. Esta función es particularmente útil en problemas de segmentación con clases desbalanceadas. [26]

- **Optimizadores**

Son algoritmos para ajustar los parámetros del modelo en función del gradiente de la función de pérdida. Algunos ejemplos incluyen **SGD (Gradiente Descendente Estocástico), Adam y RMSprop**.[6]

El Gradiente Descendente Estocástico (SGD) es una variante del algoritmo de gradiente descendente que actualiza los parámetros del modelo usando solo una muestra (o un pequeño lote de muestras a la vez) en lugar de utilizar todo el conjunto de datos. Esto hace que el SGD sea más rápido y eficiente, especialmente con grandes conjuntos de datos.[27]

Adam (Adaptive Moment Estimation) combina las ventajas de dos otros métodos de optimización: AdaGrad y RMSProp. Calcula estimaciones adaptativas de momentos de primer y segundo orden.[28]

RMSprop es un método de optimización que adapta la tasa de aprendizaje para cada parámetro dividiendo la tasa de aprendizaje por una media móvil del cuadrado de los gradientes recientes.[29]

Optimizadores	**Ventajas**	**Desventajas**
SGD	• Computacionalmente eficiente. • Converge más rápido para grandes data sets.	• Puede ser ruidoso y no converger directamente al mínimo global.
Adam	• Rápida convergencia. • Ajuste adaptativo de la tasa de aprendizaje.	• Puede no generalizar tan bien en algunos casos específicos.
RMSprop	• Maneja bien la tasa de aprendizaje variable. • Converge más rápido en muchos problemas.	• Puede ser sensible a la elección del parámetro de decaimiento.

Tabla 1. Tabla comparativa de optimizadores [29]

- **Regularización y Técnicas de Prevención del Sobreajuste**

Una vez visualizadas y evaluadas las métricas del modelo, muchas veces podemos observar el sobreajuste, que ocurre cuando un modelo se adapta demasiado bien a los datos de entrenamiento, capturando incluso el ruido, y como resultado, su rendimiento en datos nuevos o no vistos se degrada. Para evitar el sobreajuste, asegurando que el modelo pueda generalizar para los datos no vistos, se utilizan varias técnicas de regularización. [22] [6]

Dropout: es una técnica de regularización que reduce el sobreajuste en redes neuronales al desactivar aleatoriamente un porcentaje de las neuronas durante cada paso de entrenamiento. Esto evita que las neuronas se vuelvan demasiado dependientes unas de otras. [22]

Durante cada iteración de entrenamiento, se "apagan" aleatoriamente ciertas neuronas con una probabilidad p. Esto fuerza a la red a aprender representaciones redundantes y distribuidas.

Se implementa aplicando una capa dropout al final de cada capa o entre capas de la red.

La normalización por lotes (Batch Normalization): es una técnica que mejora la estabilidad y la velocidad del entrenamiento de redes neuronales al normalizar las salidas de una capa en mini-batches.[22]

Antes de aplicar la activación, las salidas de cada mini-batch se normalizan para tener media cero y varianza uno. Luego, se aplican parámetros de escalado y desplazamiento aprendibles. [22]

La aplicación de la normalización por lotes puede llevar a los siguientes beneficios:

Acelera el Entrenamiento**:** Al permitir usar tasas de aprendizaje más altas.

Estabiliza el Aprendizaje**:** Reduciendo la sensibilidad a la inicialización de parámetros**.**

Regularización Implícita**:** Reduce la necesidad de otras formas de regularización, como dropout.

Regularización L1 y L2: Estas técnicas añaden un término de penalización al valor de la función de pérdida, incentivando que los pesos del modelo sean más pequeños.

L1 (Lasso): La penalización L1 es la suma de los valores absolutos de los pesos. Promueve la disparidad, llevando muchos pesos a cero. [21]

L2 (Ridge): La penalización L2 es la suma de los cuadrados de los pesos. Promueve pesos pequeños y distribuidos uniformemente.[21]

Early stopping: es una técnica que monitorea el rendimiento del modelo en un conjunto de validación durante el entrenamiento y detiene el entrenamiento cuando el rendimiento deja de mejorar.

Se divide el conjunto de datos en un conjunto de entrenamiento y un conjunto de validación. Durante el entrenamiento, si el error en el conjunto de validación empieza a aumentar (indicación de sobreajuste), el entrenamiento se detiene. [21]

Aumento de Datos (Data Augmentation): el aumento de datos genera nuevas muestras de datos a partir de los datos existentes mediante transformaciones como rotaciones, desplazamientos, cambios de escala, etc.

El beneficio de utilizar esta técnica se basa en incrementar la cantidad y diversidad de los datos de entrenamiento, mejorando la capacidad del modelo para generalizar.

Implementar estas técnicas permite no solo desarrollar un modelo robusto sino también mejorar su capacidad para generalizar a nuevos datos, crucial para la segmentación precisa y fiable de lesiones de cáncer de mama en imágenes de resonancia magnética.[22]

- **Funciones de Activación**

Como se ha visto anteriormente, las funciones de activación juegan un papel importante en el cálculo del gradiente. Es común encontrar en una arquitectura multicapa dos tipos de funciones de activación, el que utiliza las capas internas para aprender y el que utiliza la capa de salida para producir el resultado del problema tratado [6]. Las funciones de activación principales son ReLu, Sigmoid, Softmax y Tanh.

La función sigmoide: es una de las funciones de activación más antiguas y ampliamente utilizadas. Convierte el valor de entrada en un rango entre 0 y 1, lo que facilita el tratamiento de problemas de clasificación binaria.[30]

La función ReLU (Rectified Linear Unit): es la más utilizada en redes neuronales profundas debido a su simplicidad y eficacia. Activa la neurona solo si la entrada es positiva, de lo contrario, la salida es cero.[31]

La función Tanh (Tangente Hiperbólica): esta función es similar a la función sigmoide, pero sus valores de salida están centrados en cero, lo que puede ayudar a que la red converja más rápido. .[31]

La función Softmax: se utiliza principalmente en la capa de salida de una red neuronal para tareas de clasificación multiclase. Convierte un vector de valores en una distribución de probabilidad. .[31]

Funciones de Activación	Ventajas	Desventajas
SGD	• Suaviza la salida, lo que puede ser útil para la probabilidad de clasificación.	• Problema de saturación: para valores extremos de entrada, el gradiente se vuelve muy pequeño, • No es centrada en cero, lo que puede causar problemas de convergencia en redes profundas.
ReLU	• Soluciona el problema del desvanecimiento del gradiente. • Proporciona eficiencia computacional.	• Problema de "muerte" de las neuronas: las neuronas pueden quedar atascadas en 0 durante el entrenamiento y dejar de aprender
Tanh	• Centrada en cero, lo que puede mejorar la convergencia en redes profundas. • Amplifica los gradientes para entradas pequeñas, lo que puede acelerar el aprendizaje.	• Al igual que la función sigmoide, puede sufrir de desvanecimiento del gradiente.
Softmax	• Proporciona una probabilidad de salida para cada clase. • Es útil para problemas de clasificación multiclase.	• Es computacionalmente más costosa debido a la función exponencial.

Tabla 2. Comparativa de funciones de activación

2.2.6.3 Evaluación del Modelo

Una vez entrenado el modelo, es crucial evaluar su rendimiento con un conjunto de datos no visto anteriormente para asegurar que ha aprendido a generalizar y no las características específicas del conjunto de entrenamiento.

1. Métricas de Evaluación:
   Dependiendo del tipo de tarea (clasificación, regresión, segmentación, etc.), se utilizan diferentes métricas como precisión, recall, F1-score, error cuadrático medio, entre otros, para el caso de segmentaciones es crucial mencionar la métrica de DICE el Coeficiente de correlación de Matthews (MCC) que evalúa el nivel de coincidencia entre la máscara original y la predicha, siendo 0 nada de coincidencia y 1 coincidencia exacta. [21]

2. Validación Cruzada:
   Es una técnica utilizada para identificar si nuestro modelo elegido funciona bien dándole un subconjunto de datos u otro y comprobando si los resultados siguen siendo estables. Además, nos ayuda a detectar si algún subconjunto presenta datos diferentes.

3. Ajuste de Hiperparámetros:
   Es el proceso por el cual se busca optimizar los hiperparámetros del modelo, como la tasa de aprendizaje, el número de capas/neuronas, etc., para mejorar el rendimiento, generalmente se realiza de forma manual e iterativa.

Cada uno de estos pasos es crucial para asegurar que la red neuronal no solo aprenda a predecir los datos de entrenamiento, sino que también pueda generalizar bien a nuevos datos, un factor fundamental para su aplicación en el mundo real.

3 Estado del Arte

En el ámbito de la segmentación automática de lesiones de cáncer de mama, se han realizado varios estudios que han avanzado significativamente en la precisión y eficiencia de estas técnicas. La segmentación de imágenes médicas, especialmente en el caso del cáncer de mama, es crucial para el diagnóstico y tratamiento adecuado. La evolución de las redes neuronales profundas ha permitido desarrollar modelos más sofisticados y precisos para este propósito. A continuación, se comparan los resultados obtenidos en el presente estudio con otros trabajos relevantes en el área

- **A U-Net Ensemble for Breast Lesion Segmentation in DCE MRI**

  El artículo titulado "A U-Net Ensemble for Breast Lesion Segmentation in DCE MRI" de Roa'a Khaled, Joel Vidal, Joan C. Vilanova y Robert Martí, publicado en 2022, propone un método automatizado para la segmentación de lesiones mamarias en imágenes de resonancia magnética dinámica con realce de contraste (DCE-MRI) basado en una arquitectura U-Net modificada. Este estudio fue realizado en la Universidad de Girona. El enfoque principal del estudio fue implementación de un conjunto (ensemble) de tres modelos U-Net, cada uno utilizando diferentes combinaciones de entrada, lo que permite mejorar significativamente la precisión de la segmentación. El tamaño de la muestra utilizada fue de 46 casos del conjunto de datos TCGA-BRCA. Los resultados obtenidos demuestran un coeficiente de similitud de Dice (DSC) medio de 0.680 y 0.802 para las lesiones principales, superando a los métodos existentes utilizando el mismo conjunto de datos. La arquitectura propuesta incluye bloques residuales en la U-Net modificada, lo que contribuye a una mejor capacidad de generalización y precisión en la segmentación. Este estudio muestra cómo la combinación de múltiples modelos U-Net puede mejorar la precisión de la segmentación de lesiones de cáncer de mama en imágenes DCE-MRI, destacando la importancia de los enfoques de conjunto en la mejora de la eficacia de los modelos de aprendizaje profundo.[32]

- **Automated Breast Tumor Ultrasound Image Segmentation with Hybrid UNet and Classification Using Fine-tuned CNN Model**

  El estudio titulado "Automated breast tumor ultrasound image segmentation with hybrid UNet and classification using fine-tuned CNN model" de Shahed Hossain et al., publicado en 2023, aborda la segmentación automática de imágenes de ultrasonido de tumores mamarios y la clasificación de estos en tumores benignos y malignos utilizando una arquitectura U-Net híbrida optimizada y un modelo CNN fino ajustado. Este estudio, realizado por investigadores de la Health Informatics Research Laboratory y la Facultad de Ciencia y Tecnología de la Charles Darwin University,

utiliza diversas técnicas de procesamiento de imágenes para mejorar la calidad de las mismas eliminando texto, artefactos y ruido de moteado. [33]

La investigación utiliza el Breast Ultrasound Image (BUI) dataset de la plataforma open-source Kaggle, que incluye 780 imágenes de ultrasonido, categorizadas en tres clases: 133 imágenes normales, 210 imágenes de tumores malignos y 487 imágenes de tumores benignos. Después de eliminar las imágenes normales, el dataset se enfoca en dos clases: tumores benignos y malignos. Además, se utiliza un dataset adicional de mamografías con 1459 imágenes para verificar la robustez del modelo propuesto.[33]

- **Comparación con el Estudio Actual**

En el presente estudio, se utilizó una arquitectura U-Net para la segmentación automática de lesiones de cáncer de mama en imágenes de RM, con una muestra total de 21 casos. Los resultados obtenidos muestran un coeficiente de similitud de Dice (DSC) de 0.308. La menor cantidad de casos en la muestra, en comparación con los estudios antes mencionados, que contenían más datos de entrenamiento (46 casos en el estudio de Roa'a Khaled et al. y 780 imágenes en el estudio de Shahed Hossain et al.), y la limitada capacidad computacional disponible pueden justificar los resultados más bajos. Además, el segundo estudio también trabajó con imágenes de ultrasonido, donde los patrones son más simples, con pocas variaciones y mucho más fáciles de identificar, lo que explica sus elevados resultados. Estas limitaciones subrayan la importancia de contar con recursos computacionales adecuados y un conjunto de muestras suficientemente grande para mejorar la precisión de los modelos de segmentación.

4 Metodología

4.1 Tecnologías y Herramientas

En el desarrollo de este Trabajo de Fin de Grado (TFG), se han empleado diversas herramientas y tecnologías que facilitaron tanto el procesamiento de datos como la visualización y evaluación de las imágenes de resonancia magnética (RM) y sus segmentaciones. A continuación, se detallan las principales herramientas y tecnologías utilizadas, para poner en contexto y conocer estas herramientas antes de explicar la metodología.

1. Jupyter Notebook con Anaconda

Jupyter Notebook es una aplicación web interactiva que permite la creación y el intercambio de documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. En este proyecto, Jupyter Notebook ha proporcionado un entorno flexible para desarrollar, ejecutar y documentar el código de preprocesamiento, normalización y entrenamiento del modelo de segmentación.

Anaconda es una distribución de Python que incluye una gran cantidad de paquetes y bibliotecas para ciencia de datos y aprendizaje automático. Anaconda ha facilitado la instalación y gestión de los entornos de desarrollo necesarios para este proyecto, incluyendo Jupyter Notebook y otras dependencias cruciales.

2. Mango

Mango es una herramienta de software utilizada para la segmentación de imágenes médicas. Mango ofrece una interfaz de usuario intuitiva y potentes capacidades de segmentación que permiten delinear y etiquetar estructuras anatómicas en imágenes médicas. En este TFG, Mango se ha utilizado para realizar la segmentación manual de las lesiones de cáncer de mama en las imágenes de RM, proporcionando las segmentaciones que se han utilizado como referencia para entrenar y evaluar el modelo de segmentación automática.

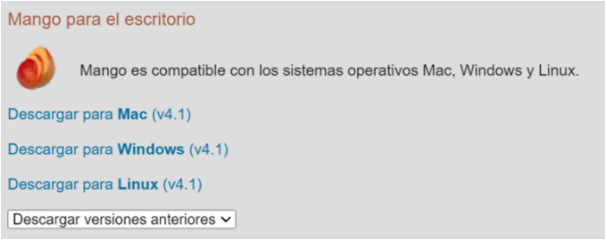


Figura 16. Herramienta de segmentación manual Mango

3. ITK-SNAP

ITK-SNAP es una aplicación de software utilizada para la visualización, segmentación y navegación de imágenes médicas tridimensionales. ITK-SNAP fue fundamental para la visualización de las imágenes de perfusión y sus segmentaciones correspondientes, permitiendo una evaluación detallada y precisa de los datos. La capacidad de ITK-SNAP para visualizar tanto las imágenes como las segmentaciones en 3D facilitó la verificación de la alineación y calidad de las segmentaciones realizadas.

4. Bibliotecas y Paquetes de Python

El desarrollo del modelo de segmentación y el preprocesamiento de datos se realizaron principalmente en Python, utilizando varias bibliotecas y paquetes especializados:

- **NumPy**: Utilizado para la manipulación de arrays y operaciones matemáticas básicas.
- **TensorFlow y Keras**: Bibliotecas de código abierto para el aprendizaje automático y la inteligencia artificial. **TensorFlow** se ha utilizado como backend para **Keras**, que ha proporcionado una interfaz de alto nivel para construir y entrenar el modelo de segmentación.
- **nibabel**: Biblioteca para la lectura y escritura de formatos de imágenes médicas, como NIfTI.
- **h5py**: Utilizado para manejar archivos en formato HDF5, permitiendo el almacenamiento eficiente de grandes volúmenes de datos de imágenes.
- **OpenCV**: Biblioteca de código abierto para la visión por computadora, utilizada para preprocesamiento de imágenes como filtrado y ajuste de contraste.
- **Matplotlib y Seaborn**: Utilizadas para la visualización de datos y resultados, permitiendo la creación de gráficos y diagramas para el análisis exploratorio y presentación de resultados.
- **TensorBoard**: Utilizada para la visualización de gráficos de entrenamiento de modelos de aprendizaje profundo, ayudando a monitorear el progreso y ajustar hiperparámetros.
- **tf.keras.callbacks**: Implementaciones de callbacks como ModelCheckpoint para guardar el modelo en intervalos regulares y TensorBoard para visualización del entrenamiento

4.2 Conjunto de datos

4.2.1 *Recopilación de Datos*

El estudio se ha realizado con un conjunto de datos proporcionados por el Hospital Universitario y Politécnico La Fe, en colaboración con el Grupo de Investigación Biomédica por Imagen (GIBI230) dentro del proyecto RadioVal. Este proyecto se centra en la investigación de imágenes médicas del cáncer de mama, con varios objetivos, desde el diagnóstico precoz hasta la toma de decisiones efectivas para determinar la aptitud para tratamientos y predecir la evolución de la enfermedad.

Para este TFG, se ha seleccionado un estudio que incluye a 21 pacientes. Las imágenes utilizadas son de perfusión de resonancia magnética (RM), específicamente las segmentaciones realizadas durante la fase dos de la perfusión sobre una potenciación T1. Cada estudio de perfusión tiene una media de 192 cortes por paciente, con una resolución de 512x512 píxeles y 7 fases temporales por estudio, aunque solo nos quedaremos con la fase dos, donde se realizaron las segmentaciones.

Dado que el análisis se ha realizado en 2D, el conjunto de datos para este TFG comprende aproximadamente de 4.000 imágenes de perfusión con sus correspondientes 4.000 imágenes de segmentación.

La estructura del conjunto de datos se realiza de manera clara y eficiente para facilitar el acceso y el procesamiento de los datos. En una carpeta principal, llamada Dataset, se almacenan todos los estudios, cada uno en una carpeta numerada de manera secuencial. Cada una de estas carpetas contiene dos subcarpetas: una para las imágenes de perfusión y otra para las segmentaciones.

Inicialmente, los documentos estaban en formato DICOM, donde cada archivo DICOM correspondía a un corte de la perfusión. Para simplificar la visualización y manejo de los datos, se han convertido estos archivos al formato NIfTI. Esta conversión ha permitido consolidar todos los archivos DICOM de un estudio en un único archivo NIfTI, haciendo mucho más manejable el conjunto de datos.

A continuación, se muestra en la siguiente figura, la estructura final del conjunto de datos organizado.

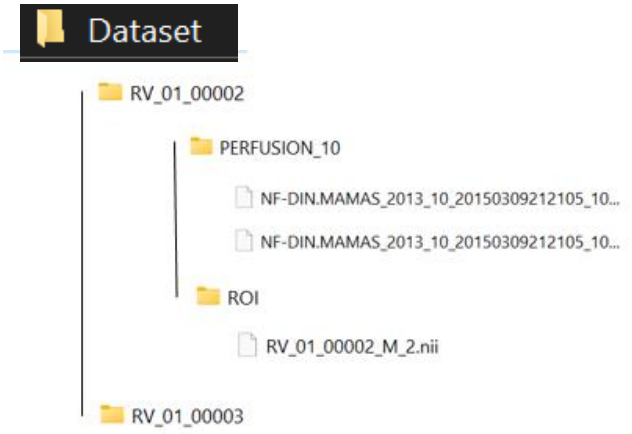


Figura 17. estructura del dataset de estudio

4.2.2 *Preprocesamiento*

A continuación, se describe el proceso que han tenido los datos desde que se ha determinado la estructura final del dataset, hasta que entran a la red.

- **Visualización de datos y flipping**

En primer lugar, se visualizaron los datos con ITK-SNAP superponiendo las segmentaciones con las perfusiones correspondientes, en este proceso se identificó que las segmentaciones aparecían volteadas sobre el eje Y, por lo que la zona de interés no concordaba con la lesión.

Se recorrió para cada segmentación se aplicó un “flipp” (voltear la imagen) sobre el eje Y, y tras esta conversión, quedaron alineadas perfectamente con las lesiones, para que la red pueda aprender donde tiene que segmentar, después de voltearlas las guardamos en un nuevo archivo NIfTI, en la siguiente figura se puede observar este manejo de datos.

```python
# Ruta al archivo de segmentación NIfTI
segmentation_path = r'C:/Users/joaqu/OneDrive - URV/SEGMENTACION/joaquin/RV_01_00012/ROI/RV_01_00012_M_1.nii'

# Cargar la imagen de segmentación
seg_nii = nib.load(segmentation_path)
seg_data = seg_nii.get_fdata()

# Hacer flip en el eje Y
seg_data_flipped = seg_data[:, ::-1, :]

# Crear un nuevo objeto NIfTI con los datos volteados
new_seg_nii = nib.Nifti1Image(seg_data_flipped, seg_nii.affine, seg_nii.header)

# Definir la ruta para guardar la imagen de segmentación volteada
flipped_path = os.path.join(os.path.dirname(segmentation_path), 'RV_01_00012_M_1_flipped.nii')

# Guardar la imagen volteada
nib.save(new_seg_nii, flipped_path)

print(f"Flipped segmentation saved to: {flipped_path}")
```

Figura 18. Proceso de volteo de segmentaciones

- **Extracción de la fase de perfusión correcta**

Una vez se han tenido todos los datos correctamente alineados y ajustados con las perfusiones, el siguiente paso ha sido extraer la fase específica de perfusión donde se han realizado las segmentaciones. Cada estudio contiene siete fases de exploración, correspondientes a distintos instantes temporales de la exploración diagnóstica. Se ha centrado en la fase dos, que es donde la radióloga realizó las segmentaciones.

Para facilitar la manipulación y limpieza de los datos, se ha creado un nuevo archivo .h5. Este archivo contiene arrays que incluyen las imágenes de segmentación, las imágenes de perfusión y el número de caso correspondiente. Este proceso ha permitido seleccionar solo los datos relevantes y preparar el conjunto para su uso en la red neuronal.

Dado que el estudio se lleva a cabo en 2D, al extraer la fase dos, se ha aprovechado para separar cada corte de las diferentes exploraciones y organizarlas en dos arrays: uno para las imágenes de perfusión y otro para las segmentaciones. Cada posición en estos arrays corresponde a la pareja de imágenes de perfusión y segmentación, asegurando que estén correctamente pareadas. Además, se ha creado un tercer array que indica a qué paciente pertenece cada corte, evitando así la fuga de datos, es decir, asegurando que las imágenes de un mismo paciente no se mezclen en conjuntos de datos diferentes.

Esta organización permite mantener una estructura clara y evitar problemas de mezcla de datos durante el entrenamiento del modelo, garantizando que las imágenes relacionadas se mantengan juntas para una evaluación coherente y precisa.

```python
# FUNCION CARGAR ARCHIVO CON CASOS
def load_data(file_path):
    with h5py.File(file_path, 'r') as file:
        X = file['perfusion'][:]  # Imágenes de perfusión como entrada al modelo
        Y = file['segmentation'][:]  # Máscaras de segmentación como etiquetas para entrenamiento
        case_indices = file['case_indices'][:]  # Índices de caso
    return X, Y, case_indices
```

Figura 19. Función para cargar el archivo de datos extraídos

- **Filtrado de imágenes**

En los arrays mencionados anteriormente, contamos con un total de 4.000 imágenes de perfusión (input) y 4.000 segmentaciones (outputs). Sin embargo, al revisar estos datos, nos dimos cuenta de que más del 60% de las imágenes no contienen lesión y por tanto no son útiles para que el modelo aprenda. Estas imágenes de fondo no contienen ninguna información valiosa para el entrenamiento del modelo, ya que no tienen lesión.

Dado que estos cortes sin máscara no aportan valor al proceso de segmentación y podrían incluso perjudicar el rendimiento del modelo, hemos decidido filtrarlas y enfocarnos únicamente en las imágenes de interés. Para lograr esto, hemos establecido un criterio de selección basado en la presencia de zonas de interés en las segmentaciones. Específicamente, buscamos aquellas imágenes donde al menos un píxel de la segmentación este clasificado como 1, indicando la presencia de una lesión o área relevante.

Este proceso de filtrado nos ha permitido eliminar los cortes sin máscara y concentrarnos en las imágenes que realmente contienen información significativa. Al hacerlo, optimizamos el conjunto de datos para que el modelo de segmentación pueda aprender de manera más eficiente y precisa, enfocándose en las características relevantes que son cruciales para la detección de las lesiones del cáncer de mama. Esta selección cuidadosa de los datos no solo mejora la calidad del entrenamiento, sino que también aumenta la eficacia del modelo en la identificación de zonas de interés en futuras imágenes de perfusión.

Como se puede observar en la figura 20, nos hemos quedado con un total de 748 imágenes de interés para el estudio.

```
# Identificar máscaras con al menos un píxel distinto de cero
masks_with_segmentation = np.array([np.any(mask > 0) for mask in segmentation_array])

# Filtrar los arrays de perfusión y segmentación para conservar solo los que tienen segmentación
filtered_perfusion_array = perfusion_array[masks_with_segmentation]
filtered_segmentation_array = segmentation_array[masks_with_segmentation]

print("Número original de imágenes:", perfusion_array.shape[0])
print("Número de imágenes después de filtrar:", filtered_perfusion_array.shape[0])
```

```
Número original de imágenes: 3992
Número de imágenes después de filtrar: 748
```

Figura 20. Proceso de filtrado de cortes de interés

- **Ordenar**

Después de hacer la selección de los cortes sin mascara cabe la posibilidad de que los arrays no esten ordenados por casos, es decir que cortes de un mismo caso esten al principio y al final del array, por lo que nos hemos asegurado de volverlos a ordenar.

El tercer array de número de casos nos ha ayudado a ordenar estos datos mediante una función que se ha creado con np.argsort y se muestra en la siguiente figura.

```
# FUNCION PARA ORDENAR LOS ARRAYS SEGUN INDICE DE CASO
def sort_data_by_case(X, Y, case_indices):
    # Ordenar los arrays por el array de índices de caso
    sorted_indices = np.argsort(case_indices)
    sorted_X = X[sorted_indices]
    sorted_Y = Y[sorted_indices]
    sorted_case_indices = case_indices[sorted_indices]
    return sorted_X, sorted_Y, sorted_case_indices
```

Figura 21. Código función de ordenación de arrays según número de caso

Como se ve en la anterior figura, la función `sort_data_by_case` toma tres arrays y los ordena según los valores en `case_indices`, asegurando que los datos de perfusión (X), las segmentaciones (Y) y los índices de caso (`case_indices`) estén todos alineados y ordenados de manera coherente. Esto es útil para garantizar que los datos se procesen y analicen de manera organizada, manteniendo la correspondencia entre las imágenes y sus respectivos índices de caso.

- **División del conjunto de datos**

Una vez que ya están ordenados los datos es más fácil dividir el conjunto sin provocar fuga de datos, esta división se ha realizado en 4 principales pasos que construyen la función de Split_data que se ve en la siguiente figura, donde la división se ha hecho por número de casos y no por número de imágenes en el array, así nos aseguramos de que todos los cortes de un mismo caso estén en un único conjunto de datos.

Respecto a la división de datos hemos decidido asignar un 70% para el entrenamiento un 20% para validación y un 10% para test.

```python
# FUNCION PARA DIVISION DEL CONJUNTO DE DATOS EN FUNCION DEL CASO
def split_data(sorted_X, sorted_Y, sorted_case_indices, train_frac=0.7, val_frac=0.2):
    # Identificar los casos únicos y calcular las proporciones para la división de datos
    unique_cases = np.unique(sorted_case_indices)
    num_cases = len(unique_cases)
    num_train = int(num_cases * train_frac)
    num_val = int(num_cases * val_frac)

    # Asignar los casos a cada conjunto
    train_cases = unique_cases[:num_train]
    val_cases = unique_cases[num_train:num_train + num_val]
    test_cases = unique_cases[num_train + num_val:]

    # Crear máscaras booleanas para filtrar los conjuntos
    train_mask = np.isin(sorted_case_indices, train_cases)
    val_mask = np.isin(sorted_case_indices, val_cases)
    test_mask = np.isin(sorted_case_indices, test_cases)

    # Extraer los subconjuntos de datos basados en las máscaras
    X_train, Y_train = sorted_X[train_mask], sorted_Y[train_mask]
    X_val, Y_val = sorted_X[val_mask], sorted_Y[val_mask]
    X_test, Y_test = sorted_X[test_mask], sorted_Y[test_mask]

    return (X_train, Y_train), (X_val, Y_val), (X_test, Y_test)
```

Figura 22. Función de división de datos por número de caso

Paso 1: Identificación de Casos Únicos

El objetivo es determinar los identificadores únicos de cada caso clínico.

Se usa `np.unique` para extraer los índices únicos de casos de `sorted_case_indices`.

Paso 2: Asignación de Casos a Subconjuntos

Cálculo de Proporciones: Calcula cuántos casos se asignan a entrenamiento (`train_frac`), validación (`val_frac`) y prueba (restante).

Paso 3: Creación de Máscaras Booleanas para Filtrado

el objetivo es generar máscaras para identificar qué imágenes pertenecen a cada conjunto.

Crea máscaras booleanas (`train_mask`, `val_mask`, `test_mask`) que correspondan a los casos asignados a cada subconjunto.

Paso 4: Extracción de Subconjuntos de Datos

Uso de Máscaras: Filtra `sorted_X` y `sorted_Y` con las máscaras para obtener:

X_train, Y_train

X_val, Y_val

X_test, Y_test

La distribución final del conjunto de datos se refleja en la siguiente figura, muestra la cantidad de cortes que pertenecen a cada conjunto y el número caso asignado a cada conjunto.

```
Entrenamiento: 503, Validación: 148, Prueba: 97
Casos en entrenamiento: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13]
Casos en validación: [14 15 16 17]
Casos en prueba: [18 19 20]
```

Figura 23. Distribución de casos para los conjuntos

- **Redimensionar**

Las imágenes tienen una dimensión de 512x512, debido al limitado equipo del que se dispone, se ha necesitado redimensionar el conjunto de datos para poder entrenar a la red con un poco más de ligereza y bajar el tiempo de entrenamiento. La redimensión ha realizado a 256x256 mediante la función que se ve en la figura 24, que se ha aplicado a cada conjunto de datos (train, test y val):

```python
def resize_images(X, new_size=(256, 256)):
    if X.ndim == 3:  # Solo (n_samples, height, width)
        X = np.expand_dims(X, axis=-1)
    return np.array([tf.image.resize(img, new_size).numpy() for img in X])
```

Figura 24. Función redimensión de imágenes

La función `resize_images` toma como entrada un array de imágenes (`X`) y un nuevo tamaño (`new_size`), con un valor predeterminado de 256x256 píxeles). La función verifica las dimensiones del array de entrada y ajusta las imágenes al nuevo tamaño especificado. El propósito es transformar todas las imágenes del conjunto a un tamaño uniforme, facilitando así su uso en modelos de segmentación y clasificación.

- **Normalización**

Y, por último, paso crucial para tener todos los datos listos, la normalización de las imágenes.

La normalización es un paso crucial en el preprocesamiento de imágenes para modelos de segmentación, ya que asegura que todas las imágenes tengan un rango de valores comparable, mejorando así la eficiencia y eficacia del entrenamiento del modelo. En este proyecto, se ha implementado una técnica de normalización basada en la escala Min-Max. Donde se ha normalizado por número de caso. A continuación, se detalla el procedimiento seguido para normalizar las imágenes utilizadas en el desarrollo del modelo de segmentación de lesiones de cáncer de mama.

La normalización Min-Max ajusta los valores de los píxeles de una imagen para que se encuentren dentro de un rango definido, típicamente entre 0 y 1. Esto se logra mediante la siguiente fórmula:

$$z = \frac{\mathrm{x} - min(x)}{[\max(x) - min(x)]} \qquad (1)$$

La implementación de la normalización Min-Max en el preprocesamiento de las imágenes ha sido un paso fundamental en la preparación de los datos para el modelo de segmentación. Este proceso no solo mejora la consistencia y estabilidad de los datos, sino que también contribuye significativamente a la eficacia del entrenamiento del modelo, permitiendo una mejor detección las lesiones de cáncer de mama en las imágenes de resonancia magnética.

Es importante normalizar por caso para que las intensidades de diferentes casos no influyan en los demás y poder individualizar correctamente cada caso. Como se observa en la función de la siguiente figura identificamos las imágenes de cada caso y las normalizamos.

```python
# FUNCION PARA NORMALIZAR POR CASOS
def normalize_min_max_by_case(X, case_indices):
    unique_cases = np.unique(case_indices)
    X_norm = np.zeros_like(X, dtype=np.float32)

    for case in unique_cases:
        case_mask = (case_indices == case)
        case_images = X[case_mask]
        min_val = case_images.min()
        max_val = case_images.max()
        if max_val > min_val:  # Evita la división por cero
            X_norm[case_mask] = (case_images - min_val) / (max_val - min_val)

    return X_norm
```

Figura 25. Función de normalización por caso

4.3 Diseño del Modelo

Descripción General del Modelo

El modelo inicial desarrollado para la segmentación de lesiones de cáncer de mama en imágenes médicas es una adaptación de la arquitectura U-Net. La U-Net es particularmente reconocida por su eficiencia en la segmentación semántica de imágenes biomédicas, donde la precisión y la conservación de los detalles son cruciales. Este modelo se caracteriza por su estructura simétrica que facilita la localización precisa de las regiones de interés en imágenes médicas, haciendo especial énfasis en la captura de contextos y detalles a diferentes escalas.

El diseño inicial de la arquitectura U-Net utilizada en este proyecto se basa en una implementación básica de U-Net disponible en el repositorio público de GitHub proporcionado por David Revelo Luna. La implementación específica puede encontrarse en https://github.com/DavidReveloLuna/Semantic-Segmentation-Basic-U-Net .Esta referencia ha servido como punto de partida para el desarrollo y la adaptación de la red a las necesidades específicas de la segmentación de imágenes médicas para el diagnóstico de cáncer de mama. Se han realizado modificaciones y mejoras en la arquitectura original para optimizar su rendimiento en nuestro conjunto de datos específico, incluyendo ajustes en el número y tamaño de los filtros y la inclusión de técnicas de regularización como el dropout y la regularización L2. [23]

Arquitectura de la Red

- **Entrada del Modelo**:

La red recibe imágenes de tamaño 256×256256 \times 256256×256 píxeles con un solo canal (escala de grises), indicado por el input_shape= (256, 256, 1).

- **Camino Contractivo (Codificador)**:

Primera Capa Convolutiva: Compuesta por dos capas de convolución de 64 filtros cada una, con un tamaño de kernel de 3×33 \times 33×3, seguida de una función de activación ReLU y un padding 'same' para mantener el tamaño de la imagen.

Pooling: Sigue una capa de MaxPooling de 2×22 \times 22×2 que reduce a la mitad las dimensiones de la imagen, ayudando a incrementar el campo receptivo y reducir el overfitting.

Este patrón se repite incrementando el número de filtros en cada bloque: 128, 256 y 512 filtros en las sucesivas capas de convolución, intercaladas con capas de MaxPooling, profundizando en la red y reduciendo la dimensión espacial mientras aumenta la profundidad de características.

- **Capa Intermedia**:

La capa más profunda contiene 1024 filtros, permitiendo que la red capture las características más complejas de la imagen.

- **Camino Expansivo (Decodificador)**:

Primera Capa de Convolución Transpuesta: Utiliza convoluciones transpuestas o deconvoluciones para comenzar el proceso, incrementando las dimensiones de las imágenes mientras reduce el número de filtros.

Concatenación: Cada capa de salida de las convoluciones transpuestas se concatena con la característica correspondiente del camino contractivo. Esto ayuda a recuperar la información espacial perdida durante el downsampling.

Este patrón de convoluciones transpuestas y concatenaciones continúa, reduciendo gradualmente la profundidad de las características a 512, 256, 128 y finalmente 64 filtros.

- **Salida del Modelo**:

La última capa es una convolución de 1×11 \times 11×1 que reduce la profundidad al final, aplicando una función de activación sigmoide para obtener una máscara binaria que predice la presencia de lesiones en cada píxel.

- **Compilación del Modelo**:

El modelo utiliza el optimizador Adam (con un learning rate de 0,0001), conocido por su eficiencia en problemas de optimización no convexos en grandes conjuntos de datos. La función de pérdida es la 'binary_crossentropy', adecuada para problemas de clasificación binaria como la segmentación de imágenes, y se monitorizan la precisión y el coeficiente de Dice como métricas para evaluar el rendimiento del modelo.

Esta arquitectura U-Net es la estructura principal y configuración inicial con la que se ha entrenado en un primer momento, a partir de esta arquitectura realizamos varias modificaciones como habíamos comentado, con el objetivo de encontrar la configuración óptima para nuestros datos.

4.4 Entrenamiento

4.4.1 *Optimizador*

Se ha utilizado el optimizador Adam debido a su eficacia demostrada en numerosas tareas de aprendizaje profundo. Adam es preferido por su manejo automático del tamaño de los pasos de aprendizaje y su adaptabilidad a diferentes formas de los datos, lo cual es crucial para la segmentación de imágenes médicas donde las variaciones entre las imágenes pueden ser significativas.

El proceso de entrenamiento del modelo U-Net para la segmentación de lesiones de cáncer de mama ha comenzado con una evaluación preliminar que reveló que las métricas de rendimiento eran invariables a lo largo del tiempo, indicando la necesidad de ajustes en la configuración de entrenamiento para mejorar la capacidad de aprendizaje del modelo.

Para abordar la falta de progreso en las métricas de rendimiento, se ha reducido la tasa de aprendizaje del optimizador Adam a 0,0001. Este cambio busca proporcionar un avance más gradual y controlado durante el entrenamiento, permitiendo que el modelo se ajuste de manera más efectiva a las complejidades de los datos de segmentación.

4.4.2 *Función de Pérdida*

Se ha seleccionado la función de pérdida binary_crossentropy por su relevancia en problemas de clasificación binaria, como es el caso de la segmentación de imágenes donde cada píxel se clasifica como perteneciente o no a una lesión. Esta función mide el error entre las máscaras binarias predichas y las verdaderas, proporcionando una cuantificación directa de la precisión de las predicciones del modelo. Pero debido a que los resultados obtenidos no son los más esperados, se ha decidido indagar en otras funciones de perdida que puedan ajustarse más a nuestro conjunto de datos, ya que nuestro conjunto de datos peca de tener clases desbalanceadas por tener un gran volumen de pixeles clasificados como fondo (0) y pocos como lesión (1). Se ha llegado a la conclusión que puede ajustarse la función de Tversky loss, que se caracteriza por ser buena opción para conjuntos de datos desbalanceados.

4.4.3 *Callbacks*

- **ModelCheckpoint:** Se ha configurado para guardar el modelo después de cada conjunto de 10 épocas, permitiendo la reanudación del entrenamiento sin pérdida de progreso y facilitando la recuperación de la versión óptima del modelo en cualquier punto del entrenamiento.

  Dado el hardware limitado disponible, se ha implementado un sistema de checkpoints para guardar el estado del modelo cada 10 épocas utilizando *ModelCheckpoint* de Keras. Esto no solo mitigaba el riesgo de pérdida de progreso en caso de interrupciones, sino que también permitía evaluar y ajustar continuamente el enfoque de entrenamiento sin sobrecargar los recursos del sistema.

  El modelo se ha entrenado en tramos de 10 épocas, alcanzando un total de 150 épocas. Este enfoque en tramos ha permitido ajustes iterativos en la configuración del entrenamiento y la evaluación detallada del progreso del modelo.

- **TensorBoard:** Se ha utilizado para proporcionar visualizaciones en tiempo real del proceso de entrenamiento, permitiendo un seguimiento detallado del progreso de las métricas y la función de pérdida a lo largo de las épocas. Esto es crucial para realizar ajustes oportunos en la configuración del entrenamiento y para documentar el proceso para análisis futuros o revisiones académicas.
- **Reduce_lr:** Se ha implementado en una de las pruebas finales para ajustar dinámicamente el learning rate cuando el rendimiento del modelo medido mediante la evaluación de los en el conjunto de validación, se estanca y no mejora.

4.4.4 *Implementación de Técnicas de Regularización*

Con el fin de combatir el sobreajuste observado en las primeras pruebas, se han implementado modificaciones sucesivas en la arquitectura del modelo:

1. **Incorporación de Dropout:** Se han añadido capas de dropout con una probabilidad de 0,5 después de cada bloque convolucional. Esta técnica ayuda a reducir el sobreajuste al "desconectar" aleatoriamente ciertas neuronas durante el entrenamiento, lo que impide que el modelo dependa demasiado de cualquier conjunto pequeño de neuronas y fomente una generalización más robusta.
2. **Regularización L2:** Se ha aplicado la regularización L2 a los pesos de las capas convolucionales con un coeficiente inicial de 0,001, que más tarde se ajustó a 0,0001 en respuesta a la dinámica del entrenamiento. La regularización L2 penaliza los pesos grandes, simplificando el modelo y evitando que se ajuste excesivamente a los datos de entrenamiento.
3. **Batch normalization:** Se ha podido aplicar en el último modelo de la investigación**,** para poder combatir el sobreajuste y acelerar el proceso de aprendizaje.

4.4.5 *Modelos entrenados*

Modelo	Épocas	F. loss	Learning rate	Dropout	Regularizador L2
1	150	binary_crossentropy	0,0001	X	X
2	150	binary_crossentropy	0,0001	0,5	X
3	150	binary_crossentropy	0,0001	0,5	0,01
4	150	binary_crossentropy	0,0001	0,3	0,0001
5	150	Tversky loss	0,0001	0,3	0,0001
6	130	Tversky loss	Reduce_lr	0,5	0,0001

Tabla 3. Parámetros de entrenamiento de los modelos**.**

4.4.6 *Métricas de Evaluación*

Para evaluar el rendimiento del modelo, se han empleado dos métricas principales:

- **Accuracy (Exactitud):** Mide la proporción de etiquetas predichas correctamente en relación con el total de etiquetas. Es útil para obtener una visión rápida de la eficacia general del modelo.
- **Coeficiente de Dice:** Específicamente adaptado para la evaluación de modelos de segmentación, el coeficiente de Dice compara la similitud entre las máscaras predichas

y las verdaderas, siendo especialmente útil en contextos donde las clases están desbalanceadas, como en segmentaciones médicas donde las regiones de interés suelen ser mucho menores que el fondo.

- **Coeficiente de correlación de Matthews (MCC):** es particularmente útil en situaciones de clasificación binaria y problemas de clasificación con clases desequilibradas. El MCC considera verdaderos y falsos positivos y negativos y es generalmente considerado como una medida balanceada que puede ser utilizada incluso si las clases son de tamaños muy diferentes, como es nuestro caso.
- **Loss:** Se ha utilizado ya que es crucial para la evaluación de aprendizaje automático, representa la diferencia entre las predicciones y los valores reales.

División de Datos

Como hemos descrito anteriormente, el conjunto de datos se ha dividido de tal manera que el 70% de los datos se asignaron al grupo de entrenamiento, un 20% al grupo de validación, y el 10% restante al grupo de prueba. Esto se traduce en la asignación de 503 imágenes correspondientes a los casos del 0 al 13 para el entrenamiento. Para la validación se han asignado148 imágenes que corresponden a los casos del 14 al 17. Finalmente, el grupo de prueba recibe 97 imágenes que pertenecen a los casos del 18 al 20. Esta distribución asegura que cada grupo contiene una representación proporcional y adecuada del rango completo de casos, permitiendo una evaluación efectiva y equilibrada del modelo de segmentación.

```
Entrenamiento: 503, Validación: 148, Prueba: 97
Casos en entrenamiento: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13]
Casos en validación: [14 15 16 17]
Casos en prueba: [18 19 20]
```

5 Resultados

5.1 Métricas y configuración

5.1.1 *Modelo 1*

Iniciamos con un el primer modelo**,** el más básico**,** este, tiene la configuración básica de la U-net sin dropout ni ningún tipo de regularizador, simplemente la arquitectura que se ha introducido en la metodología.

- **Configuración**

Optimizador: Adam.
Función de perdida: Binary_crossentropy.
Learning rate: 0,0001
Callbacks: ModelCheckpoint, TensorBoard.
Batch Size: 8

- **Métricas**

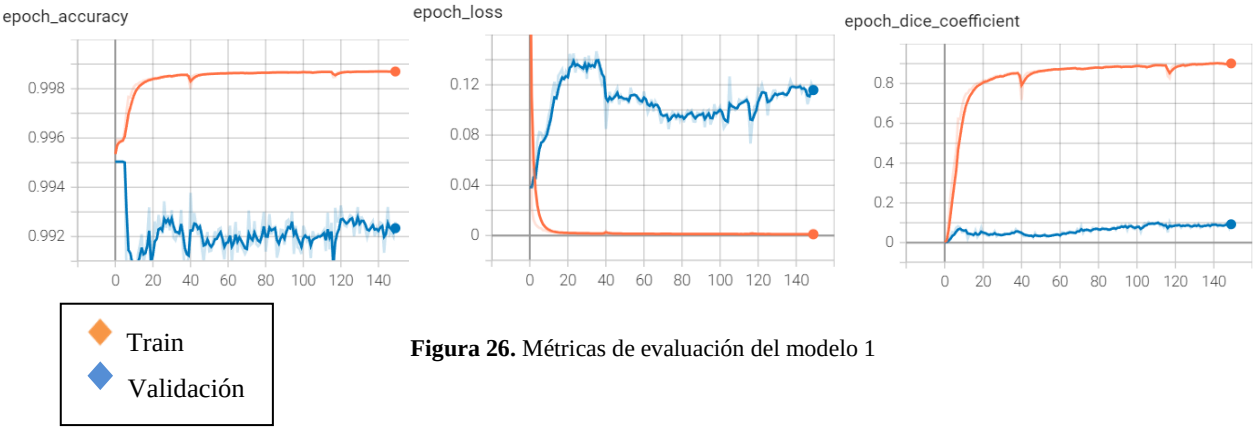


Figura 26. Métricas de evaluación del modelo 1

En estas métricas se ve un claro sobreajuste en el modelo, como se puede observar en las gráficas del coeficiente de Dice y de pérdida. Desde la primera época, la curva de pérdida muestra un cruce entre los datos de entrenamiento y validación, un indicativo clásico de sobreajuste. Esto sugiere que, mientras el modelo aprende eficientemente los datos de entrenamiento, ajustándose excesivamente a sus

características y ruido específicos, falla en generalizar adecuadamente a nuevos ejemplos no vistos, lo que se refleja en su rendimiento en el conjunto de validación.

Similarmente, la métrica del coeficiente de Dice confirma esta tendencia, alcanzando altos valores en el entrenamiento, lo que indica una alta coincidencia con las máscaras de segmentación originales. Sin embargo, su desempeño en la validación es significativamente inferior, mostrando que el modelo no es capaz de mantener la misma precisión en los datos desconocidos. Esto nos indica la necesidad de implementar estrategias adicionales para mejorar la capacidad del modelo para generalizar, como puede ser la integración de técnicas de regularización más efectivas, ajustes en la arquitectura de la red o la optimización de los parámetros de entrenamiento.

Como vemos la precisión es alta, pero el rendimiento general del modelo no es bueno. Esto ocurre porque la métrica de precisión no penaliza adecuadamente los píxeles que no contienen lesión y, en consecuencia, los considera como clasificaciones correctas. Debido al desbalance significativo de clases, donde la mayoría de los píxeles no representan lesiones, la precisión aparenta ser engañosamente elevada. Esto nos lleva a concluir que la precisión, como métrica, no proporciona información valiosa para nuestro caso específico, razón por la cual decidimos descartarla como indicador de rendimiento en futuras evaluaciones.

5.1.2 *Modelo 2*

Tras observar que las métricas del primer modelo indican un rendimiento insatisfactorio y un claro sobreajuste, se ha decidido implementar una estrategia de regularización para mejorar la generalización del modelo, añadiendo capas de dropout con una tasa del 0,5 en cada bloque convolucional. Con la expectativa de que el modelo muestre una mejora en la capacidad de generalizar para nuevos casos.

- **Configuración**

Optimizador: Adam.
Función de perdida: Binary_crossentropy.
Learning rate: 0.0001
Callbacks: ModelCheckpoint, TensorBoard.
Batch Size: 8
Dropout: 0,5

- **Métricas**

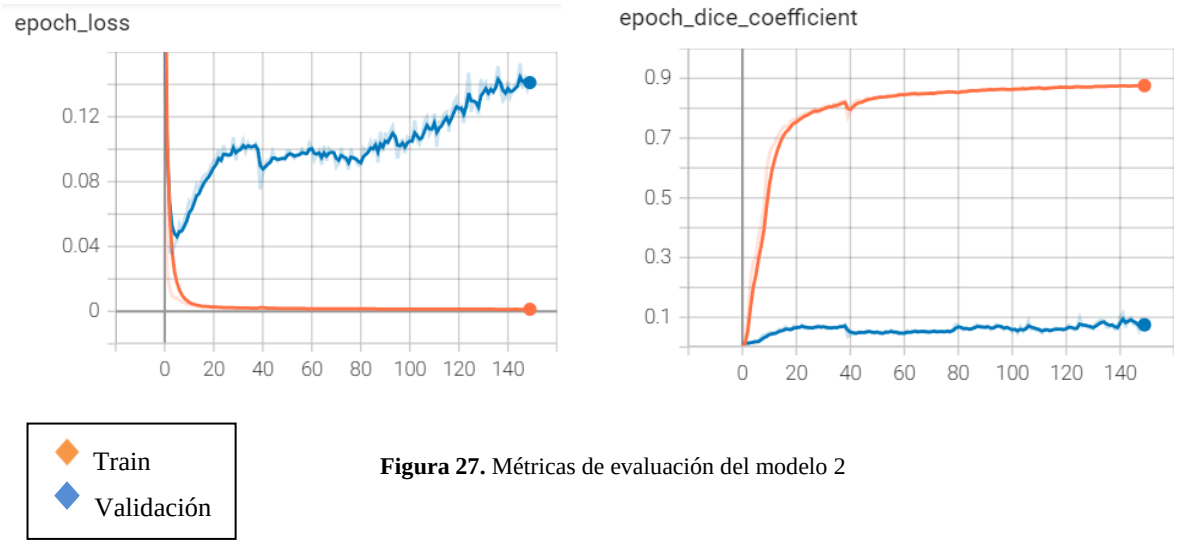


Figura 27. Métricas de evaluación del modelo 2

A pesar de la implementación del dropout, se siguen observando evidencias de sobreajuste. Las métricas resultantes del modelo modificado no han mostrado variaciones significativas en comparación con el modelo anterior, manteniendo patrones similares de rendimiento. Aunque se ha observado una ligera mejora en la métrica del DICE, alcanzando en algunas épocas hasta un 10% de coincidencia con las máscaras originales, estas mejoras no han sido considerables ni suficientes para concluir que el modelo ha superado el problema de generalización. Esto indica la necesidad de explorar otras estrategias de regularización o ajustes en los parámetros para combatir el sobreajuste de manera más efectiva.

5.1.3 *Modelo 3*

Tras observar que las capas de dropout no han logrado evitar el sobreajuste, hemos decidido incorporar un regularizador L2, como se detalla en la sección de metodología. Se ha establecido una tasa de regularización de 0,001, manteniendo además la tasa de dropout en 0,5. Esta combinación de técnicas se ha implementado con el objetivo de reforzar la capacidad del modelo para generalizar mejor a nuevos datos, aprovechando las propiedades de regularización tanto del dropout como del L2 para reducir la complejidad del modelo y evitar el ajuste excesivo a los datos de entrenamiento.

- **Configuración**

  Optimizador: Adam.
  Función de perdida: Binary_crossentropy.
  Learning rate: 0,0001
  Callbacks: ModelCheckpoint, TensorBoard.
  Batch Size: 8
  Dropout: 0,5
  Regularizador L2: 0,001

- **Métricas**

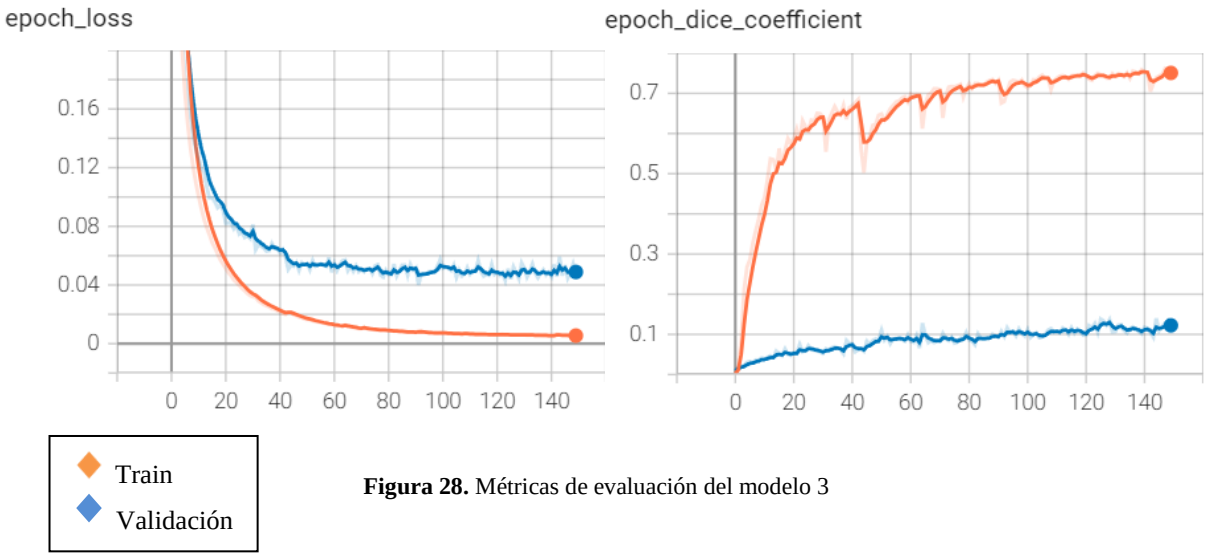


Figura 28. Métricas de evaluación del modelo 3

En este modelo, se observa una mejora significativa en la capacidad de evitar el sobreajuste. Esto se evidencia en la gráfica de la pérdida por época (epoch_loss), donde tanto los datos de entrenamiento como de validación muestran una disminución gradual, siguiendo patrones similares, aunque en niveles diferentes. Esta mejora también se refleja de manera moderada en la gráfica del coeficiente de Dice (epoch_dice), donde se percibe un incremento leve en el valor de DICE para el conjunto de validación, superando el umbral del 10% que teníamos en el modelo anterior. Además, se observa una reducción en el valor de DICE para el conjunto de entrenamiento, lo que indica que el modelo se ajusta menos a los datos de entrenamiento, y generaliza algo mejor a nuevos casos.

5.1.4 *Modelo 4*

Tras lograr una mejora significativa y mitigar el sobreajuste, aún es necesario elevar los valores del coeficiente Dice y reducir los valores de pérdida para obtener un modelo con resultados aceptables. El cuarto modelo se ha concebido con un enfoque exploratorio, buscando optimizar aún más los resultados. Se ha decidido mantener las técnicas de regularización previamente implementadas, pero ajustando las métricas: se ha reducido la tasa de dropout a 0,3 y la tasa del regularizador L2 a 0,0001. Este ajuste tiene como objetivo afinar el equilibrio entre adaptabilidad y generalización, procurando una mejora continua en el desempeño del modelo en tareas de segmentación.

- **Configuración**

Optimizador: Adam.
Función de perdida: Binary_crossentropy.
Learning rate: 0.0001
Callbacks: ModelCheckpoint, TensorBoard.
Batch Size: 8
Dropout: 0,3
Regularizador L2: 0,0001

- **Métricas**

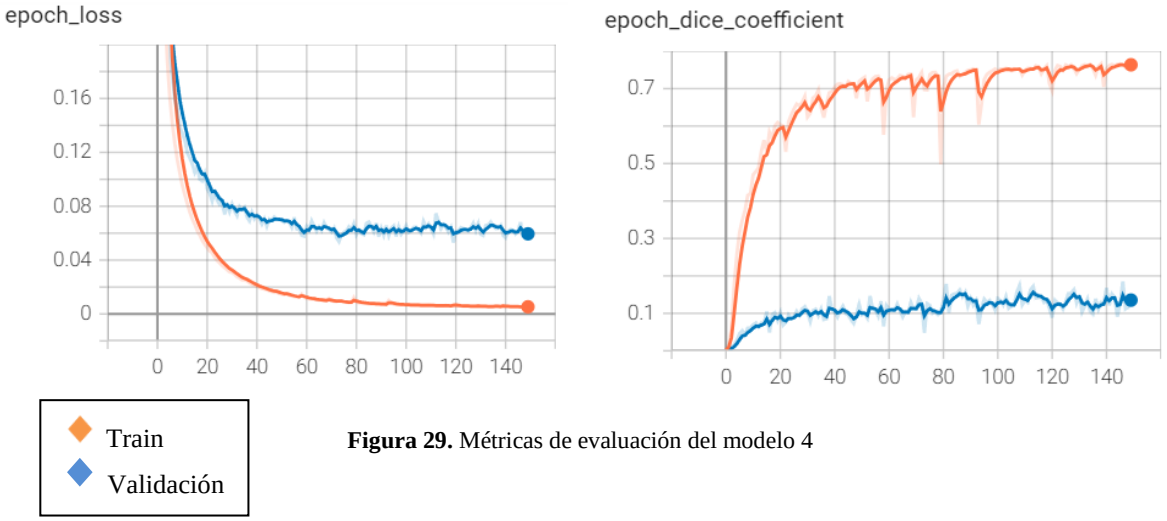


Figura 29. Métricas de evaluación del modelo 4

Los ajustes realizados al modelo han resultado en progresos positivos en comparación con el modelo anterior, evidenciado por una mejora en el coeficiente Dice del conjunto de validación, que ha llegado a superar el 15%. Este aumento muestra una tendencia de crecimiento moderado y constante, sin ajustarse en exceso a los datos de entrenamiento, donde los valores no superaron el 80%. Aunque la gráfica de pérdida (loss) ha mostrado valores ligeramente superiores a los del modelo previo, estos podrían considerarse comparables, ya que la variación ha sido mínima, oscilando sobre un 0,05 en comparación con el 0,04 del modelo anterior.

5.1.5 *Modelo 5*

Después de una evaluación global de las modificaciones implementadas, hemos observado un progreso constante, aunque moderado, por lo que hemos decidido adoptar un nuevo enfoque para mejorar y optimizar el rendimiento del modelo. En este punto de la investigación, se ha optado por enriquecer las métricas de evaluación incorporando el Coeficiente de Correlación de Matthews (MCC), reconocido por su utilidad en la predicción de clases binarias desbalanceadas, como es el caso de nuestro estudio. Además, hemos cambiado la función de pérdida a Tversky loss, decisión motivada por las mismas razones que nos han llevado a seleccionar el MCC. Estos cambios están dirigidos a adaptarnos mejor a las características específicas de nuestros datos y superar los resultados previos, manteniendo las técnicas y los parámetros que han demostrado ser efectivos en los modelos anteriores.

- **Configuración**

Optimizador: Adam.
Función de perdida: Tversky loss
Learning rate: 0,0001
Callbacks: ModelCheckpoint, TensorBoard.
Batch Size: 8
Dropout: 0,3
Regularizador L2: 0,0001

- **Métricas**

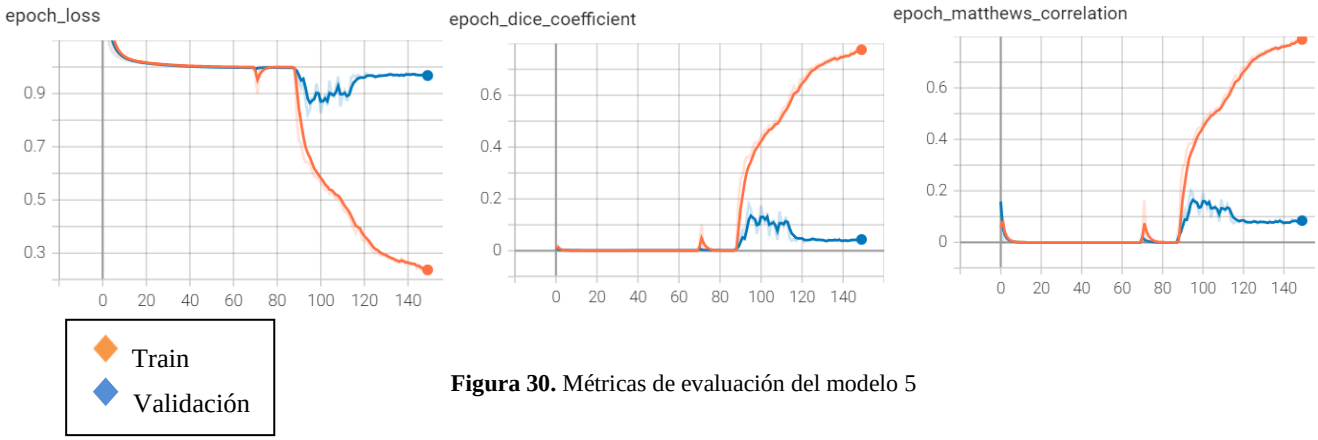


Figura 30. Métricas de evaluación del modelo 5

En la representación gráfica del quinto modelo, observamos un comportamiento inicialmente estancado, donde tanto el DICE como el MCC permanecen en valores cercanos a cero durante las primeras 80 épocas. Posteriormente, ambos indicadores experimentan un aumento abrupto, alcanzando niveles significativamente altos en el conjunto de entrenamiento (cerca del 80%) mientras que en validación apenas superan el 10%, y muestran una tendencia decreciente a medida que avanzan las épocas. Este patrón es indicativo de un sobreajuste.

Este sobreajuste se refleja igualmente en la gráfica de 'epoch_loss', que muestra una disminución lenta y constante en las pérdidas, hasta que ambas (entrenamiento y validación) experimentan mejoras marcadas. Sin embargo, la gráfica revela una divergencia creciente entre los dos conjuntos, especialmente después de la repentina mejora, donde la pérdida de validación comienza a incrementar mientras que la de entrenamiento sigue descendiendo, corroborando aún más la presencia de sobreajuste. Este comportamiento sugiere que, aunque se han hecho cambios importantes en la configuración, el modelo necesita más modificaciones para mejorar su capacidad de generalización.

5.1.6 *Modelo 6*

Viendo que el modelo anterior no ha logrado alcanzar los resultados esperados y ha mostrado signos claros de estancamiento, se ha decidido tomar medidas más drásticas para explorar el potencial de la función de pérdida Tversky. Un factor que podría estar contribuyendo al estancamiento es la dificultad del modelo para encontrar el equilibrio óptimo en los pesos de aprendizaje. Por ello, se ha implementado el callback de reducción de tasa de aprendizaje (reduce_lr), que ajusta automáticamente el learning rate cuando no se observan mejoras en la métrica 'epoch_loss'.

Adicionalmente, se ha aumentado la tasa de dropout al 0,3 y se ha introducido la técnica de Batch Normalization con el objetivo de acelerar el proceso de aprendizaje y mitigar aún más el sobreajuste. Estas adiciones, aunque beneficiosas para el rendimiento del modelo, incrementan la demanda computacional, lo que ha llevado a la necesidad de reducir el tamaño del lote a 4 para adaptarse a las limitaciones de hardware disponibles. Estos ajustes están orientados a proporcionar al modelo herramientas adicionales para afinar su capacidad de generalización y mejorar su rendimiento en la segmentación de imágenes médicas.

- **Configuración**

Optimizador: Adam.
Función de perdida: Tversky loss
Learning rate: 0,0001
Callbacks: ModelCheckpoint, TensorBoard, reduce_lr.
Batch Size: 4
Dropout: 0,5
Regularizador L2: 0,0001
Batch normalization

- **Métricas**

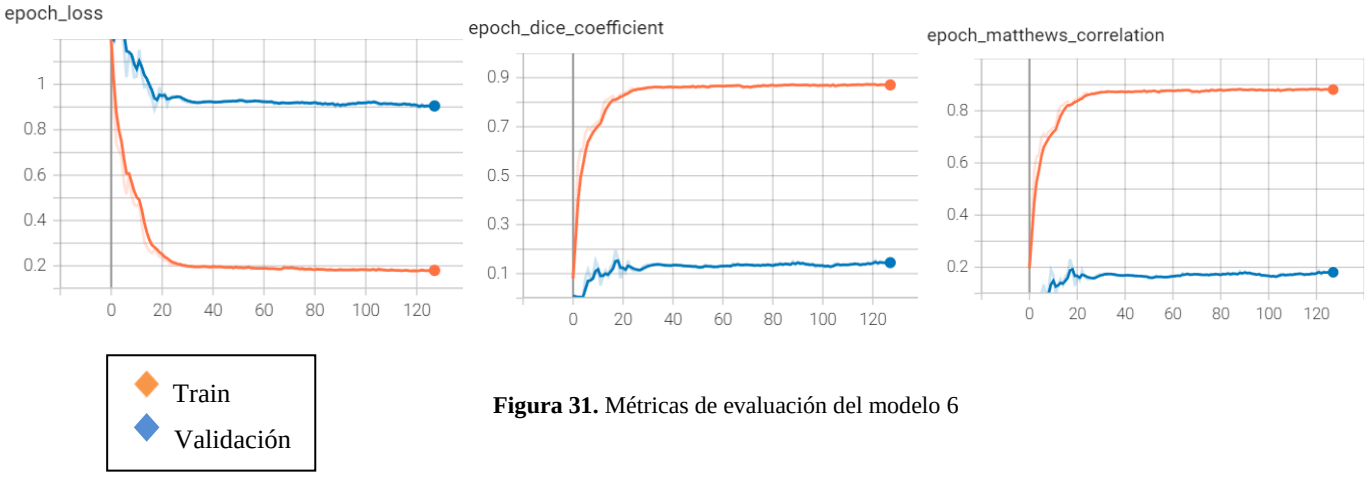


Figura 31. Métricas de evaluación del modelo 6

En este último modelo observamos una mejora significativa comparado con el modelo anterior, y una mejora moderada en comparación con los modelos que utilizaban la función de pérdida Binary_crossentropy. Los resultados se manifiestan en una métrica de Dice alrededor del 17% y un Coeficiente de Correlación de Matthews (MCC) cercano al 20%. Además, se aprecia una progresión constante y favorable en las métricas a lo largo de las épocas, indicando una mejora sostenida en la capacidad predictiva del modelo.

La gráfica de la pérdida (loss) para este modelo muestra un rendimiento sano, mostrando una disminución estable en el conjunto de entrenamiento y, aunque los valores siguen siendo relativamente altos en la validación, se observa una buena convergencia que indica una reducción del sobreajuste que padecía el modelo anterior

Este último modelo no solo supera ligeramente en rendimiento al **modelo 4**, considerado hasta ahora el mejor, sino que también demuestra un comportamiento más robusto en las predicciones. A continuación, veremos que este modelo presenta un rendimiento significativamente superior que los anteriores, lo cual será evidente en las visualizaciones de las predicciones y en cómo maneja los casos de prueba.

5.2 Predicciones

En esta sección se presentan tres predicciones correspondientes a los casos 18, 19 y 20, con el objetivo de proporcionar una perspectiva práctica sobre cómo el modelo maneja la segmentación en casos nuevos que nunca ha analizado. Para mantener una base de comparación constante, siempre hemos seleccionado el mismo corte de cada caso, lo que nos permite evaluar de manera consistente cómo diferentes modelos abordan idénticas situaciones de segmentación.

Además, se incluyen en la siguiente tabla, las métricas de evaluación para ofrecer una visión detallada del desempeño del modelo a lo largo de todo el conjunto de prueba. Esto nos ayuda a entender no solo cómo se comporta el modelo en casos individuales, sino también su efectividad general y precisión en un espectro más amplio de datos.

Métricas de predicción del conjunto de prueba

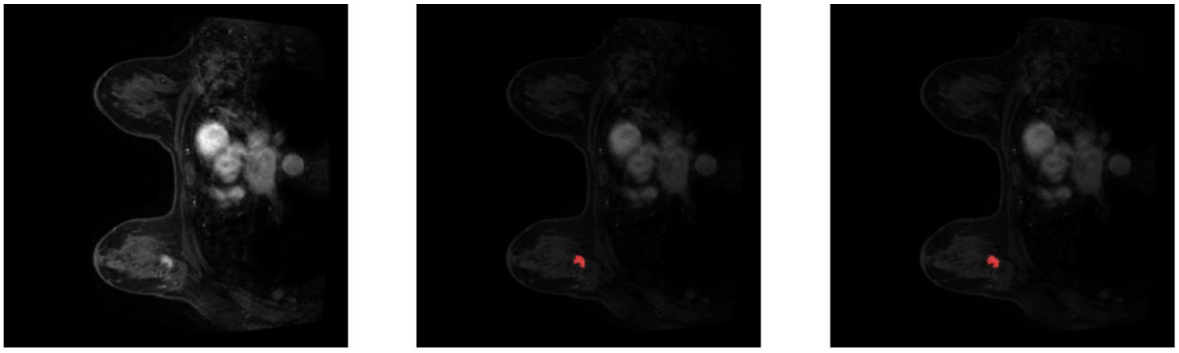
Modelo	Loss	DICE
1	0,11	6%
2	0,05	12,3%
3	0,05	9,4%
4	0,04	16,5%
5	0,87	13,5%
6	0,73	30,8%

Tabla 4. Métricas de predicción del conjunto de prueba de todos los modelos.

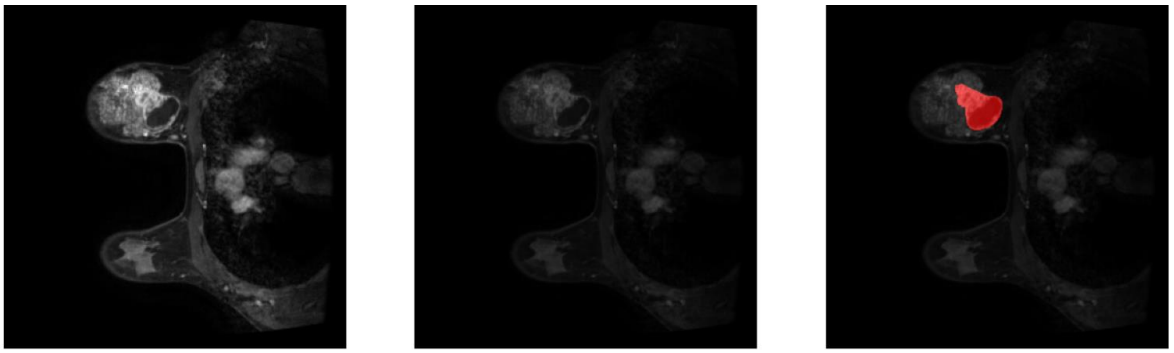
- **Modelo 1**

Imagen Normal	Predicción Binarizada	Ground Truth

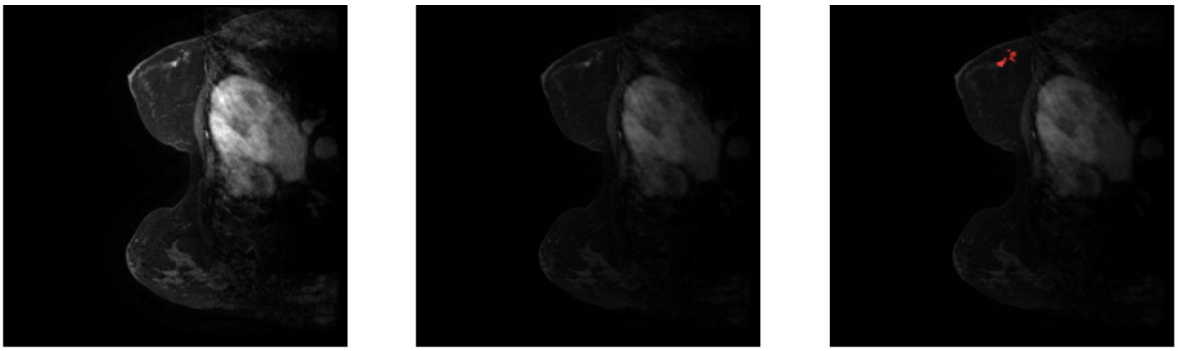
Caso 18



Caso 19



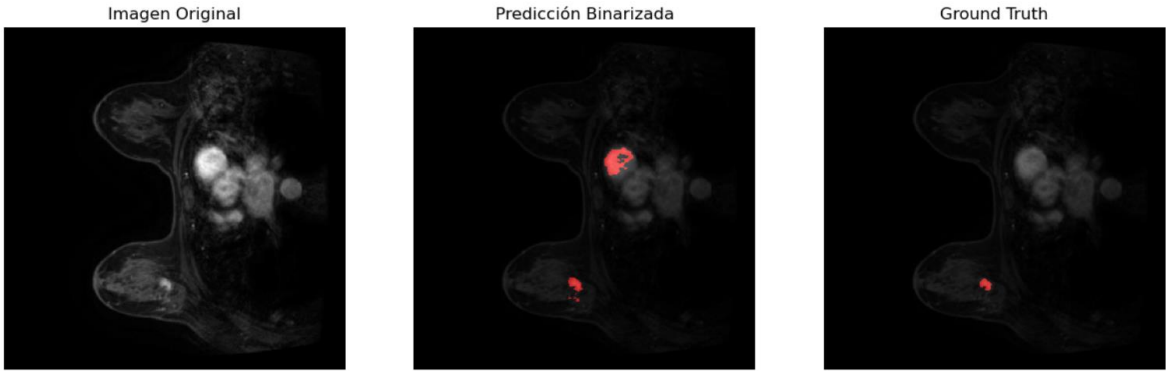
Caso 20



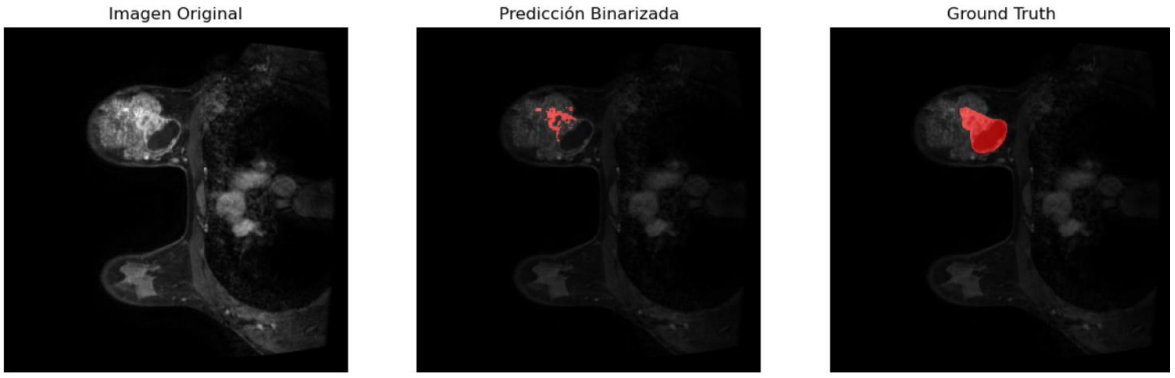
En el caso 18 hace una buena predicción, pero como vemos en las siguientes imágenes no es capaz de clasificar los pixeles del tumor como se ve en el ground truth, ni si quiera ubicar y clasificar alguno de la zona.

- **Modelo 2**

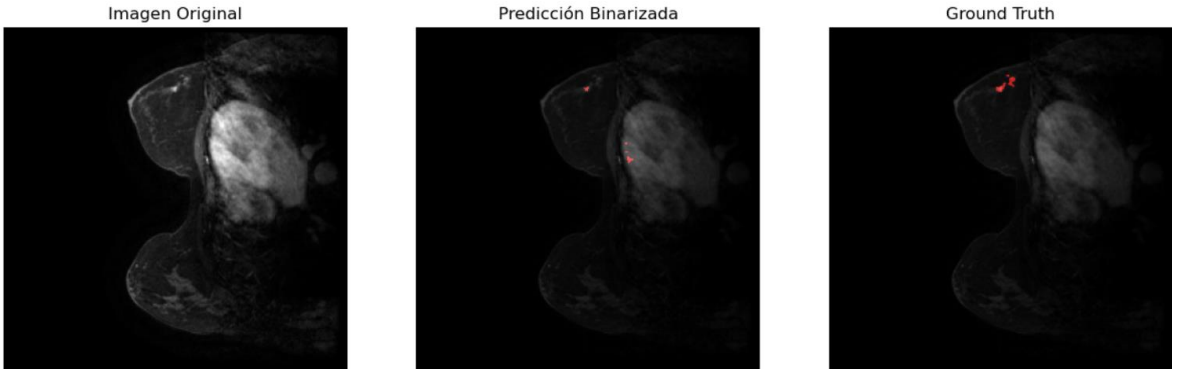
Caso 18



Caso 19



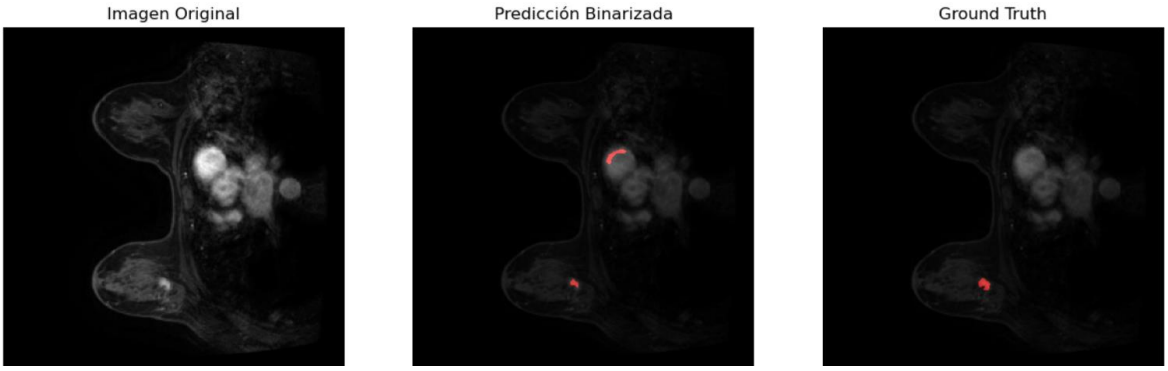
Caso 20



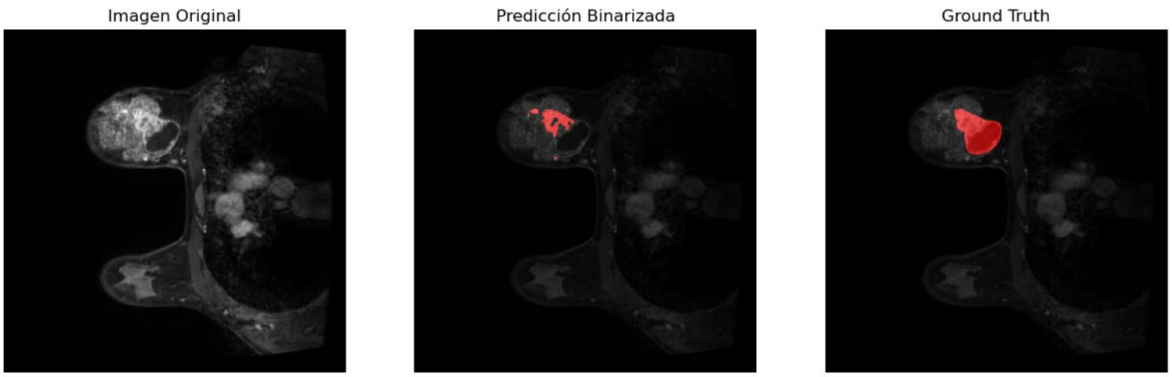
Este modelo esta más abierto en la clasificación teniendo algo más de tendencia a clasificar como tumor, ubicando correctamente los tumores sin llegar a perfilarlos, pero al tener más tendencia a clasificar como lesión acaba clasificando zonas erróneas. En el caso 18 no identifica la región mamaria, ya que no es capaz de generalizar para nuevos casos y clasifica como lesión cualquier píxel con una intensidad alta, siendo esto característico de una lesión.

- **Modelo 3**

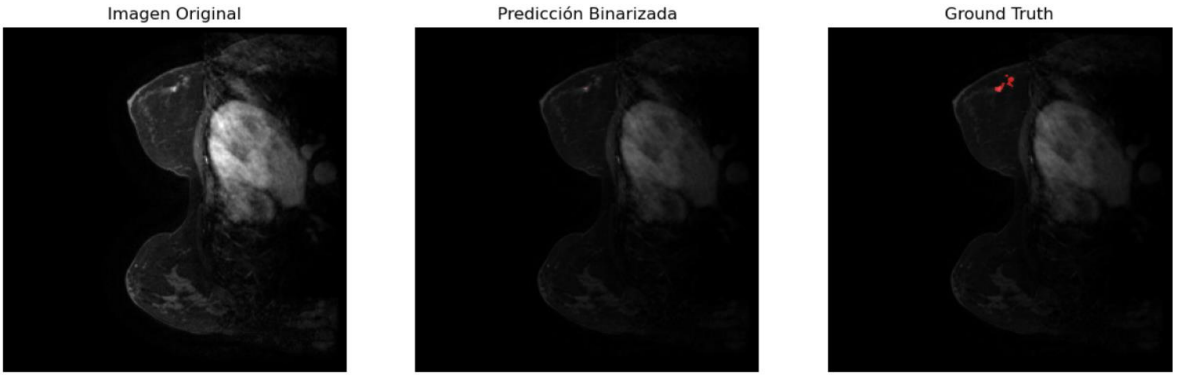
Caso 18



Caso 19



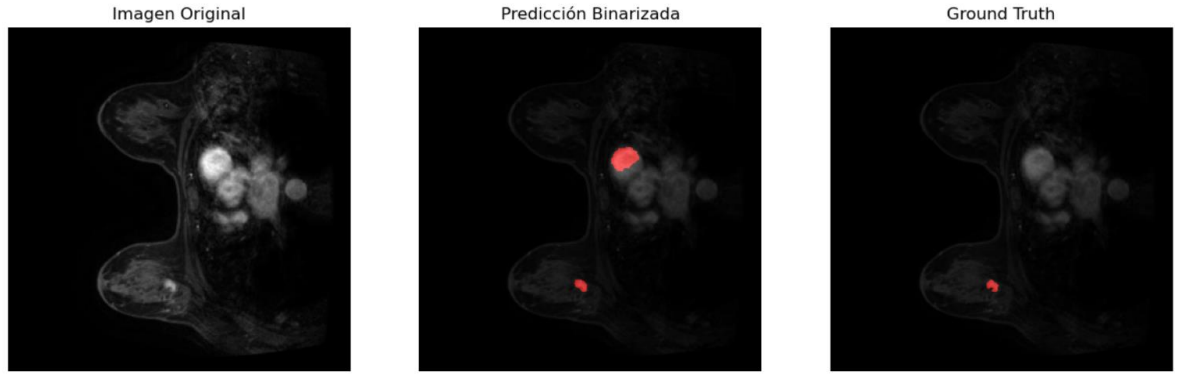
Caso 20



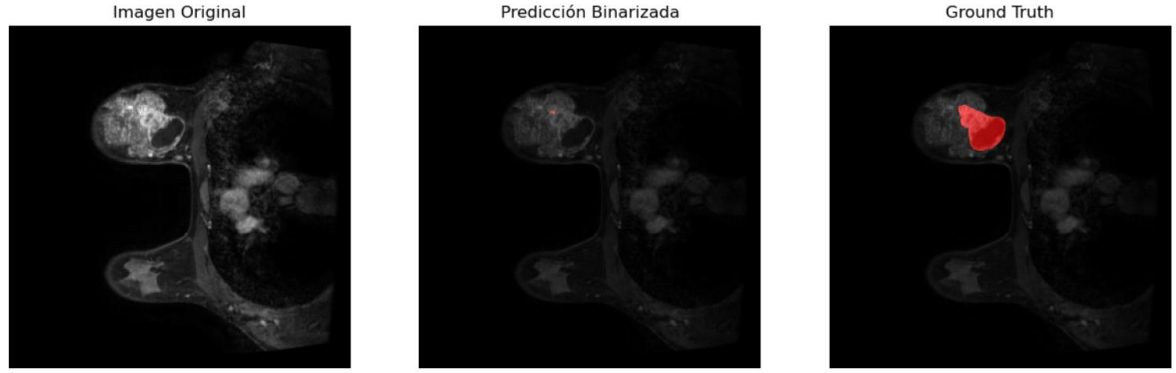
El tercer modelo tiene mejores predicciones que su anterior configuración, teniendo cierta precisión en la clasificación de una parte de la región anómala y equivocándose menos en la clasificación de pixeles que forman parte de otras estructuras anatómicas.

- **Modelo 4**

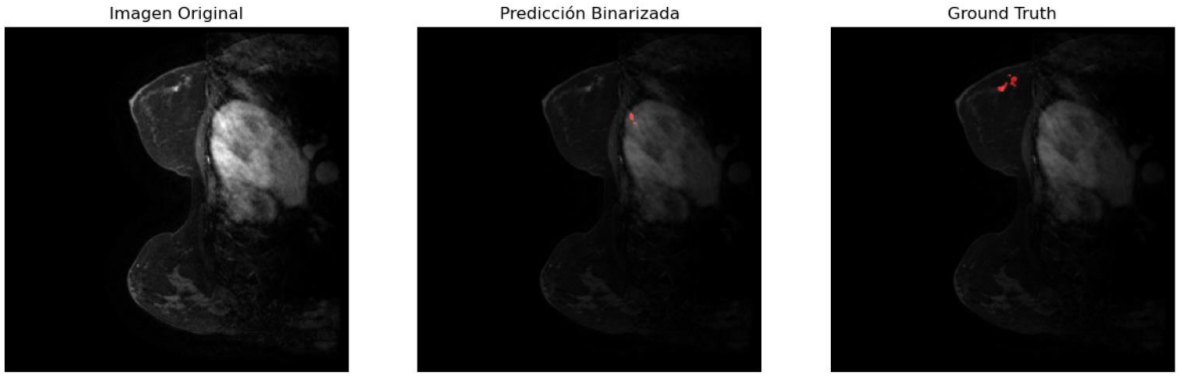
Caso 18



Caso 19



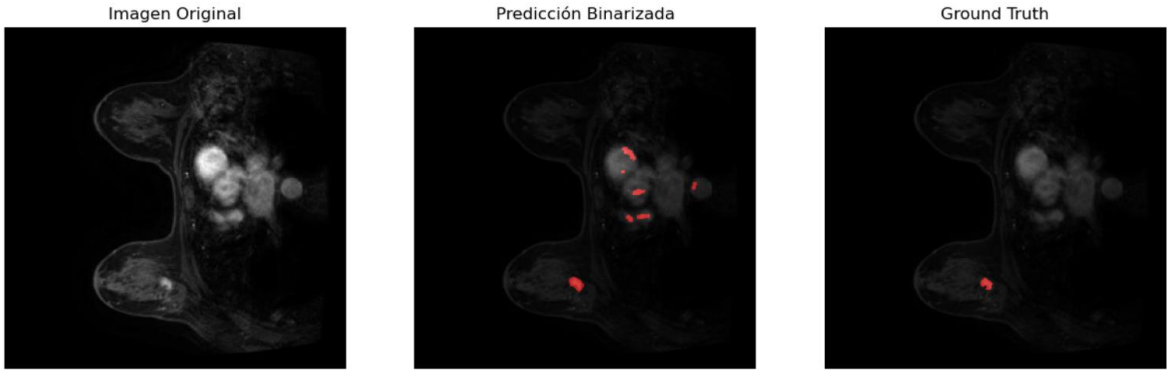
Caso 20



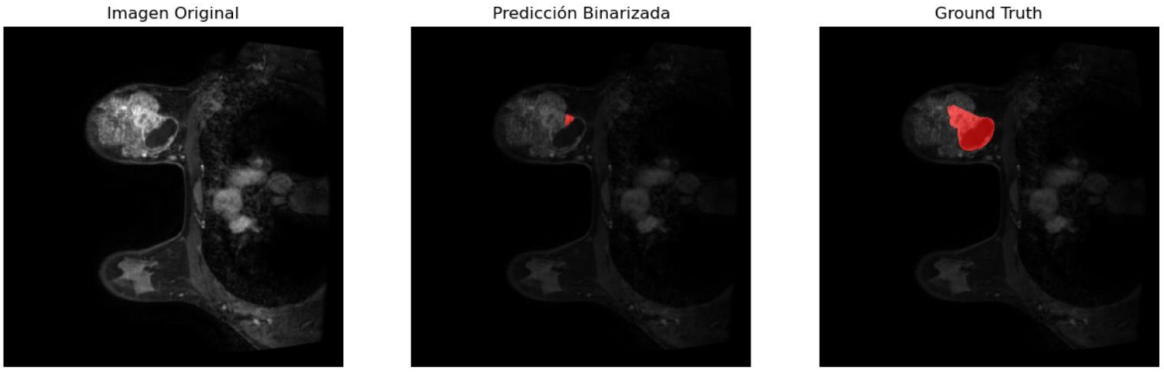
El cuarto modelo muestra unas predicciones bastante pobres siendo incapaz de ubicar las zonas o regiones donde aparece la lesión como se ve en el caso 19 y 20, al contrario que el caso 18 que sí que ha sido capaz de clasificar la lesión con bastante precisión, a cambio de clasificar zonas que ni si quiera pertenecen a la mama y mucho menos a una lesión.

- **Modelo 5**

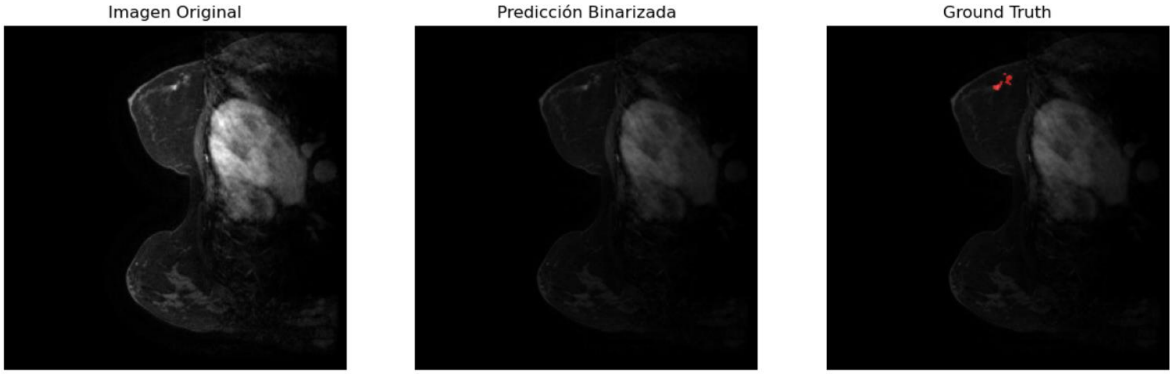
Caso 18



Caso 19



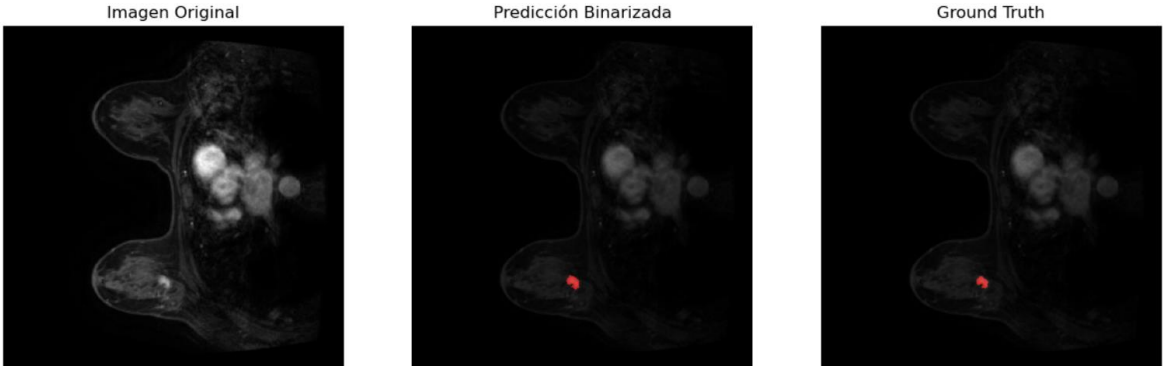
Caso 20



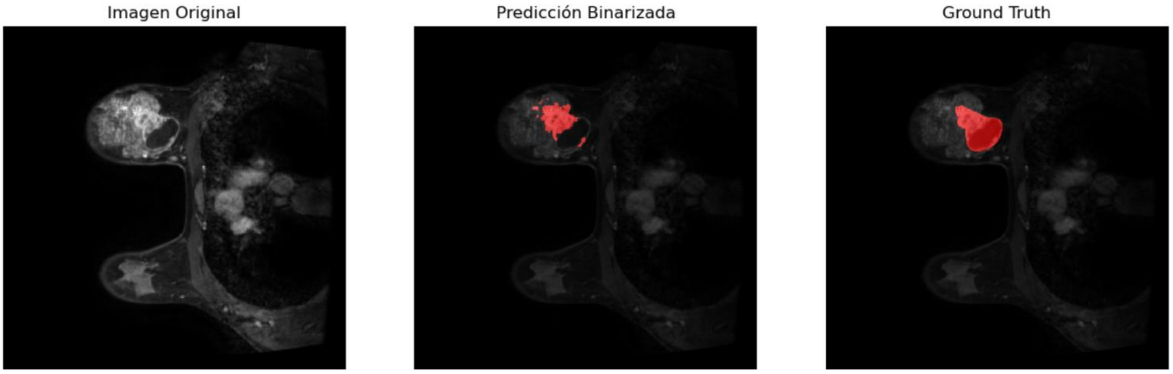
El quinto modelo es de los que peores predicciones muestra, clasificando cualquier píxel con una intensidad parecida a las que pertenecen las lesiones, sin poder generalizar para nuevos casos. En el caso 18 y 19 ubica ciertas regiones de la lesión de mama junto con otras estructuras anatómicas, mientras que el en caso 20 es incapaz de clasificar cualquier píxel que pertenezca a la lesión.

- **Modelo 6**

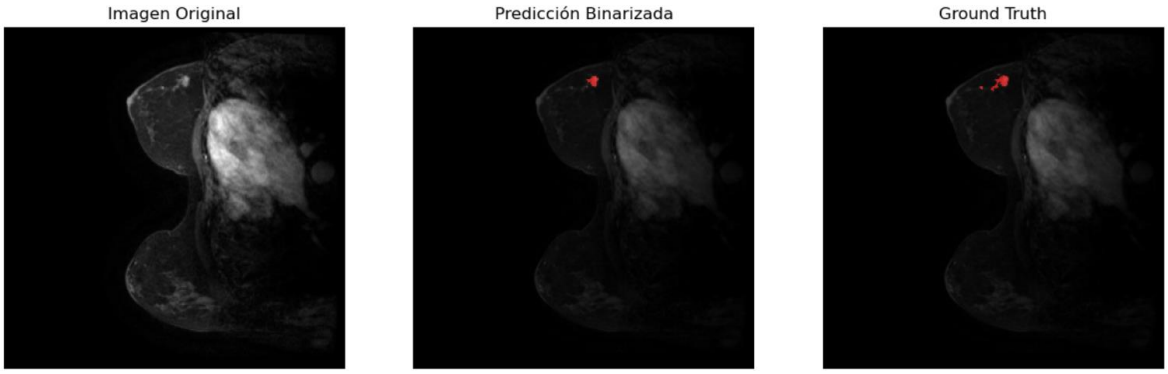
Caso 18



Caso 19



Caso 20



Este modelo es el que mejores métricas ha mostrado y se refleja en sus predicciones. Este modelo es capaz de clasificar muy bien las diferentes lesiones perfilándolas bastante bien y diferenciando las estructuras que no pertenecen a la región mamaria. Los tres casos han sido clasificados correctamente y se ha segmentado la mayor parte de la región lesionada.

6 Conclusiones

En esta investigación, hemos explorado la aplicación de redes neuronales para la segmentación de imágenes médicas, enfocándonos específicamente en la detección y segmentación de lesiones de mama. A través de una estructuración en cinco bloques, hemos abordado los componentes teóricos fundamentales, revisado diversas técnicas y estudios previos, detallando la metodología utilizada para la creación y entrenamiento de los modelos.

Los resultados obtenidos durante esta investigación, mediante el desarrollo de seis modelos, indican una evolución en el desempeño de la segmentación de nuestras muestras. Aunque los resultados no son excelentes, debemos considerar que el conjunto de datos utilizado es limitado, ya que el entrenamiento se ha realizado con imágenes de 14 pacientes, lo que representa 14 lesiones diferentes. Esta muestra no abarca la gran variedad de lesiones que pueden presentarse en el contexto de las lesiones de mama. Las lesiones de mama no siguen un patrón ni forma específicos, lo que dificulta que la red neuronal pueda captar todas las variaciones y características de las lesiones, dado que cada caso tiene particularidades únicas. Esto es más complejo que, por ejemplo, la segmentación de una estructura anatómica como la aorta, donde la mayoría de los pacientes tienen una misma localización, forma y pocas variaciones, facilitando así la captura de patrones y la generalización.

Estos resultados indican que, para lograr un buen desempeño en la segmentación de lesiones de cáncer de mama, es necesario proporcionar una mayor variedad de ejemplos para que la red pueda aprender de todas las variaciones posibles de las lesiones.

A pesar de estas limitaciones, el modelo 6 logró un 30,8% de coincidencia para la predicción nuevos casos, lo cual es un resultado aceptable dadas las condiciones del estudio. Este modelo mostró potencial para identificar la lesión con bastante precisión, aunque sin profundizar en la segmentación de los bordes. Este resultado puede ser útil para los radiólogos, permitiéndoles localizar y modificar la segmentación basada en una primera aproximación proporcionada por el modelo, agilizando así el proceso de diagnóstico al evitar que el radiólogo comience desde cero.

La continuación de este estudio debería incluir la ampliación de la capacidad del hardware para permitir más modificaciones y la adopción de técnicas que logren una mayor optimización del modelo. Además, sería beneficioso nutrir el modelo con más casos, posiblemente introduciendo nuevos casos para que sean segmentados por el modelo, supervisados por un radiólogo y luego utilizados para reentrenar la red. Este proceso de retroalimentación podría llevar a un crecimiento considerable del modelo, mejorando significativamente su desempeño en la segmentación de lesiones de mama.

Referencias

1. Página Web: Organización Mundial de la Salud. (s.f.). Cáncer de mama. Recuperado de https://www.who.int/es/news-room/fact-sheets/detail/breast-cancer [consulta] 15 de abril de 2024.

2. Página Web: Observatorio Global del Cáncer. (s.f.). Recuperado de https://gco.iarc.fr/en [consulta] 23 de mayo de 2024.

3. Libro: Bankman, I. N. (Ed.). (2009). Handbook of Medical Image Processing and Analysis (2a ed.). Academic Press.

4. Página Web: MedlinePlus. (s.f.). Cáncer de mama. Recuperado de https://medlineplus.gov/spanish/ency/article/000913.htm [consulta] 5 de junio de 2024.

5. Artículo de Revista: Lubinus Badillo, F., Rueda Hernández, C. A., Marconi Narváez, B., & Arias Trillos, Y. E. (2021). Redes neuronales convolucionales: un modelo de Deep Learning en imágenes diagnósticas. Revisión de tema. Revista Colombiana De Radiología, 32(3), 5591–5599. https://doi.org/10.53903/01212095.161

6. Trabajo de Fin de Grado: García González, E. (2018). Segmentación 3D de la Aorta. Trabajo de Fin de Grado, Universidad de Las Palmas de Gran Canaria.

7. Página Web: The Data Schools. (s.f.). ¿Qué es una red neuronal? Recuperado de https://thedataschools.com/que-es/red-neuronal/ [consulta] 10 de abril de 2024.

8. Artículo de Preimpresión: Cuevas-Tello, J. (2018). Apuntes de Redes Neuronales Artificiales. ArXiv, abs/1806.05298.

9. Video de YouTube: (s.f.). Recuperado de https://www.youtube.com/watch?v=CU24iC3grq8 [consulta] 28 de mayo de 2024.

10. Página Web: The Data Schools. (s.f.). ¿Qué es una red neuronal? Recuperado de https://thedataschools.com/que-es/red-neuronal/ [consulta] 15 de junio de 2024.

11. Lugar web personal: Barrios, J. (s.f.). Las Redes Neuronales Convolucionales. Recuperado de www.juanbarrios.com [consulta] 29 de marzo de 2024.

12. Página Web: IBM. (s.f.). Redes neuronales convolucionales. Recuperado de https://www.ibm.com/es-es/topics/convolutional-neural-networks#:~:text=Se%20compone%20de%20un%20cuadro,caracter%C3%ADsticas%20dentro%20de%20la%20CNN [consulta] 20 de abril de 2024.

13. Página Web: Aprende Machine Learning. (s.f.). Cómo funcionan las Convolutional Neural Networks: visión por ordenador. Recuperado de https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/ [consulta] 3 de mayo de 2024.

14. Página Web: Codificando Bits. (s.f.). Función de activación. Recuperado de https://www.codificandobits.com/blog/funcion-de-activacion/#la-funci%C3%B3n-relu [consulta] 12 de junio de 2024.

15. Página Web:Data Scientest. (s.f.). Convolutional Neural Network (CNN): todo lo que necesitas saber. Recuperado de https://datascientest.com/es/convolutional-neural-network-es [consulta] 8 de abril de 2024.

16. Página Web: Data Scientest. (s.f.). Convolutional Neural Network (CNN): todo lo que necesitas saber. Recuperado de https://datascientest.com/es/convolutional-neural-network-es [consulta] 21 de mayo de 2024.

17. Página Web: Aprende Machine Learning. (s.f.). Crear una red neuronal en Python desde cero. Recuperado de https://www.aprendemachinelearning.com/crear-una-red-neuronal-en-python-desde-cero/ [consulta] 30 de mayo de 2024.

18. Trabajo de Fin de Grado: Soto Sánchez, P. (2021). Diseño de una red neuronal convolucional para la segmentación de estructuras subcorticales cerebrales. Trabajo de Fin de Grado, Universidad Politécnica de Valencia.

19. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. En Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18* (pp. 234-241). Springer International Publishing.

20. Página Web: Data Scientest. (s.f.). U-Net: lo que tienes que saber. Recuperado de https://datascientest.com/es/u-net-lo-que-tienes-que-saber [consulta] 18 de mayo de 2024.

21. Informe Universitario: University of Colorado Boulder. (s.f.). Técnicas de regularización y métodos. Recuperado de https://home.cs.colorado.edu/~mozer/Teaching/syllabi/4576/materials/ Regularization.pdf [consulta] 7 de abril de 2024.

22. Informe Universitario: University of Wisconsin–Madison. (s.f.). Técnicas para la regularización en redes neuronales. Recuperado de https://pages.cs.wisc.edu/~dpage/cs760/lecture_notes/NN_part2.pdf [consulta] 2 de junio de 2024.

23. Proyecto GitHub: Revelo Luna, D. (s.f.). Semantic Segmentation Basic U-Net. Recuperado de https://github.com/DavidReveloLuna/Semantic-Segmentation-Basic-U-Net [consulta] 19 de mayo de 2024.

24. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60-88. https://arxiv.org/pdf/1803.04346.pdf

25. Rahman, M. A., & Wang, Y. (2016). Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. En International Symposium on Visual Computing (ISVC). https://arxiv.org/pdf/1901.07839.pdf

26. Salehi, S. S. M., Erdogmus, D., & Gholipour, A. (2017). Tversky loss function for image segmentation using 3D fully convolutional deep networks. ArXiv. https://arxiv.org/pdf/1706.05721.pdf

27. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

28. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

29. Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

30. Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann.

31. Parisi, L., Neagu, D., RaviChandran, N., & Ma, R. (2024). Optimal Evolutionary Framework-based Activation Function for Image Classification. Knowledge-Based Systems. https://www.sciencedirect.com/science/article/pii/S0950705124006592

32. Khaled, R., Vidal, J., Vilanova, J. C., & Martí, R. (2022). Un conjunto de U-Net para la segmentación de lesiones mamarias en DCE MRI. Computadoras en Biología y Medicina. Disponible en: https://www.sciencedirect.com/science/article/pii/S0010482524000093

33. Hossain, S., Azam, S., Montaha, S., Karim, A., Chowa, S. S., Mondol, C., Hasan, M. Z., & Jonkman, M. (2023). Segmentación automatizada de imágenes de ultrasonido de tumores de mama con UNet híbrido y clasificación utilizando un modelo CNN ajustado. Heliyón. Recuperado de https://www.cell.com/heliyo