




14 DE MAYO DE 2024

ANÁLISIS DE DATOS NO ESTRUCTURADOS

SEGMENTADOR DE IMÁGENES DE CORAZÓN

ALBERTO SÁNCHEZ BONASTRE
LAWRENCE JAVIER MINGUILLÁN VAN KAPEL



Índice

1.	<u>Introducción y objetivo.....</u>	<u>2</u>
2.	<u>Explicación Dataset.....</u>	<u>2</u>
3.	<u>Preprocesamiento de la imagen.....</u>	<u>4</u>
4.	<u>Definición UNET desde scratch.....</u>	<u>4</u>
5.	<u>Evaluación de resultados.....</u>	<u>6</u>
6.	<u>Refinamiento de modelo.....</u>	<u>9</u>
7.	<u>Evaluación de resultados finales.....</u>	<u>12</u>
8.	<u>Conclusiones.....</u>	<u>15</u>

Introducción y objetivos

El diagnóstico y tratamiento precisos de malformaciones cardíacas, particularmente en pacientes con síndrome de Down, requieren un análisis minucioso de las estructuras del corazón. Una de las áreas más desafiantes en la imagenología médica es la segmentación eficaz de estas estructuras complejas, especialmente la válvula mitral. Este proyecto se centra en el desarrollo de un modelo robusto de segmentación automática que utiliza técnicas avanzadas de aprendizaje profundo para mejorar la interpretación de imágenes cardíacas.

El objetivo principal del proyecto es construir un modelo de segmentación basado en la arquitectura UNET, comenzando desde cero y reforzándolo con un modelo pre-entrenado de cerebros para mejorar su precisión y robustez. Los objetivos específicos incluyen:

- **Desarrollo de un Modelo de Segmentación Robusto:** Crear un modelo de red UNET desde scratch, destinado a la segmentación precisa de las estructuras cardíacas en imágenes médicas.
- **Refinamiento Utilizando un Modelo Pre-entrenado:** Incorporar un modelo pre-entrenado de segmentación de cerebros para transferir el aprendizaje y mejorar la precisión del modelo, adaptándolo a las características únicas de las imágenes cardíacas.
- **Validación del Modelo:** Evaluar la efectividad del modelo en segmentar con precisión las estructuras cardíacas.

De cara a establecer pasos futuros, tras la validación del modelo, se procederá a su aplicación en contextos clínicos, enfocándose en dos principales áreas de implementación:

- **Extracción de la ROI de la Válvula Mitral:** Aplicar el modelo refinado para segmentar y extraer la Región de Interés de la válvula mitral. Este paso es crucial para permitir un análisis detallado, que es fundamental para el tratamiento efectivo de las malformaciones cardíacas.
- **Integración Clínica:** Facilitar la integración del modelo en la práctica clínica, proporcionando una herramienta valiosa para el diagnóstico y la planificación terapéutica. También se incluirá la capacitación del personal médico en el uso eficiente de esta tecnología avanzada.

Este enfoque estratégico no solo impulsa el campo de la imagenología médica mediante el uso de tecnologías de aprendizaje profundo, sino que también mejora significativamente la calidad de vida de los pacientes con condiciones cardíacas complejas, optimizando los procesos de diagnóstico y tratamiento.

Explicación Dataset

El dataset utilizado para el proyecto de segmentación automática del corazón se compone de imágenes obtenidas a través de Tomografía Axial Computarizada (TAC) de un paciente. Las imágenes originales, en formato DICOM, característico de los estudios médicos de imagen, han sido convertidas a formato PNG para facilitar su manejo y procesamiento en las etapas de entrenamiento y prueba del modelo de red neuronal. Esta conversión ayuda a estandarizar el dataset y simplifica la integración con la arquitectura de la red UNET.

Para la creación del dataset de entrenamiento y prueba, se realizaron segmentaciones manuales de las estructuras cardíacas en las imágenes. Cada estructura relevante del corazón fue delineada y coloreada con diferentes colores para representar distintas regiones del corazón. Esta segmentación manual es crucial para entrenar el modelo en la identificación precisa de estas estructuras en imágenes no segmentadas. De esta manera se representa:

- Blanco: aorta ascendente AA
- Rojo: aurícula izquierda AI
- Azul: ventrículo izquierdo VI
- Amarillo: aurícula derecha AD
- Verde: ventrículo derecho VD
- Morado: aorta descendente AD
- Cyan: arteria pulmonar AP
- Naranja: vena cava VC

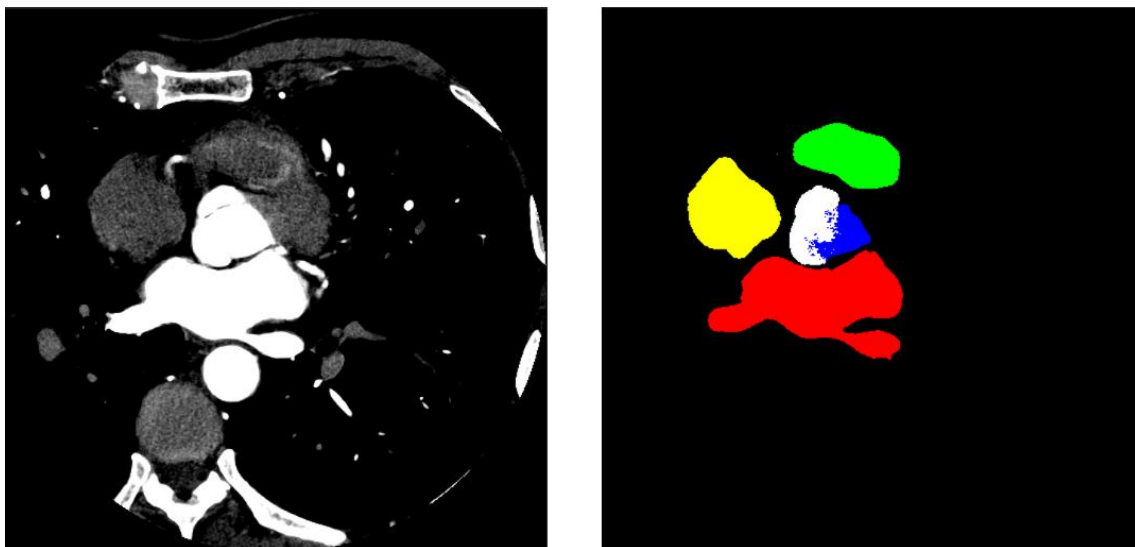


Figura 1. Imagen de corazón con su correspondiente segmentación manual

El dataset sin data augmentation consta de 640 imágenes para el entrenamiento, cada una acompañada de su correspondiente imagen segmentada manualmente. Además, se dispone de 320 imágenes destinadas a la fase de prueba, también con sus respectivas imágenes segmentadas.

Aunque el proceso de data augmentation se llevó a cabo para aumentar la diversidad del dataset y prevenir el sobreajuste, pasando de 640 imágenes a 6120, debido a restricciones

de tiempo de procesamiento, se optó por utilizar la versión original del dataset para las fases de entrenamiento y pruebas. La técnica de data augmentation empleada incluye rotaciones horizontales y verticales, borrosidad, todas las transformaciones a la vez y cada una de ellas con un peso del 25% de que ocurran, todo esto para generar variantes de las imágenes que refuercen la capacidad del modelo de generalizar a partir de nuevos datos.

Preprocesamiento de la imagen

A continuación, se detallan los pasos seguidos en el preprocesamiento de las imágenes de tomografía axial computarizada (TAC) utilizadas para la segmentación del corazón:

- Organización de Imágenes:

Las imágenes están divididas en dos grupos: imágenes de entrenamiento (input y output) e imágenes de prueba (solo input).

- Normalización de Dimensiones:

Todas las imágenes se ajustan a las dimensiones de la imagen más grande dentro del dataset. Para las imágenes más pequeñas, se añaden píxeles negros para asegurar la uniformidad en el tamaño, facilitando el procesamiento por la red.

- Conversión a RGB:

Aunque las imágenes de TAC originalmente son en escala de grises, se transforman a formato RGB. Esto es para estandarizar el tratamiento de las imágenes y prepararlas para la segmentación de múltiples clases.

- Data Augmentation (código implementado, pero no empleado)
- Renombrado Secuencial:

Para simplificar el manejo y el acceso a las imágenes, se renombran siguiendo un orden secuencial. Esto facilita la automatización de procesos y la carga de datos durante el entrenamiento.

- Preprocesamiento de Máscaras:

Las máscaras segmentadas manualmente, que están en formato a color, se convierten a etiquetas de clase. Cada color único se mapea a una etiqueta específica, por ejemplo, blanco (255,255,255) para la clase 1 y rojo (255,0,0) para la clase 2.

Definición UNET desde scratch

A continuación, se detalla el funcionamiento de la arquitectura U-Net propuesta para el uso de imágenes de un corazón de 512x512 píxeles y su correspondiente máscara con 8 clases segmentando los elementos del corazón.

- Paso 1: Entrada

Imagen de Entrada: Tenemos una imagen de un corazón de 512x512 píxeles. Esta imagen es procesada por la capa de entrada de la red U-Net.

- Paso 2: Downsampling o Codificación

Durante esta fase, la red comienza a entender y codificar el contexto de la imagen (en este caso, las diferentes partes y características del corazón) reduciendo progresivamente su resolución espacial, pero aumentando la profundidad (cantidad de filtros), lo que nos permite capturar características más abstractas y globales.

1. Primera Capa Conv2D y MaxPooling: Las primeras capas convolucionales aplican 64 filtros para detectar características básicas como bordes o curvas. Después del pooling, la imagen tiene una resolución de 256x256 píxeles. (MaxPooling(2,2) --> de 512 a 256)

2. Segunda Capa Conv2D y MaxPooling: Esta fase utiliza 128 filtros y después del pooling, la imagen se reduce a 128x128 píxeles (MaxPooling(2,2) --> de 256 a 128). Aquí, la red empieza a reconocer patrones más complejos, como la forma del corazón y sus grandes componentes.

- Paso 3: Bottleneck

Esta es la fase más profunda, donde la red tiene la capacidad de aprender las características más críticas y detalladas del corazón, aún con una resolución reducida (128x128 píxeles después del último MaxPooling). Aquí, con 256 filtros, la red puede identificar las diferencias sutiles entre las diversas partes del corazón que son cruciales para una segmentación precisa.

- Paso 4: Upsampling o Decodificación

Ahora, la red comienza a reconstruir la imagen desde las características codificadas, aumentando su resolución mientras fusiona la información aprendida durante el downsampling para asegurar que los detalles finos no se pierdan.

1. Primera Capa de Upsampling: La imagen se aumenta de 128x128 a 256x256 píxeles (con Conv2DTranspose(2,2) --> de 128 a 256). Se concatenan las características aprendidas durante el downsampling correspondientes para mantener los detalles.

2. Segunda Capa de Upsampling: Continúa el proceso hasta restaurar la imagen a su tamaño original de 512x512 píxeles (con Conv2DTranspose(2,2) --> de 256 a 512). En este punto, la red ha aprendido a distinguir entre las diferentes partes del corazón.

- Paso 5: Salida

Capa de Salida: Finalmente, una capa convolucional ajusta el número de canales de salida al número de clases (8 en este caso). La activación softmax asegura que cada píxel de la imagen de salida representa la probabilidad de pertenecer a cada una de las 8 clases

segmentando los diferentes elementos del corazón (por ejemplo, las cámaras del corazón, las válvulas, los grandes vasos, etc.).

- Resultado

La imagen de salida será una segmentación semántica del corazón, donde cada píxel está clasificado en una de las 8 clases predefinidas, proporcionando una máscara detallada que identifica las distintas partes del corazón. Esto es especialmente útil en aplicaciones médicas, donde tales segmentaciones precisas pueden ayudar en el diagnóstico y tratamiento de condiciones cardíacas.

Evaluación de resultados

Tras realizar varias pruebas para optimizar la relación entre el tiempo de ejecución y la eficacia del entrenamiento de nuestro modelo de segmentación cardíaca basado en la red U-Net, hemos determinado que el modelo entrenado durante 75 epochs presenta el equilibrio más óptimo.

Los resultados han sido los siguientes:

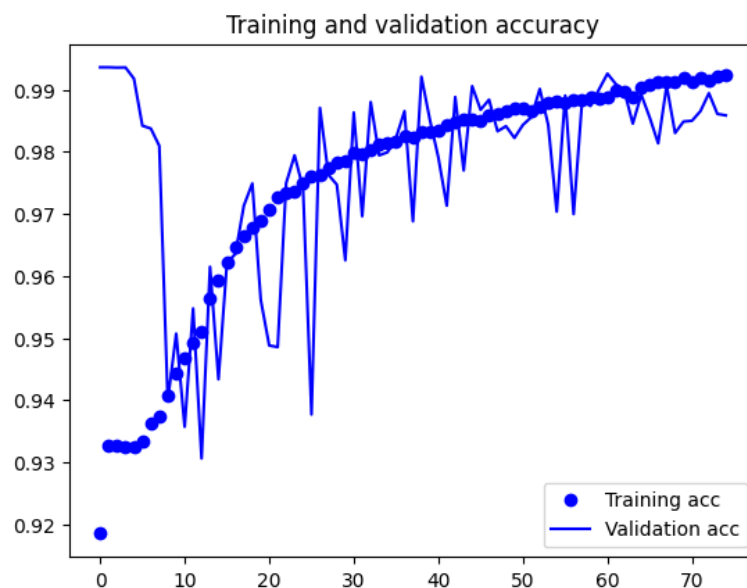


Figura 2. Curva training y validation accuracy (ejes ampliados).

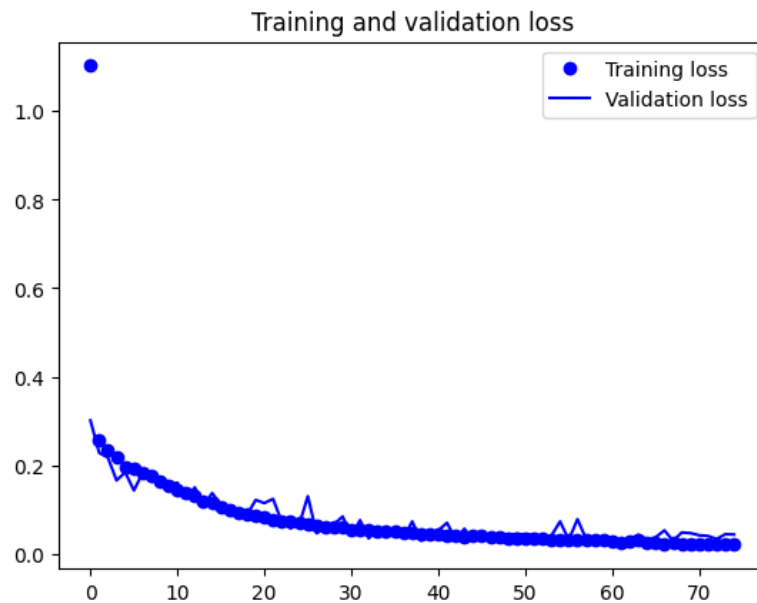


Figura 3. Curva training y validation loss.

Estas dos gráficas que presentaste muestran la evolución de la precisión y la pérdida durante el entrenamiento y validación de nuestro modelo de red neuronal UNET. Atendiendo a la figura 2, la precisión aumenta rápidamente al inicio y luego se estabiliza. La precisión de validación sigue de cerca a la de entrenamiento, indicando que el modelo generaliza bien y no muestra sobreajuste significativo. Y la figura 3, nos muestra como la pérdida disminuye consistentemente, indicando que el modelo está aprendiendo adecuadamente. La pérdida de validación disminuye en paralelo, con ligeras fluctuaciones, lo que sugiere estabilidad y buena generalización.

Si observamos algunos ejemplos de predicción junto con las máscaras reales para poder comprobar visualmente los resultados de esta red.

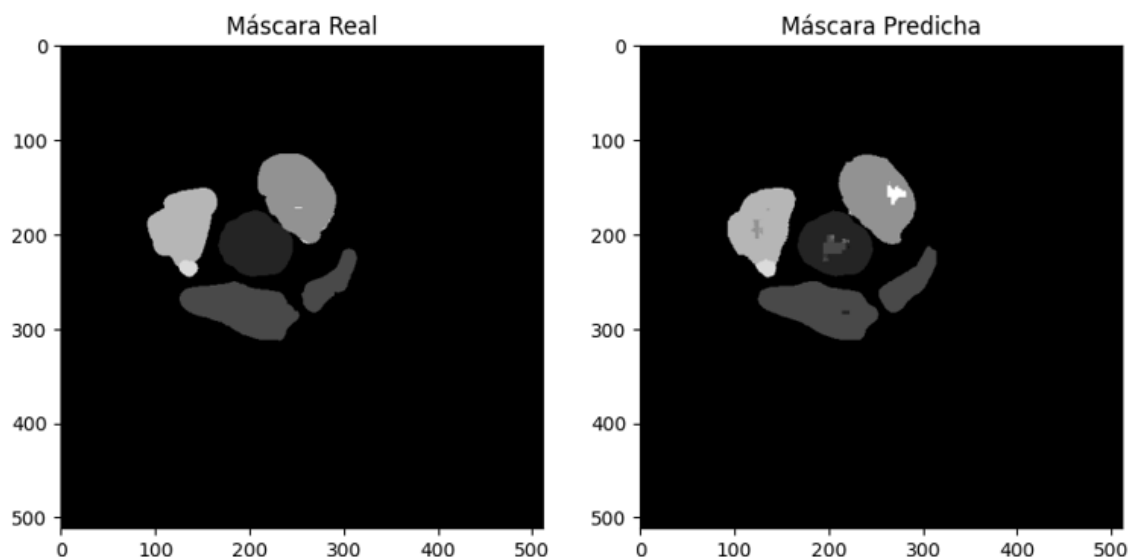


Figura 4. Predicción visual 1.

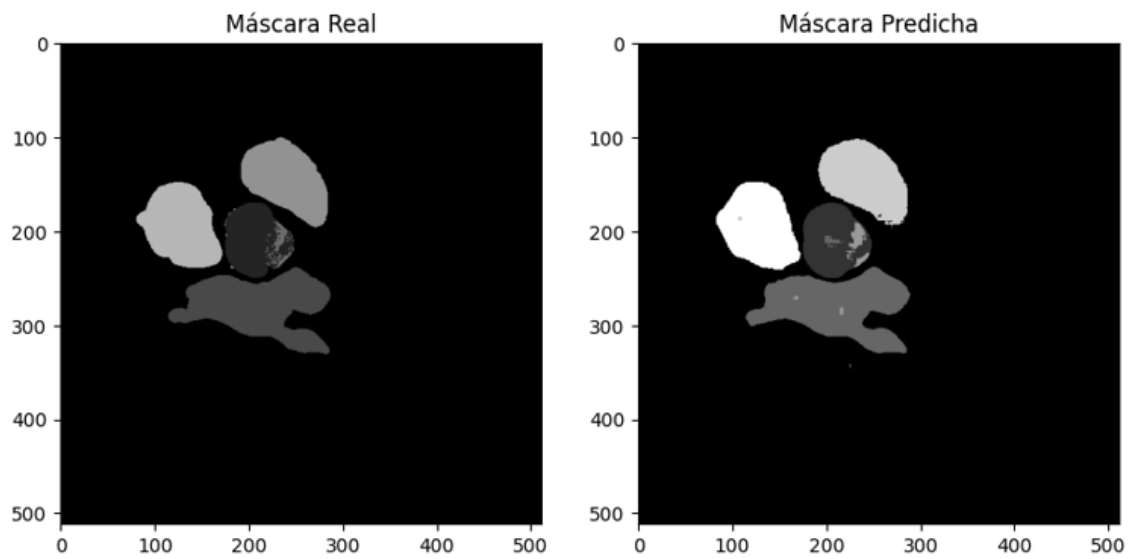


Figura 5. Predicción visual 2.

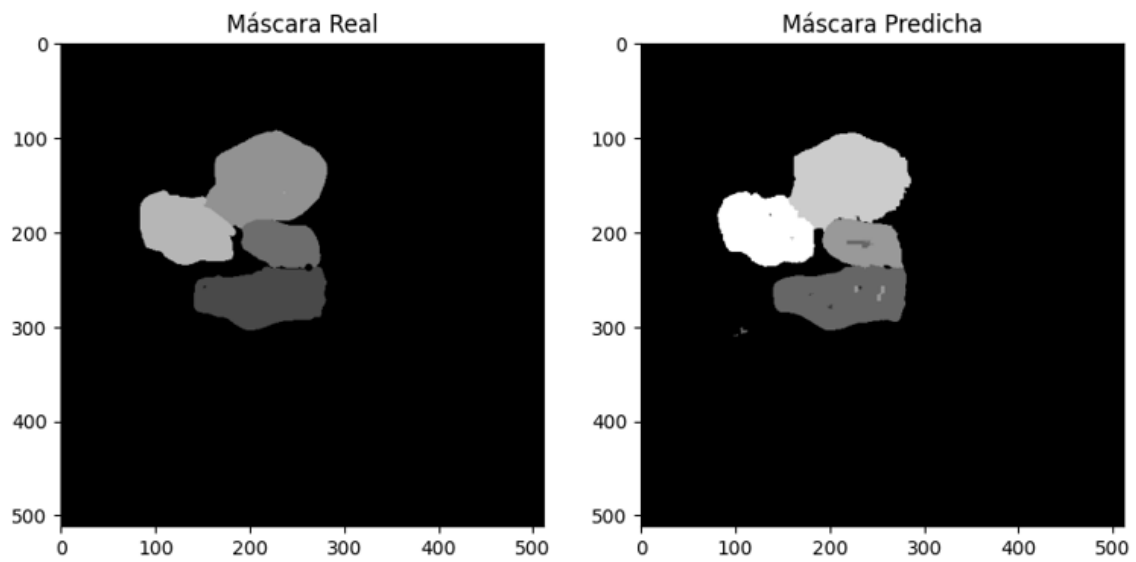


Figura 6. Predicción visual 3.

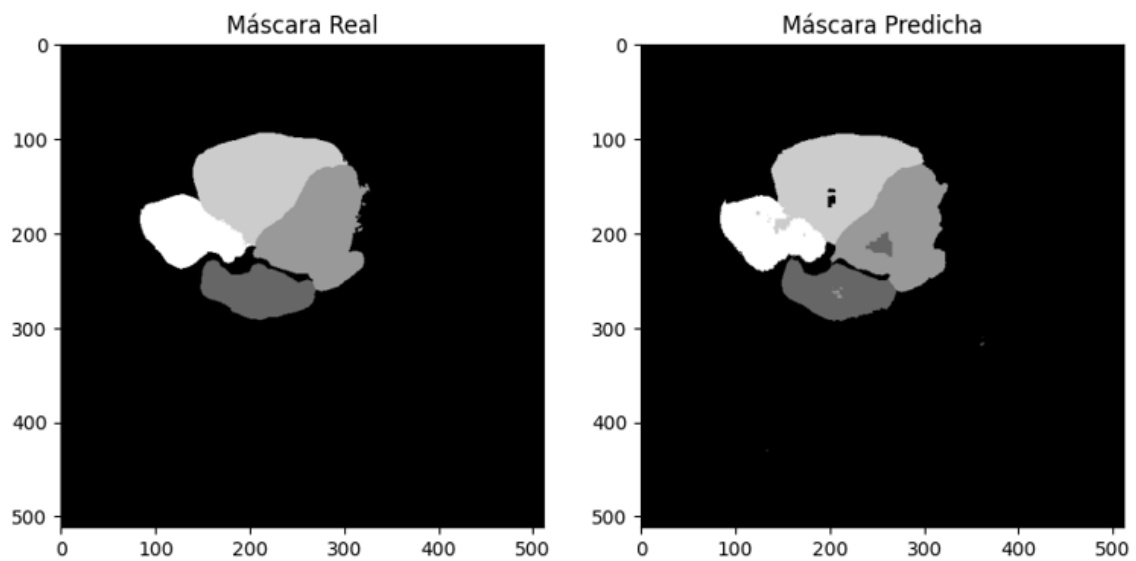


Figura 7. Predicción visual 4.

Como se observa en las anteriores predicciones, el modelo actual ofrece resultados aparentemente correctos. Sin embargo, con el objetivo de mejorar aún más estas predicciones, vamos a utilizar la parte convolucional de un modelo pre-entrenado de cerebro. Este enfoque comenzará con la feature extraction, aprovechando las capacidades ya aprendidas del modelo pre-entrenado. Posteriormente, se procederá al descongelamiento progresivo de los pesos para permitir un fine tuning en las capas específicas. Esto permitirá al modelo adaptarse más efectivamente a las particularidades de nuestro dataset de imágenes cardíacas, buscando optimizar aún más la precisión de la segmentación.

Refinamiento del modelo

Para este proceso de feature extraction y fine tuning, se utiliza un modelo pre-entrenado de cerebros de hugging face. La idea es utilizar su parte convolucional, y con sus pesos congelados para que al enseñar nuestro dataset de corazones aprenda nuevas features de este. Una vez hecho eso, puedo descongelar capas de la parte convolucional para ir viendo la mejora (fine tuning).

Con un summary del modelo pre-entrenado podemos ver la estructura de este y así identificar su última capa convolucional.

conv2d_7 (Conv2D)	(None, 32, 32, 512)	255360	['activation_6[0][0]']
batch_normalization_7 (Batch Normalization)	(None, 32, 32, 512)	2048	['conv2d_7[0][0]']
activation_7 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_7[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 512)	0	['activation_7[0][0]']
conv2d_8 (Conv2D)	(None, 16, 16, 1024)	4719616	['max_pooling2d_3[0][0]']
batch_normalization_8 (Batch Normalization)	(None, 16, 16, 1024)	4096	['conv2d_8[0][0]']
activation_8 (Activation)	(None, 16, 16, 1024)	0	['batch_normalization_8[0][0]']
conv2d_9 (Conv2D)	(None, 16, 16, 1024)	9438208	['activation_8[0][0]']
batch_normalization_9 (Batch Normalization)	(None, 16, 16, 1024)	4096	['conv2d_9[0][0]']
activation_9 (Activation)	(None, 16, 16, 1024)	0	['batch_normalization_9[0][0]']
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 512)	2097664	['activation_9[0][0]']
concatenate (Concatenate)	(None, 32, 32, 1024)	0	['conv2d_transpose[0][0]', 'activation_7[0][0]']
conv2d_10 (Conv2D)	(None, 32, 32, 512)	4719104	['concatenate[0][0]']
batch_normalization_10 (Batch Normalization)	(None, 32, 32, 512)	2048	['conv2d_10[0][0]']

Figura 8. Estructura del modelo pre-entrenado. (Únicamente se está mostrando lo correspondiente a la última capa convolucional).

Vemos como la última capa convolucional del modelo (usualmente antes de cualquier capa Flatten o Dense) en la cual realizar feature extraction es la capa conv2d_9 (None, 16, 16, 1024). Con lo cual cortamos el modelo pre-entrenado hasta la capa específica.

De cara a hacer feature extraction tenemos que convertir nuestras imágenes a (256, 256, 3) ya que como podemos ver en la siguiente imagen son las dimensiones de entrada al modelo pre-entrenado.

Model: "UNET"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[None, 256, 256, 3]	0	[]
conv2d (Conv2D)	(None, 256, 256, 64)	1792	['input_1[0][0]']
batch_normalization (Batch Normalization)	(None, 256, 256, 64)	256	['conv2d[0][0]']
activation (Activation)	(None, 256, 256, 64)	0	['batch_normalization[0][0]']
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36928	['activation[0][0]']

Figura 9. Dimensiones de entrada del modelo pre-entrenado.

Una vez realizada la feature extraction, vamos a crear un nuevo modelo que contenga la parte convolucional del modelo pre-entrenado y la parte deconvolucional del nuestro.

El modelo final que obtenemos es el siguiente:

Model: "model_2"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[None, 256, 256, 3]	0	[]
conv2d (Conv2D)	(None, 256, 256, 64)	1792	['input_1[0][0]']
batch_normalization (Batch Normalization)	(None, 256, 256, 64)	256	['conv2d[0][0]']
activation (Activation)	(None, 256, 256, 64)	0	['batch_normalization[0][0]']
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36928	['activation[0][0]']
batch_normalization_1 (Batch Normalization)	(None, 256, 256, 64)	256	['conv2d_1[0][0]']
activation_1 (Activation)	(None, 256, 256, 64)	0	['batch_normalization_1[0][0]']
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	['activation_1[0][0]']
conv2d_2 (Conv2D)	(None, 128, 128, 128)	73856	['max_pooling2d[0][0]']
batch_normalization_2 (Batch Normalization)	(None, 128, 128, 128)	512	['conv2d_2[0][0]']
activation_2 (Activation)	(None, 128, 128, 128)	0	['batch_normalization_2[0][0]']
conv2d_3 (Conv2D)	(None, 128, 128, 128)	147584	['activation_2[0][0]']

batch_normalization_3 (Batch Normalization)	(None, 128, 128, 128)	512	['conv2d_3[0][0]']
activation_3 (Activation)	(None, 128, 128, 128)	0	['batch_normalization_3[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 128)	0	['activation_3[0][0]']
conv2d_4 (Conv2D)	(None, 64, 64, 256)	295168	['max_pooling2d_1[0][0]']
batch_normalization_4 (Batch Normalization)	(None, 64, 64, 256)	1024	['conv2d_4[0][0]']
activation_4 (Activation)	(None, 64, 64, 256)	0	['batch_normalization_4[0][0]']
conv2d_5 (Conv2D)	(None, 64, 64, 256)	590080	['activation_4[0][0]']
batch_normalization_5 (Batch Normalization)	(None, 64, 64, 256)	1024	['conv2d_5[0][0]']
activation_5 (Activation)	(None, 64, 64, 256)	0	['batch_normalization_5[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 256)	0	['activation_5[0][0]']
conv2d_6 (Conv2D)	(None, 32, 32, 512)	1180160	['max_pooling2d_2[0][0]']
batch_normalization_6 (Batch Normalization)	(None, 32, 32, 512)	2048	['conv2d_6[0][0]']
activation_6 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_6[0][0]']
conv2d_7 (Conv2D)	(None, 32, 32, 512)	2359808	['activation_6[0][0]']
batch_normalization_7 (Batch Normalization)	(None, 32, 32, 512)	2048	['conv2d_7[0][0]']
activation_7 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_7[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 512)	0	['activation_7[0][0]']
conv2d_8 (Conv2D)	(None, 16, 16, 1024)	4719616	['max_pooling2d_3[0][0]']
batch_normalization_8 (Batch Normalization)	(None, 16, 16, 1024)	4096	['conv2d_8[0][0]']
activation_8 (Activation)	(None, 16, 16, 1024)	0	['batch_normalization_8[0][0]']
conv2d_9 (Conv2D)	(None, 16, 16, 1024)	9438208	['activation_8[0][0]']
batch_normalization_9 (Batch Normalization)	(None, 16, 16, 1024)	4096	['conv2d_9[0][0]']
activation_9 (Activation)	(None, 16, 16, 1024)	0	['batch_normalization_9[0][0]']
conv2d_transpose_2 (Conv2D Transpose)	(None, 32, 32, 512)	2097664	['activation_9[0][0]']

concatenate_2 (Concatenate)	(None, 32, 32, 1024)	0	['conv2d_transpose_2[0][0]', 'activation_7[0][0]']
conv2d_11 (Conv2D)	(None, 32, 32, 512)	4719104	['concatenate_2[0][0]']
dropout_5 (Dropout)	(None, 32, 32, 512)	0	['conv2d_11[0][0]']
conv2d_12 (Conv2D)	(None, 32, 32, 512)	2359808	['dropout_5[0][0]']
conv2d_transpose_3 (Conv2D Transpose)	(None, 64, 64, 256)	524544	['conv2d_12[0][0]']
concatenate_3 (Concatenate)	(None, 64, 64, 512)	0	['conv2d_transpose_3[0][0]', 'activation_5[0][0]']
conv2d_13 (Conv2D)	(None, 64, 64, 256)	1179904	['concatenate_3[0][0]']
dropout_6 (Dropout)	(None, 64, 64, 256)	0	['conv2d_13[0][0]']
conv2d_14 (Conv2D)	(None, 64, 64, 256)	590080	['dropout_6[0][0]']
conv2d_transpose_4 (Conv2D Transpose)	(None, 128, 128, 128)	131200	['conv2d_14[0][0]']
concatenate_4 (Concatenate)	(None, 128, 128, 256)	0	['conv2d_transpose_4[0][0]', 'activation_3[0][0]']
conv2d_15 (Conv2D)	(None, 128, 128, 128)	295040	['concatenate_4[0][0]']
dropout_7 (Dropout)	(None, 128, 128, 128)	0	['conv2d_15[0][0]']
conv2d_16 (Conv2D)	(None, 128, 128, 128)	147584	['dropout_7[0][0]']
conv2d_transpose_5 (Conv2D Transpose)	(None, 256, 256, 64)	32832	['conv2d_16[0][0]']
concatenate_5 (Concatenate)	(None, 256, 256, 128)	0	['conv2d_transpose_5[0][0]', 'activation_1[0][0]']
conv2d_17 (Conv2D)	(None, 256, 256, 64)	73792	['concatenate_5[0][0]']
dropout_8 (Dropout)	(None, 256, 256, 64)	0	['conv2d_17[0][0]']
conv2d_18 (Conv2D)	(None, 256, 256, 64)	36928	['dropout_8[0][0]']
conv2d_19 (Conv2D)	(None, 256, 256, 8)	520	['conv2d_18[0][0]']
=====			
Total params: 31048072 (118.44 MB)			
Trainable params: 31040136 (118.41 MB)			
Non-trainable params: 7936 (31.00 KB)			

Figura 10. Summary del modelo final.

Ahora transformamos nuestro vector flatten de características y nuestras máscaras a dimensiones (256, 256, 3) y ya podríamos entrenar este nuevo modelo final.

Evaluación de resultados finales

Este modelo final ha sido entrenado con 50 epochs y este ha sido el resultado:

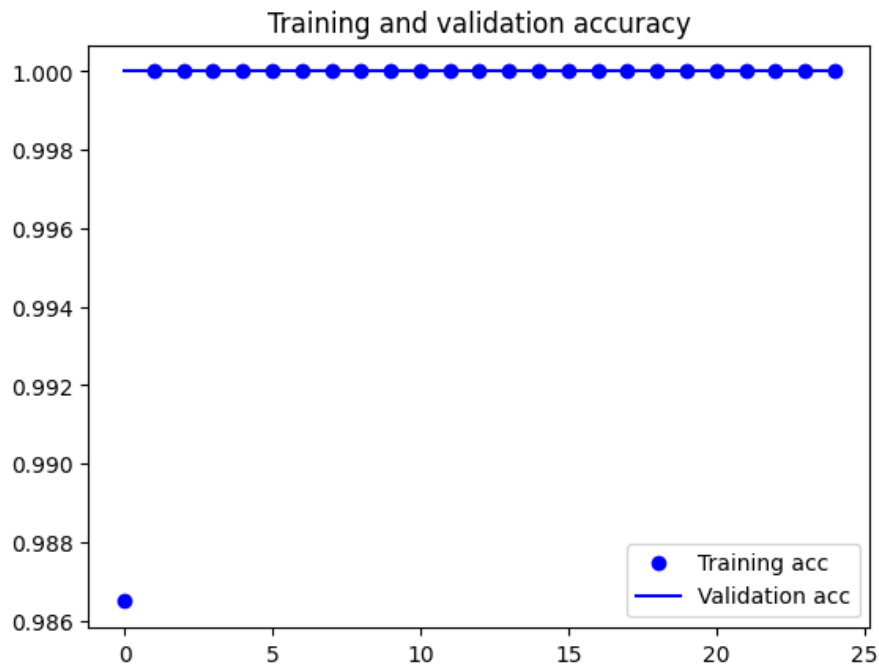


Figura 11. Training y validation accuracy para el modelo final.

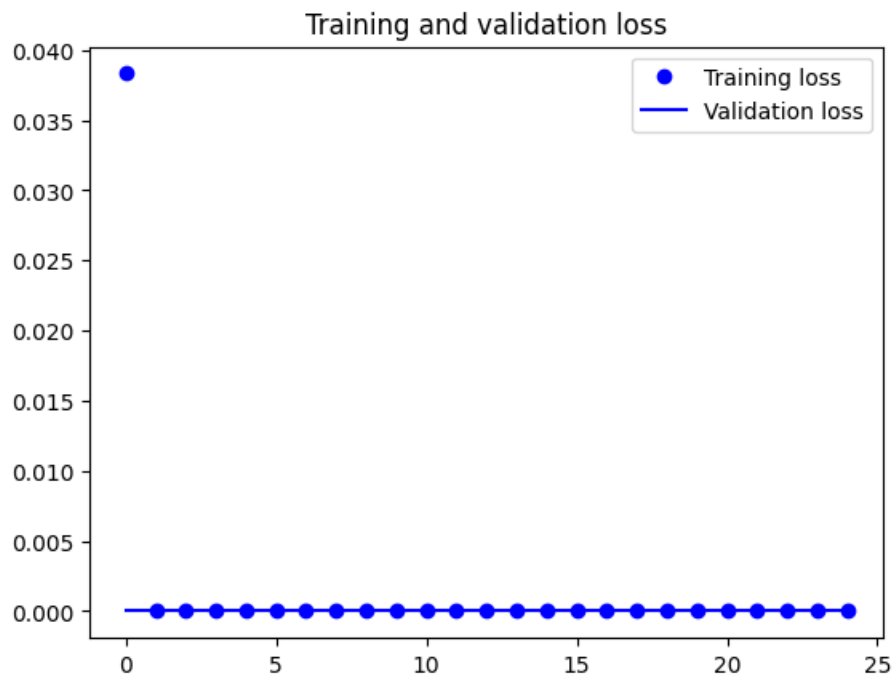


Figura 12. Training y validation los para el modelo final.

Aunque las gráficas muestren una precisión casi perfecta y una pérdida extremadamente baja, como se representa en la siguiente imagen los resultados prácticos no son satisfactorios. Así, representando una predicción del modelo:

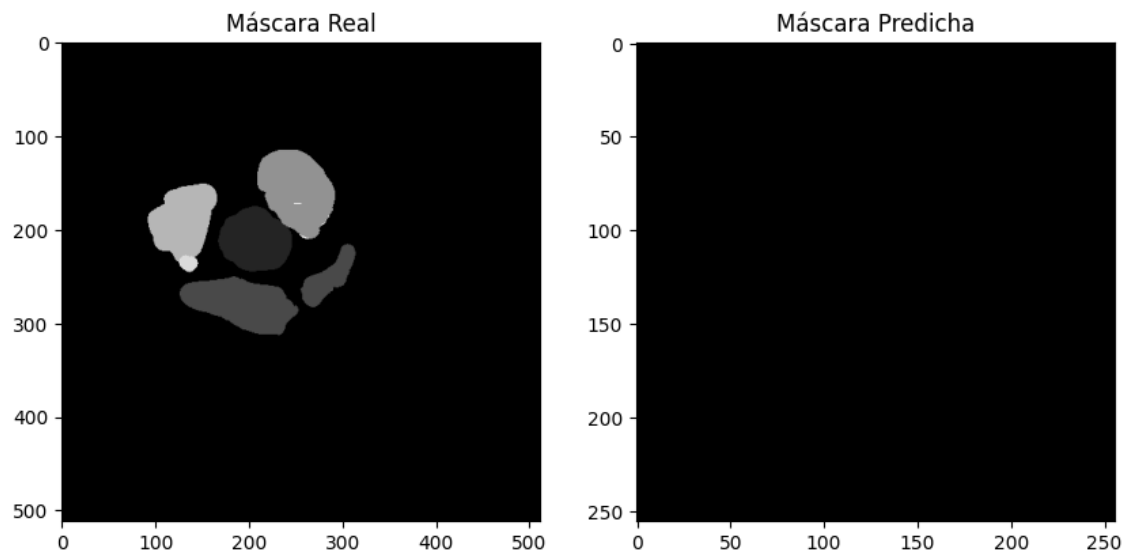


Figura 13. Predicción visual modelo final.

Como se puede observar en la imagen anterior, los resultados obtenidos no son los esperados, ya que la máscara generada es completamente negra. Esto indica un problema significativo en el proceso de segmentación, donde el modelo no está logrando identificar ninguna de las características relevantes de la imagen.

Para abordar este problema, hemos revisado los datos y los vectores de imágenes de entrenamiento en cada uno de los pasos del proceso. Tras este análisis, podemos intuir que el error podría estar ubicado en la fase de extracción de características utilizando el modelo pre-entrenado. Esto se puede apreciar en la imagen siguiente, que es un ejemplo de las imágenes y máscaras de dimensiones (256, 256, 3) que se introducen al modelo final, las cuales no están proporcionando la información necesaria para una segmentación adecuada.

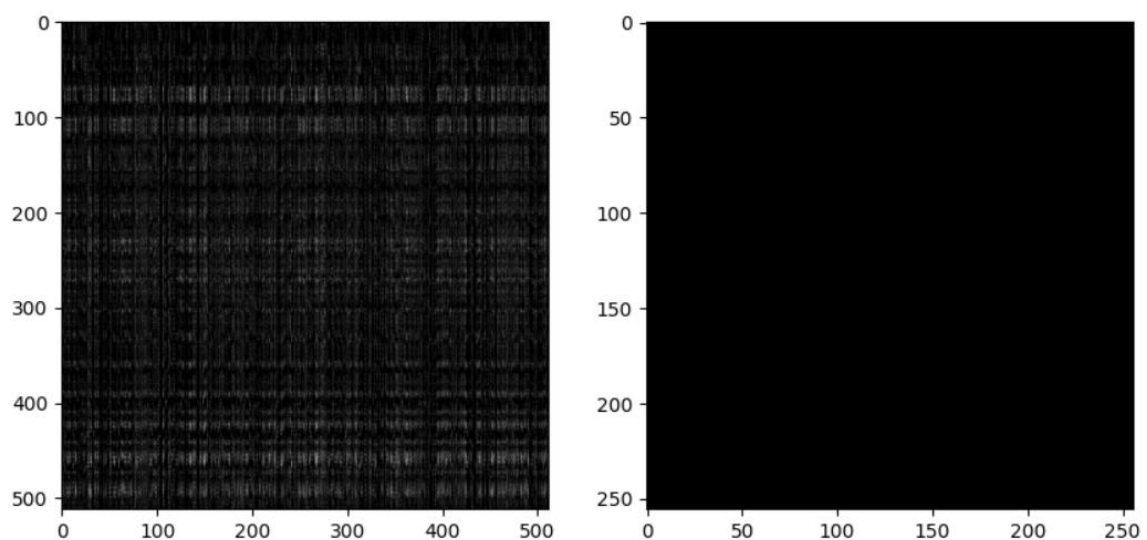


Figura 14. Izquierda imagen de entrenamiento (256, 256, 3), derecha imagen de máscara (256, 256, 3) que se introducen al modelo final.

Conclusiones

A lo largo de este proyecto, hemos llevado a cabo una serie de etapas para desarrollar un modelo de segmentación cardíaca eficaz, comenzando con la exploración y el etiquetado manual del dataset. Posteriormente, se realizó un preprocesamiento avanzado de las imágenes para estandarizar sus dimensiones, asegurando así la uniformidad necesaria para el entrenamiento eficaz del modelo.

La creación de una red convolucional UNET desde cero es el componente central del proyecto, diseñada específicamente para abordar la complejidad propia de la segmentación de imágenes médicas del corazón. Implementamos técnicas de data augmentation para enriquecer nuestro dataset (aunque no fueron utilizadas). Los resultados obtenidos tras esta implementación han sido satisfactorios, reflejando la adecuación del enfoque de aprendizaje profundo y la arquitectura seleccionada para las necesidades específicas del análisis cardíaco.

A pesar del éxito inicial, buscamos un refinamiento adicional mediante la incorporación de un modelo pre-entrenado de cerebros. La estrategia consistió en utilizar únicamente la parte convolucional de este modelo para la extracción de características, seguida de un ajuste fino de los pesos mediante técnicas de fine tuning, con el objetivo de mejorar aún más la precisión y la generalización del modelo UNET.

Sin embargo, este enfoque de refinamiento no resultó en mejoras visibles, probablemente debido a un error en el proceso de extracción de características. Este revés nos proporcionó una valiosa perspectiva sobre la importancia de una integración adecuada y un ajuste fino de los modelos pre-entrenados, especialmente cuando se aplican a contextos y datos que difieren de aquellos con los que fueron originalmente entrenados.

En conclusión, el proyecto ha logrado establecer un método efectivo y eficiente para la segmentación de imágenes del corazón utilizando una red UNET desarrollada desde cero. Aunque la exploración de técnicas adicionales de refinamiento no ha mejorado los resultados, el proceso ha enriquecido nuestro entendimiento sobre la aplicación de redes neuronales convolucionales en el ámbito de la imagen médica y ha señalado áreas clave para futuras investigaciones y desarrollo. Esta experiencia subraya la importancia de la adaptabilidad y la evaluación continua en el campo del aprendizaje automático y la visión por computadora aplicada a la medicina.