Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias.
Maestría en Ciencia de Datos.
Anteproyecto de Grado.

Análisis comparativo de Modelos de Segmentación en imágenes de tomografía computarizada (CT) del área abdominal

Jan Polanco Velasco
Stefania Astudillo Bello

Director: Dr. Julián Gil González

8 de Agosto de 2024

Pontificia Universidad
JAVERIANA
Cali

Índice general

1. Definición Del Problema **3**
1.1. Planteamiento Del Problema . . . 3
1.2. Formulación del Problema . . . 4

2. Objetivos Del Proyecto **6**
2.1. Objetivo General . . . 6
2.2. Objetivos Específicos . . . 6

3. Marco Teórico y Antecedentes **7**
3.1. Marco Teórico . . . 7
3.2. Antecedentes . . . 11

4. Justificación **20**

5. Metodología **25**
5.1. Preprocesamiento de la base de datos . . . 25
5.1.1. Descripción de la base de datos . . . 26
5.2. Propuestas de preprocesamiento para las imágenes . . . 27
5.2.1. Preprocesamiento y Eliminación de Ruido en Imágenes de *CT* . . . 28
5.2.2. Ampliación de Datos (*Data Augmentation*) . . . 29
5.3. *Backbones* . . . 33
5.3.1. VGG16 . . . 33
5.3.2. Resnet50 . . . 34
5.4. Arquitecturas de segmentación de *Deep Learning* . . . 35
5.4.1. Modelo *U-Net* . . . 35
5.4.2. Modelo *SegNet* . . . 37
5.4.3. Modelo *FPN (Feature Pyramid Network)* . . . 39
5.4.4. Modelo *LinkNet* . . . 41

6. Resultados **44**
6.1. Validación de los modelos obtenidos . . . 44
6.1.1. Análisis de resultados y comparación . . . 46
6.1.2. Tabla general de puntajes . . . 65

7. Conclusiones y trabajos futuros **76**
7.1. Conclusiones . . . 76
7.2. Trabajos Futuros . . . 77

8. Anexos **80**
8.1. Repositorio de GitHub 80
8.2. Formula de Puntaje 80
8.3. Códigos 80

Bibliografía **84**

Índice de figuras

5.1. Imagen y mascara original del paciente cero. Fuente: *Autores*. 26
5.2. División porcentual del conjunto de datos. Fuente: *Autores*. 27
5.3. Imagen y mascara sin camilla del paciente cero. Fuente: *Autores*. 29
5.4. Imagen del paciente cero, aumentada. Fuente: *Autores*. 31
5.5. Diagrama de flujo - *Data Augmentation*. Fuente: *Autores*. 32
5.6. Backbone VGG16. Fuente: *[17]*. 34
5.7. Arquitectura *U-net*. Fuente: *[34]*. 36
5.8. Arquitectura *SegNet*. Fuente: *[4]*. 38
5.9. Arquitectura *FPN*. Fuente: *[9]*. 40
5.10. Arquitectura LinkNet. Fuente: *[6]*. 42

6.1. Gráfico de imágenes y mascaras en *One-Hot encoding*. Fuente: *Autores*. 44
6.2. Función de perdidas - U-Net. Fuente: *Autores*. 48
6.3. Métrica Accuracy - U-Net. Fuente: *Autores*. 49
6.4. Función de perdidas - SegNet. Fuente: *Autores*. 51
6.5. Métrica Accuracy - SegNet. Fuente: *Autores*. 53
6.6. Función de perdidas - U-Net con VGG16. Fuente: *Autores*. 55
6.7. Métrica *Accuracy* - U-Net con VGG16. Fuente: *Autores*. 56
6.8. Función de perdidas - U-Net con ResNet50. Fuente: *Autores*. 58
6.9. Métrica *Accuracy* - U-Net con ResNet50. Fuente: *Autores*. 58
6.10. Función de perdidas - FPN con ResNet50. Fuente: *Autores*. 60
6.11. Métrica *Accuracy* - FPN con ResNet50. Fuente: *Autores*. 61
6.12. Función de perdidas - LinkNet con VGG16. Fuente: *Autores*. 63
6.13. Métrica *Accuracy* - LinkNet con VGG16. Fuente: *Autores*. 64
6.14. Inferencia No. 29 con *U-Net*. Fuente: *Autores*. 66
6.15. Inferencia No. 29 con *SegNet*. Fuente: *Autores*. 68
6.16. Inferencia No. 29 con *U-Net* con *ResNet50*. Fuente: *Autores*. 68
6.17. Inferencia No. 29 con *FPN* con *ResNet50*. Fuente: *Autores*. 70

Índice de cuadros

5.1. Mapas de características de entrada y salida 42

6.1. Características generales y tiempos de entrenamiento de las arquitecturas 46
6.2. Tabla de resultados de U-Net en Train 50
6.3. Tabla de resultados de U-Net en Test 50
6.4. Tabla de resultados de SegNet en Train 54
6.5. Tabla de resultados de SegNet en Test 54
6.6. Tabla de resultados de U-Net con VGG16 en Train 56
6.7. Tabla de resultados de U-Net con VGG16 en Test 57
6.8. Tabla de resultados de U-Net con ResNet50 en Train 59
6.9. Tabla de resultados de U-Net con ResNet50 en Test 59
6.10. Tabla de resultados de FPN con ResNet50 en Train 62
6.11. Tabla de resultados de FPN con ResNet50 en Test 62
6.12. Tabla de resultados de LinkNet con VGG16 en Train 63
6.13. Tabla de resultados de LinkNet con VGG16 en Test 64
6.14. Tabla general de puntajes 65

Introducción

La segmentación de imágenes médicas es una tarea esencial en el ámbito de la radiología y la medicina moderna, ya que permite identificar y delimitar estructuras anatómicas y patológicas en las imágenes obtenidas mediante técnicas avanzadas de diagnóstico por imágenes. En particular, la tomografía computarizada (CT) del área abdominal es una herramienta fundamental para detectar y evaluar diversas patologías, incluyendo lesiones en órganos abdominales causadas por diferentes enfermedades [26]. Sin embargo, la segmentación manual de estas imágenes presenta desafíos significativos debido a la similitud de tonalidades entre ciertos órganos y las características intrínsecas de las imágenes de CT, tales como el ruido, la baja resolución y el contraste limitado[28].

En este contexto, los avances en técnicas de *Machine Learning*, y más específicamente en *Deep Learning*, han abierto nuevas oportunidades para mejorar la precisión y eficiencia de la segmentación de imágenes médicas [3]. Las redes neuronales profundas (*Deep Neural Networks*) han demostrado ser particularmente efectivas en diversas tareas de visión por computador, incluyendo la clasificación, detección y segmentación de imágenes, gracias a su capacidad para aprender características jerárquicas y abstractas a partir de los datos de entrada sin necesidad de extracción manual de rasgos [8].

El presente trabajo de grado titulado "Análisis comparativo de Modelos de Segmentación en imágenes de tomografía computarizada (CT) del área abdominal" se propone desarrollar y comparar seis modelos de *Deep Learning* ampliamente utilizados en tareas de segmentación: *U-Net* [34], *U-Net con Backbone VGG16*, *U-Net con Backbone ResNet50*, *SegNet*, *FPN con Backbone ResNet50* y *LinkNet con Backbone VGG16*. El objetivo principal de este estudio es determinar cuál de estos modelos ofrece el mejor rendimiento en términos de precisión, *recall*, coeficiente *Dice*, *accuracy*, *IoU* y *loss*, al segmentar imágenes de CT del área abdominal.

El proyecto se estructura en varias etapas clave. En primer lugar, se lleva a cabo una recolección y procesamiento exhaustivo de los datos de CT abdominal, incluyendo técnicas de limpieza de artefactos y ruido adicional, así como el uso de *Data Augmentation* para aumentar la diversidad del conjunto de datos [1]. Posteriormente, se implementan y entrenan los modelos de segmentación seleccionados utilizando plataformas avanzadas de computación, como *Kaggle*, ajustando los parámetros específicos para el problema en cuestión.

La evaluación del rendimiento de cada modelo se realiza utilizando un conjunto de métricas rigurosas y específicas, tales como precisión, *recall*, coeficiente *Dice*, *accuracy*, *IoU*, *loss*, *F1* y *F2*, permitiendo un análisis comparativo detallado de los resultados obtenidos. Además, se consideran aspectos como el tiempo de entrenamiento y la capacidad de generalización de los modelos a nuevos datos no vistos durante el entrenamiento [26].

Entre los resultados esperados, se anticipa que algunos modelos, como *U-Net* sin *Backbone*, puedan sobresalir en términos de precisión y estabilidad en las métricas de evaluación, demostrando una alta capacidad para generalizar sobre datos no vistos y una eficiente adaptación a las características específicas de las imágenes de *CT* abdominal. La implementación de *Backbones* preentrenados como *VGG16* y *ResNet50* en las arquitecturas de *U-Net* y *FPN* también se espera que mejore la extracción de características y la precisión de la segmentación, aprovechando las capacidades de estas redes profundas para capturar detalles finos y complejos en las imágenes [15], [36].

La investigación no solo tiene el potencial de contribuir al avance de la ciencia de datos aplicada al campo de la medicina, sino que también puede tener un impacto significativo en la práctica clínica, proporcionando herramientas más precisas y eficientes para el diagnóstico y tratamiento de enfermedades abdominales. Además, este estudio representa una valiosa oportunidad de formación y capacitación para los estudiantes involucrados, equipándolos con conocimientos especializados y habilidades prácticas en el desarrollo e implementación de tecnologías de vanguardia en el análisis de imágenes médicas.

El trabajo "Análisis comparativo de Modelos de Segmentación en imágenes de tomografía computarizada (CT) del área abdominal" busca abordar los desafíos inherentes a la segmentación de imágenes médicas mediante la implementación y evaluación de diversas arquitecturas de *Deep Learning*, con el objetivo de identificar el modelo más efectivo y robusto para esta tarea crítica en el ámbito de la radiología y la medicina moderna.

Capítulo 1

Definición Del Problema

1.1. Planteamiento Del Problema

La tomografía computarizada (CT) es una técnica de diagnóstico por imágenes que permite obtener cortes axiales del cuerpo humano y reconstruirlos en diferentes planos. Esta técnica es fundamental para detectar y evaluar diversas patologías, incluyendo lesiones en órganos abdominales causadas por diversas enfermedades. Sin embargo, la tomografía computarizada presenta desafíos significativos, como la similitud de tonalidades entre ciertos órganos y las características intrínsecas de las imágenes de CT, tales como el ruido, la baja resolución y el contraste limitado [28].

Actualmente, los especialistas médicos son responsables de la cuantificación de la forma y el tamaño de los órganos, el diagnóstico de enfermedades, la evaluación de la respuesta al tratamiento, y la planificación y navegación quirúrgica. Este proceso generalmente requiere una segmentación manual de las regiones de interés [26]. Sin embargo, la segmentación manual es un proceso lento, tedioso y propenso a errores humanos. Para superar estas limitaciones, se han propuesto métodos automáticos o semiautomáticos que buscan facilitar esta tarea y ofrecer resultados precisos, rápidos y reproducibles.

Los métodos de segmentación automática de imágenes de CT también enfrentan otros desafíos, como la reticencia de los centros médicos a adoptar nuevas tecnologías, la disponibilidad limitada de datos y códigos, y la baja eficiencia computacional. Estas limitaciones dificultan la aplicación generalizada de la segmentación automática de imágenes CT abdominales en entornos clínicos [29].

El Deep Learning (DL), una técnica avanzada de Machine Learning (ML), ha demostrado su eficacia en diversos problemas de visión artificial, incluyendo la clasificación, detección y segmentación de imágenes [3]. Las redes neuronales profundas son capaces de aprender características jerárquicas y abstractas a partir de los datos de entrada sin necesidad de extracción manual de rasgos [8]. Además, estas redes pueden adaptarse a diferentes dominios y tipos de imágenes, lo que las hace especialmente adecuadas para el análisis de imágenes médicas. A pesar de los avances logrados en la segmentación de imágenes CT utilizando redes neuronales profundas, persisten desafíos importantes [2].

Uno de los principales desafíos en la segmentación de imágenes CT con DL es la precisión y robustez. Las redes neuronales deben ser capaces de segmentar con precisión las regiones de interés en las imágenes, incluso en presencia de ruido, artefactos y variaciones en la forma y tamaño de

las estructuras anatómicas [33]. Además, estas redes deben generalizar eficazmente el conocimiento adquirido durante el entrenamiento para segmentar diferentes tipos de imágenes de CT [8]. Esto implica que la red debe ser capaz de aplicar lo aprendido a nuevos conjuntos de datos no vistos previamente, que pueden incluir diversas modalidades de imagen y una variedad de pacientes [26].

Otro desafío crítico es la eficiencia computacional. Dado que las imágenes CT suelen tener alta resolución espacial y se generan en grandes volúmenes, es necesario desarrollar técnicas que permitan segmentar estas imágenes de manera rápida y eficiente, sin comprometer la calidad de los resultados [42].

La segmentación de órganos abdominales en imágenes de CT es un problema de alta relevancia investigativa, como lo demuestra la realización continua de desafíos como el MICCAI FLARE 2023-2024 [1]. Esto indica que es un área activa y relevante de investigación, con el objetivo de mejorar las técnicas de segmentación y sus aplicaciones clínicas.

Por lo tanto, este proyecto se centra en abordar la problemática de la segmentación de órganos abdominales en imágenes de CT mediante la comparación de distintas redes neuronales profundas establecidas. El objetivo es desarrollar un modelo de comparación robusto y eficiente, utilizando métricas precisas que permitan determinar cuál modelo es el más adecuado para este problema de segmentación. Se espera que este enfoque tenga un alto grado de generalización y sea capaz de procesar grandes volúmenes de datos CT en tiempo real, mejorando significativamente el diagnóstico y tratamiento de enfermedades basadas en imágenes médicas.

1.2. Formulación del Problema

En el marco del análisis comparativo de modelos de segmentación aplicados a imágenes de tomografía computarizada *(CT)* del área abdominal, la formulación del problema facilita la identificación del objetivo principal. Esta formulación orienta la investigación al definir las cuestiones específicas que requieren atención, tales como las técnicas y algoritmos idóneos para el preprocesamiento de imágenes, los modelos de *Deep Learning* más eficaces para la segmentación, y los métodos de validación apropiados para evaluar el rendimiento del modelo resultante.

Pregunta Problema: ¿Cuál modelo de segmentación demostró ser más eficiente para lograr una segmentación precisa y confiable en imágenes de tomografía computarizada *(CT)* del área abdominal, al exhibir los mejores resultados en términos de *precisión*, *recall*, y otras métricas pertinentes, a través de un análisis comparativo de diferentes modelos?

Las cuestiones de sistematización planteadas fueron las siguientes:

- ¿Qué técnicas y algoritmos se evidenciaron como los más adecuados para llevar a cabo un

preprocesamiento exhaustivo de las imágenes de tomografía computarizada *(CT)* del área abdominal, con la finalidad de eliminar el ruido y optimizar la calidad de las imágenes?

- ¿Cuáles fueron los tres modelos de *Deep Learning* más eficaces y apropiados para la segmentación de las imágenes de tomografía computarizada *(CT)* del área abdominal, tomando en consideración la precisión y confiabilidad en los resultados de segmentación?
- ¿Cómo se efectuó la validación del modelo de segmentación obtenido, evaluando su desempeño en términos de *precisión*, *recall*, *coeficiente Dice*, *accuracy*, *IoU* y *loss*, garantizando su robustez y generalización en diferentes conjuntos de datos?

Capítulo 2

Objetivos Del Proyecto

2.1. Objetivo General

Realizar un análisis comparativo de modelos de segmentación en imágenes de tomografía computarizada *(CT)* del área abdominal, con el fin de determinar el modelo más efectivo para obtener una segmentación precisa y confiable, que logre los mejores resultados en términos de *precisión*, *recall*, *coeficiente Dice*, *accuracy*, *IoU* y *loss*.

2.2. Objetivos Específicos

- Realizar un procesamiento previo y exhaustivo de las imágenes de tomografía computarizada *(CT)* del área abdominal, con el objetivo de eliminar el ruido y mejorar la calidad de las imágenes, facilitando así el proceso de segmentación.
- Implementar y entrenar seis modelos definidos de *Deep Learning* para la segmentación de las imágenes de tomografía computarizada *(CT)* del área abdominal, utilizando modelos adecuados y técnicas avanzadas de *Machine Learning*, con el propósito de lograr una segmentación precisa y confiable.
- Realizar la validación del modelo de segmentación obtenido, evaluando su desempeño en términos de *precisión*, *recall*, *coeficiente Dice*, *accuracy*, *IoU* y *loss*.

Capítulo 3

Marco Teórico y Antecedentes

3.1. Marco Teórico

La segmentación de imágenes médicas es una tarea fundamental en el análisis de imágenes médicas, que consiste en identificar y delimitar regiones de interés en las imágenes, como órganos, lesiones y tejidos. La segmentación tiene muchas aplicaciones clínicas, como el diagnóstico, el tratamiento, la planificación quirúrgica y la evaluación de resultados. Sin embargo, la segmentación de imágenes médicas también presenta muchos desafíos, como la variabilidad anatómica, la calidad de imagen, el ruido, la oclusión y la falta de datos etiquetados [14].

Entre las diferentes modalidades de imagen médica, las imágenes de tomografía computarizada *(CT)* son ampliamente utilizadas para la segmentación de órganos y lesiones debido a su alta resolución espacial y su capacidad para capturar estructuras internas. Las imágenes de *CT* se obtienen mediante un equipo de rayos X y tecnología computarizada que permite obtener imágenes de cortes axiales del cuerpo. Estas imágenes representan la atenuación de los rayos X al atravesar los tejidos y se expresan mediante números *CT* o coeficientes de atenuación [18].

Para segmentar las imágenes de *CT*, se han desarrollado diversos métodos y algoritmos basados en técnicas clásicas o en aprendizaje automático. Entre ellos, los métodos basados en redes neuronales convolucionales *(CNN)* han demostrado un gran potencial y rendimiento debido a su capacidad para extraer características complejas y proporcionar resultados precisos y eficientes. Las *CNN* son modelos de aprendizaje profundo que utilizan operaciones de convolución para extraer características de las imágenes. [22].

Sin embargo, las *CNN* también tienen algunas limitaciones, como la necesidad de una gran cantidad de datos etiquetados para entrenarlas, la dificultad para capturar el contexto global y local de las imágenes y la falta de generalización a nuevos objetos o dominios. Para superar estas limitaciones, se han propuesto algunos métodos innovadores que combinan las *CNN* con otras técnicas o paradigmas [8].

Uno de ellos es el *Segment Anything Model (SAM)*, que es un modelo de segmentación fundacional que puede segmentar cualquier objeto en una imagen en función de indicaciones proporcionadas por el usuario, como puntos, cajas o textos. *SAM* está basado en redes neuronales transformadoras *(TNN)* o simplemente como "Transformers" y se entrena con el conjunto de datos de segmentación más grande disponible. *SAM* ha logrado resultados impresionantes en la segmentación de imágenes

naturales, y se espera que también pueda ser útil en la segmentación de imágenes médicas. Sin embargo, su rendimiento en este ámbito requiere una mayor validación, por lo que su aplicación específica en la segmentación de imágenes médicas se realizará en trabajos futuros [20].

Para evaluar la calidad o el rendimiento de los métodos o algoritmos de segmentación, se utilizan diferentes medidas o métricas que pueden ser objetivas o subjetivas, dependiendo de si se basan en criterios matemáticos o en la percepción humana [7]. Para esto se pueden considerar: La función de perdidas *(Loss Function)*, Intersección sobre Unión *(Intersection over Union IOU)*, Coeficiente de *Dice*, *precisión* y exhaustividad *(recall)*, para esto se detallarán de forma general las distintas métricas [8].

La función de pérdida es una métrica utilizada en *Machine Learning* para evaluar qué tan bien se están realizando las predicciones de un modelo en comparación con los valores reales. Esta función cuantifica la discrepancia o error entre las predicciones del modelo y las etiquetas reales de los datos. La Entropía Cruzada (*Cross Entropy*) es una función de pérdida comúnmente utilizada en problemas de clasificación (*véase Ecuación 3.1*). Esta función compara la distribución de probabilidades predicha por el modelo con la distribución de probabilidades reales de las clases de los datos [8].

$$CE(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i). \tag{3.1}$$

Donde CE es la función de pérdida de Entropía Cruzada, y es el vector de probabilidades reales de las clases, $\hat{y}$ es el vector de probabilidades predicho por el modelo y n es el número de clases. La fórmula realiza una sumatoria sobre todas las clases y compara la probabilidad real y_i con la probabilidad predicha $\hat{y}_i$, calculando el logaritmo de esta última. El signo negativo se utiliza para invertir la dirección de la pérdida, ya que se busca minimizar la función de pérdida.

En particular, la Entropía Cruzada Categórica (*Categorical Cross Entropy*) es una variante de la Entropía Cruzada utilizada específicamente para problemas de clasificación multiclase (*véase Ecuación 3.2*), donde cada instancia de los datos pertenece a una sola clase entre un conjunto de clases posibles. La fórmula de la Entropía Cruzada Categórica se expresa como:

$$CCE(y, \hat{y}) = -\sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} \log(\hat{y}_{ij}). \tag{3.2}$$

Donde CCE es la función de pérdida de Entropía Cruzada Categórica, y_{ij} es un valor binario (0 o 1) por la codificación *One-Hot* indicando si la instancia i pertenece a la clase j, $\hat{y}_{ij}$ es la probabilidad predicha de que la instancia i pertenezca a la clase j, n es el número de instancias y k es el número de clases. Esta fórmula realiza una doble sumatoria sobre todas las instancias y todas

las clases, evaluando la discrepancia entre las probabilidades reales y predichas para cada clase y cada instancia.

La Entropía Cruzada Categórica es particularmente efectiva en escenarios donde se desea penalizar fuertemente las predicciones incorrectas y es una elección común en modelos de redes neuronales para tareas de clasificación multiclase. Minimizar esta función de pérdida durante el entrenamiento del modelo conduce a una mejor precisión en la clasificación de las instancias en sus respectivas clases.

Sin embargo, la Entropía Cruzada Categórica presenta algunas desventajas. Aunque generalmente es estable, las operaciones logarítmicas en la Entropía Cruzada Categórica pueden llevar a problemas de estabilidad numérica en algunos casos. Además, las etiquetas verdaderas necesitan estar codificadas en *One-Hot*, lo que puede aumentar el uso de memoria. Por último, esta función de pérdida puede ser sensible a conjuntos de datos desbalanceados, donde el número de muestras en cada clase no es igual.

La función de Intersección sobre Unión *(IoU)*, también conocida como índice de *Jaccard* o coeficiente de similitud de *Jaccard* originalmente denominado *coefficient de communauté* por Paul Jaccard, es una métrica comúnmente utilizada para evaluar la precisión de la detección o segmentación de objetos en tareas de visión por computador *(véase Ecuación 3.3)*. Mide la proporción de píxeles comunes entre la imagen segmentada y la imagen de referencia *(ground truth)*.

$$IoU = \frac{A \cap B}{A \cup B} = \frac{\text{Área de Intersección}}{\text{Área de Unión}}. \tag{3.3}$$

El índice de *Jaccard* mide la similitud entre conjuntos finitos de muestras, y se define como el tamaño de la intersección dividido por el tamaño de la unión de los conjuntos de muestras. La IoU proporciona un valor entre 0 y 1, donde 0 indica una falta de superposición entre las regiones y 1 indica una coincidencia perfecta entre ellas. Un valor más alto de IoU indica una mayor precisión en la detección o segmentación de los objetos.

El coeficiente de Dice, también conocido como coeficiente de *Sørensen-Dice* o similitud de Dice, es una medida que evalúa la superposición espacial entre dos conjuntos, como imágenes segmentadas y una imagen de referencia *(véase Ecuación 3.4)*. Es una estadística utilizada para evaluar la similitud de dos muestras, y se emplea frecuentemente en problemas de segmentación binaria y *multiclase*.

$$DICE = \frac{2\,|A \cap B|}{|A| + |B|}$$

A diferencia de otras métricas, el Coeficiente de Dice penaliza tanto los falsos positivos como los falsos negativos. Se calcula mediante el doble del área de intersección entre las máscaras predichas

y verdaderas, dividido por el área total de ambas máscaras. Un valor más alto indica una mejor coincidencia entre las máscaras [32], [37], [12].

$$\text{Dice} = \frac{2TP}{2TP + FP + FN}. \tag{3.4}$$

El Coeficiente de Dice proporciona un valor entre 0 y 1, donde 0 indica una falta de superposición entre las regiones y 1 indica una coincidencia perfecta entre ellas. Un valor más alto de Dice indica una mayor precisión en la segmentación de los objetos.

La fórmula DICE y el F-score con $\beta = 1$ están relacionados entre sí, ya que ambas medidas intentan capturar la similitud entre dos conjuntos (o la precisión y el recall de un clasificador). Sin embargo, la fórmula DICE es más general y se puede aplicar a cualquier par de conjuntos, mientras que el F-score está específicamente diseñado para evaluar el desempeño de un clasificador.

La precisión es una medida que evalúa la exactitud de un algoritmo al clasificar píxeles como parte de la región de interés, creando un criterio que mide la precisión entre los valores verdaderos *(ground truth)* y la predicción *(véase Ecuación 3.5)*. Se calcula como la proporción de píxeles correctamente clasificados como pertenecientes al objeto, respecto a todos los píxeles clasificados como tal [7].

$$\text{Precisión} = \frac{TP}{TP + FP}. \tag{3.5}$$

Un valor más alto de precisión indica una mayor exactitud en la clasificación de los píxeles como parte del objeto de interés. La precisión proporciona un valor entre 0 y 1, donde 0 indica que ningún píxel clasificado como parte del objeto es correcto y 1 indica que todos los píxeles clasificados como parte del objeto son correctos.

El *Recall*, también conocido como sensibilidad, es una medida que evalúa la exhaustividad de un algoritmo al clasificar píxeles como parte de la región de interés, creando un criterio que mide la exhaustividad entre el valor verdadero *(ground truth)* y la predicción *(véase Ecuación 3.6)*. Se calcula como la proporción de píxeles correctamente clasificados como pertenecientes al objeto, respecto a todos los píxeles que realmente pertenecen al objeto [7].

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3.6}$$

El *Recall* proporciona un valor entre 0 y 1, donde 0 indica que ningún píxel perteneciente al objeto fue correctamente clasificado y 1 indica que todos los píxeles pertenecientes al objeto fueron correctamente clasificados.

El *F-score* puede interpretarse como un promedio ponderado de la precisión y el *recall (véase Ecuación 3.7)*, donde el *F-score* alcanza su mejor valor en 1 y su peor valor en 0. La contribución relativa de la precisión y el *recall* al *F1-score* es igual. La fórmula para el *F-score* es:

$$F_\beta(\text{precisión}, \text{recall}) = (1 + \beta^2) \cdot \frac{\text{precisión} \cdot \text{recall}}{\beta^2 \cdot \text{precisión} + \text{recall}}. \tag{3.7}$$

La fórmula en términos de errores de tipo I y tipo II es:

$$L(\text{TP}, \text{FP}, \text{FN}) = \frac{(1 + \beta^2) \cdot \text{TP}}{(1 + \beta^2) \cdot \text{FP} + \beta^2 \cdot \text{FN} + \text{FP}}, \tag{3.8}$$

donde:

- TP - verdaderos positivos;
- FP - falsos positivos;
- FN - falsos negativos.

La métrica F1 *(véase Ecuación 3.9)*combina la precisión y el *recall* en una sola medida utilizando la media armónica (con $\beta = 1$), y se define como:

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}}. \tag{3.9}$$

La métrica F2 *(véase Ecuación 3.10)* es similar a la F1, pero pone más peso en el *recall* (con $\beta = 2$), y se define como:

$$F2 = 5 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{4 \cdot \text{Precisión} + \text{Recall}}. \tag{3.10}$$

Un valor más alto de *F1* y *F2* indica un mejor equilibrio entre precisión y *recall*, con la métrica *F2* favoreciendo escenarios donde la exhaustividad (*recall*) es más importante que la exactitud (precisión). La segmentación puede influir en la eficiencia y la precisión de los modelos, así como en la capacidad para manejar grandes volúmenes de datos y complejidad anatómica o de pequeños elementos [2], [23], [28].

3.2. Antecedentes

Se utilizó *Google Scholar* para establecer un rango temporal entre 2021 y 2023, seleccionando los artículos más pertinentes y actualizados en el área de modelos de segmentación de imágenes de tomografía computacional. Estos artículos actúan como punto de partida para la construcción de

los antecedentes más relevantes de la investigación actual.

Los modelos actuales suelen estar adaptados a modalidades y objetivos específicos de imagen, y su capacidad de generalización es limitada, por lo tanto, desarrollar modelos fundamentales que puedan adaptarse a diversas modalidades y objetivos de imagen es de suma importancia para avanzar en el análisis de imágenes médicas. En este contexto, se han realizado diversas investigaciones para proponer y evaluar diferentes métodos y modelos para la segmentación de imágenes de *CT*, especialmente en el ámbito del tórax y el abdomen. A continuación, se presentan algunos ejemplos recientes de estos trabajos:

En el artículo *"An automatic method for segmentation of liver lesions in computed tomography images using deep neural networks"* realizado por Araújo et al. (2021) se presenta un método computacional para segmentar lesiones hepáticas en imágenes de tomografía computarizada *(CT)*. El método propuesto utiliza dos modelos de redes neuronales convolucionales profundas y técnicas de procesamiento de imágenes. El método se evaluó usando un conjunto de 131 imágenes de *CT* del conjunto de datos *LiTS (Liver Tumor Segmentation Challenge)* [2].

El mejor resultado fue un coeficiente de correlación de Matthews *(MCC)* del 83,62 %, una sensibilidad del 83,86 %, una especificidad del 99,96 %, un coeficiente de Dice del 82,99 %, un error de superposición volumétrica *(VOE)* del 27,89 % y una diferencia relativa de volumen *(RVD)* del 1,69 %. Los autores muestran que el problema de la segmentación de lesiones hepáticas en imágenes de *CT* se puede resolver eficientemente mediante el uso de redes neuronales convolucionales profundas para definir el alcance del problema y segmentar las lesiones con precisión [2].

El proyecto actual se inspira en el enfoque exitoso de Araújo et al. (2021), pero se diferencia al ampliar el alcance a toda el área abdominal, ajustar las técnicas de procesamiento previo y considerar la evaluación con métricas específicas para la anatomía abdominal.

Por su parte, Radiuk (2020) elabora una investigación llamada *"Applying 3D U-Net Architecture to the Task of Multi-Organ Segmentation in Computed Tomography"* donde propone un modelo modificado de redes neuronales convolucionales totalmente *(FCNs)* dedicadas al procesamiento de imágenes volumétricas de tomografía computarizada *(CT)* en las tareas de segmentación semántica automática. El método propuesto se entrenó y probó en un conjunto de datos compilado manualmente de escaneos de *CT*. El modelo mejorado de 3D *U-Net* logró un puntaje promedio del coeficiente de similitud de *Sørensen-Dice (SDSC)* del 84,8 % en el subconjunto de prueba entre varios órganos abdominales. El autor también comparó su modelo con los resultados reconocidos del estado del arte y demostró que los modelos basados en 3D *U-Net* podrían lograr un rendimiento y una eficiencia competitivos en la tarea de segmentación multiorgánica [33].

El trabajo de Radiuk (2020), ofrece una perspectiva valiosa al proponer una arquitectura específica para la segmentación multiorgánica con un énfasis en el uso de modelos 3D *U-Net*. El proyecto

actual puede beneficiarse de estas ideas al adaptar la arquitectura, considerar métricas específicas y explorar la eficiencia en la segmentación de órganos en el área abdominal.

En su trabajo llamado *"When SAM Meets Medical Images: An Investigation of Segment Anything Model (SAM) on Multi-phase Liver Tumor Segmentation"*, Hu y Li (2023) exploran la capacidad de *SAM* para el análisis de imágenes médicas, especialmente para la segmentación de tumores hepáticos multifase *(MPLiTS)*, en términos de indicaciones, resolución de datos y fases. *SAM* con tres variantes *(ViT-B, ViT-L, ViT-H)* se valida en un conjunto de datos interno que recopila 388 pacientes con 1552 volúmenes de tomografía computarizada con contraste mejorado multifase *(CECT)*. Los resultados muestran que *SAM* podría no alcanzar el rendimiento esperado para *MPLiTS*. Sin embargo, los resultados cualitativos muestran que *SAM* es una herramienta poderosa para mejorar la eficiencia de la anotación significativamente [19].

Aunque los resultados muestran que *SAM* no puede alcanzar el rendimiento esperado para *MPLiTS*, se destaca que *SAM* es una herramienta poderosa para mejorar la eficiencia de la anotación. Esto puede proporcionar información valiosa para el proyecto actual al considerar no solo el rendimiento cuantitativo sino también aspectos prácticos como la eficiencia en el proceso de anotación. El trabajo de Hu y Li aporta al proyecto actual al explorar la aplicación de *SAM* en la segmentación de tumores hepáticos multifase. El proyecto puede beneficiarse de esta investigación al adaptar las variantes de modelos y considerar la eficiencia práctica en la anotación de imágenes en el área abdominal.

En el proyecto *"Lung computed tomography image segmentation based on U-Net network fused with dilated convolution"* Chen et al. (2021) proponen una red *U-Net (DC-U-Net)* fusionada con convolución dilatada y comparan los resultados de la segmentación de imágenes de tomografía computarizada *(CT)* de pulmón con *DC-U-Net*, Otsu y el crecimiento regional. Utilizan la intersección sobre la unión *(IOU)*, el coeficiente de Dice y la precisión para evaluar el rendimiento de los tres algoritmos. Los resultados muestran que la imagen segmentada por *DC-U-Net* es más cercana a la verdad de fondo que las otras dos. El *IOU* de *DC-U-Net* es 0.9627 y el coeficiente de Dice es 0.9743, que es cercano a 1 y mucho más alto que los otros dos algoritmos. Los autores concluyen que este método acelera la segmentación, simplifica los pasos de la segmentación de imágenes médicas y proporciona una mejor segmentación para los tejidos posteriores de vasos sanguíneos pulmonares, tráquea y otros [7].

El trabajo de Chen et al. (2021) ofrece un enfoque específico para la segmentación de imágenes de tomografía computarizada de pulmón utilizando la arquitectura *DC-U-Net*. Mientras que el proyecto actual se centra en el área abdominal, puede tomar ideas sobre arquitecturas de redes y métricas de evaluación para adaptarlas a sus necesidades específicas.

En el artículo *"Lung Segmentation on High-Resolution Computerized Tomography Images Using Deep Learning: A Preliminary Step for Radiomics Studies"* , Comelli et al. (2020) tienen como ob-

jetivo identificar un enfoque de segmentación de aprendizaje profundo automático, preciso y rápido, aplicado al parénquima, utilizando un conjunto de datos muy pequeño de imágenes de tomografía computarizada de alta resolución de pacientes con fibrosis pulmonar idiopática. De esta manera, pretenden mejorar la metodología realizada por los operadores sanitarios en los estudios radiómicos, donde se deben utilizar métodos de segmentación independientes del operador para identificar correctamente el objetivo y, en consecuencia, el modelo de predicción basado en la textura [10].

Los autores investigan dos modelos de aprendizaje profundo: El primero es *U-Net*, ya utilizado en muchas tareas de segmentación de imágenes biomédicas, y el segundo es *E-Net*, utilizado para tareas de segmentación de imágenes en coches autónomos, donde la disponibilidad de hardware es limitada y la segmentación precisa es crítica para la seguridad del usuario. Su pequeño conjunto de datos de imágenes está compuesto por 42 estudios de pacientes con fibrosis pulmonar idiopática, de los cuales solo 32 se utilizaron para la fase de entrenamiento. Comparan el rendimiento de los dos modelos en términos de la similitud de su resultado de segmentación con el estándar de oro y en términos de sus requisitos de recursos.

Los resultados muestran que *E-Net* se puede utilizar para obtener una segmentación precisa (coeficiente de similitud de dice = 95.90 %), rápida (20.32 s) y clínicamente aceptable de la región pulmonar. Los autores concluyen que los modelos de aprendizaje profundo se pueden aplicar eficientemente para segmentar y cuantificar rápidamente el parénquima de los pacientes con fibrosis pulmonar, sin supervisión alguna del radiólogo, con el fin de producir resultados independientes del usuario [10].

El trabajo de Comelli et al. (2020) proporciona una perspectiva valiosa al aplicar modelos de aprendizaje profundo para la segmentación del parénquima pulmonar. Aunque el contexto es diferente, el proyecto actual puede beneficiarse de la elección de modelos y la evaluación de rendimiento presentadas en este trabajo.

En el artículo *"Kidney segmentation from computed tomography images using deep neural network"* Batista da Cruz et al. (2020) proponen un método automático para delimitar los riñones en imágenes de tomografía computarizada *(CT)* usando técnicas de procesamiento de imágenes y redes neuronales convolucionales profundas *(CNNs)* para minimizar los falsos positivos. El método propuesto tiene cuatro pasos principales: (1) adquisición del conjunto de datos *KiTS19*, (2) reducción del alcance usando *AlexNet*, (3) segmentación inicial usando *U-Net* 2D y (4) reducción de falsos positivos usando procesamiento de imágenes para mantener los elementos más grandes (riñones) [11].

El método propuesto se evaluó en 210 *CTs* del conjunto de datos *KiTS19* y obtuvo el mejor resultado con un coeficiente de Dice promedio de 96.33 %, un índice de *Jaccard* promedio de 93.02 %, una sensibilidad promedio de 97.42 %, una especificidad promedio de 99.94 % y una precisión promedio de 99.92 %. En el desafío *KiTS19*, presentó un coeficiente de Dice promedio de 93.03 %. Los autores

concluyen que el problema de segmentación de riñones en *CT* se puede resolver eficientemente usando redes neuronales profundas para definir el alcance del problema y segmentar los riñones con alta precisión y con el uso de técnicas de procesamiento de imágenes para reducir los falsos positivos [11].

El trabajo de Batista da Cruz et al. (2020) proporciona un enfoque eficiente para la segmentación de riñones en imágenes de tomografía computarizada, destacando el uso de redes neuronales profundas y técnicas de procesamiento de imágenes. Aunque el contexto es diferente, el proyecto actual puede beneficiarse de la metodología estructurada y la atención específica a la reducción de falsos positivos presentes en este trabajo.

En el artículo *"Deep Learning for Image Segmentation: A Focus on Medical Imaging"* Khalifa y Badr (2023) presentan una revisión de los modelos y mecanismos de redes neuronales convolucionales profundas *(CNN)* más avanzados utilizados en la segmentación de imágenes. Los autores clasifican los modelos de segmentación según su modelo y su principio de funcionamiento principal. Luego, describen las categorías de *CNN* y discuten varios modelos dentro de cada categoría [22].

A diferencia de otras revisiones existentes, se analizan varias aplicaciones con múltiples adaptaciones de los modelos dentro de cada categoría. Un resumen comparativo se incluye para dar al lector una visión general de los modelos utilizados en diferentes aplicaciones y conjuntos de datos. Este estudio se centra en las aplicaciones de segmentación de imágenes médicas, donde se ilustran los modelos más utilizados y se sugieren otros modelos prometedores que han demostrado su éxito en diferentes dominios. Finalmente, el trabajo actual discute las limitaciones y soluciones actuales junto con las tendencias futuras en el campo [22].

La revisión de Khalifa y Badr (2023) proporciona un marco integral que abarca modelos, adaptaciones, aplicaciones y tendencias en el ámbito de la segmentación de imágenes mediante *CNN*, con un énfasis específico en la imagen médica. El proyecto actual puede beneficiarse de esta revisión al obtener una visión global de las opciones disponibles y comprender las tendencias actuales en el campo de la segmentación de imágenes médicas.

En el artículo *"Deep learning approaches to biomedical image segmentation"* Haque y Neubert (2020) presentan una revisión de los métodos y mecanismos de redes neuronales convolucionales profundas *(CNN)* más avanzados utilizados en la segmentación de imágenes. Los autores revisan los conceptos básicos de los métodos de aprendizaje profundo y ofrecen una visión general de las implementaciones exitosas que involucran la segmentación de imágenes para diferentes aplicaciones médicas [14].

Además, se incluye un resumen comparativo para dar al lector una idea de los modelos utilizados en diferentes aplicaciones y conjuntos de datos. Este estudio se centra en las aplicaciones de segmentación de imágenes biomédicas, donde se ilustran los modelos más utilizados y se sugieren otros modelos prometedores que han demostrado su éxito en diferentes dominios. Finalmente, el

trabajo actual discute las limitaciones y soluciones actuales junto con las tendencias futuras en el campo [14].

La revisión de Haque y Neubert (2020) ofrece una perspectiva integral sobre los enfoques de segmentación de imágenes biomédicas mediante *CNN*, con un enfoque particular en conceptos básicos, implementaciones exitosas y tendencias futuras. El proyecto actual puede beneficiarse de esta revisión al obtener información detallada sobre modelos exitosos y considerar las tendencias actuales en el campo de la segmentación abdominal.

En el artículo *"Accuracy of Segment-Anything Model (SAM) in Medical Image Segmentation Tasks"* He et al. (2023) evalúan la generalización de cero disparos de *SAM* en 12 conjuntos de datos de imágenes médicas diferentes. Los autores comparan la precisión de *SAM* con la de cinco algoritmos de segmentación basados en aprendizaje profundo diseñados específicamente para imágenes médicas. Los autores utilizan diferentes indicadores para medir la dificultad de segmentación y el contraste entre el objetivo y el fondo. Los resultados muestran que *SAM* tiene una precisión significativamente menor que los otros algoritmos en todos los conjuntos de datos, especialmente en imágenes 3D, regiones objetivo que son pequeñas y de bajo contraste, y casos más complejos. Los autores sugieren algunas posibles mejoras para aumentar la precisión de *SAM* en imágenes médicas [16].

El trabajo de He et al. (2023) ofrece una evaluación crítica de la precisión de *SAM* en tareas de segmentación de imágenes médicas, identificando sus limitaciones en ciertos escenarios. Aunque la evaluación se centra en el ámbito médico, el proyecto actual puede aprender de estas limitaciones y sugerencias de mejoras para ajustar su enfoque de segmentación abdominal según las necesidades específicas del conjunto de datos y la tarea a realizar.

En el artículo *"Generalist Vision Foundation Models for Medical Imaging: A Case Study of Segment Anything Model on Zero-Shot Medical Segmentation"*, Shi et al. (2023) examinan el reciente modelo de segmentación de cualquier cosa *(SAM)* en imágenes médicas y reportan resultados cuantitativos y cualitativos de segmentación de cero disparos en nueve conjuntos de datos de segmentación de imágenes médicas, que abarcan diversas modalidades de imagen, como tomografía de coherencia óptica *(OCT)*, resonancia magnética *(MRI)* y tomografía computarizada *(CT)*, así como diferentes aplicaciones que incluyen dermatología, oftalmología y radiología. Los autores comparan el rendimiento de *SAM* con el de otros modelos de visión generalista y específicos para imágenes médicas [35].

Los autores encuentran que, mientras *SAM* presenta un rendimiento de segmentación notable en imágenes del dominio general, su capacidad de segmentación de cero disparos sigue siendo limitada para imágenes fuera de distribución, por ejemplo, imágenes médicas. Además, *SAM* muestra un rendimiento inconsistente de segmentación de cero disparos en diferentes dominios médicos no vistos [35].

Para ciertos objetivos estructurados, por ejemplo, vasos sanguíneos, la segmentación de cero disparos de *SAM* falló completamente. En contraste, un simple ajuste fino del mismo con una pequeña cantidad de datos podría conducir a una mejora notable de la calidad de la segmentación, mostrando el gran potencial y la viabilidad de utilizar *SAM* ajustado para lograr una segmentación precisa de imágenes médicas para un diagnóstico de precisión. El estudio indica la versatilidad de los modelos de visión generalista en la imagen médica y su gran potencial para lograr el rendimiento deseado mediante el ajuste fino y finalmente abordar los desafíos asociados al acceso a conjuntos de datos médicos grandes y diversos en apoyo del diagnóstico clínico [35].

El trabajo de Shi et al. (2023) ofrece una evaluación crítica de *SAM* en el contexto de imágenes médicas, destacando sus limitaciones en la segmentación de cero disparos y su capacidad de ajuste fino para mejorar la precisión en objetivos médicos específicos. Este estudio puede proporcionar valiosas lecciones sobre la aplicabilidad de *SAM* en el proyecto actual y resalta la importancia del ajuste fino para adaptar modelos generalistas a tareas médicas específicas.

En el artículo *"Segment Anything in Medical Images"*, Ma y Wang (2023) presentan *MedSAM*, el primer intento de extender el éxito del modelo de segmentación de cualquier cosa *(SAM)* a las imágenes médicas, con el objetivo de crear una herramienta universal para la segmentación de diversos objetivos médicos. Específicamente, los autores curan un conjunto de datos de imágenes médicas a gran escala, que abarca más de 200.000 máscaras en 11 modalidades diferentes. Luego, desarrollan un método simple de ajuste fino para adaptar *SAM* a la segmentación general de imágenes médicas. Experimentos exhaustivos en 21 tareas de segmentación 3D y 9 tareas de segmentación 2D demuestran que *MedSAM* supera al modelo *SAM* predeterminado con un coeficiente de similitud de dice *(DSC)* promedio de 22.5 % y 17.6 % en las tareas de segmentación 3D y 2D, respectivamente. El código y el modelo entrenado están disponibles públicamente en https://github.com/bowang-lab/MedSAM [27].

El trabajo de Ma y Wang (2023) presenta *MedSAM* como una extensión exitosa de *SAM* para la segmentación de imágenes médicas, destacando la importancia de un conjunto de datos extenso y diversificado, así como un método de ajuste fino para mejorar el rendimiento. Este artículo puede inspirar al proyecto actual al proporcionar un ejemplo de cómo adaptar modelos generales para tareas médicas específicas.

En el artículo *"Segment Anything Model for Medical Images?"*, Huang et al. (2023) analizan el rendimiento del modelo de segmentación de cualquier cosa *(SAM)* en imágenes médicas, y reportan resultados de segmentación de cero disparos en 52 conjuntos de datos de segmentación de imágenes médicas, que abarcan 16 modalidades, 68 objetos y 553K rebanadas. Los autores comparan diferentes estrategias de prueba de *SAM* en el conjunto de datos COSMOS 553K y evalúan la influencia de diferentes factores (como la complejidad del borde basada en *Fourier* y el tamaño de los objetos segmentados) en el rendimiento de la segmentación [20].

Los resultados muestran que *SAM* tiene un mejor rendimiento con pistas manuales como puntos y cajas para la percepción de objetos en imágenes médicas, lo que lleva a un mejor rendimiento en el modo de indicación que en el modo automático. Sin embargo, *SAM* muestra un rendimiento variable en algunos objetos y modalidades específicos, y es imperfecto o incluso falla totalmente en otras situaciones. Los autores concluyen que la capacidad de segmentación de cero disparos de *SAM* no es suficiente para garantizar su aplicación directa a la segmentación de imágenes médicas [20].

El trabajo de Huang et al. (2023) ofrece una evaluación crítica del rendimiento de *SAM* en imágenes médicas, identificando tanto fortalezas como limitaciones. Este análisis puede ser fundamental para el proyecto actual al proporcionar información sobre la viabilidad de *SAM* en el contexto de segmentación abdominal y resaltar áreas donde podrían ser necesarios ajustes o estrategias adicionales.

En el artículo *"Fast and Low-GPU-memory abdomen CT organ segmentation: The FLARE challenge"*, Ma et al. (2022) presentan el primer desafío de segmentación de órganos abdominales que evalúa tanto la precisión como la eficiencia de los métodos de segmentación. Los autores organizan un conjunto de datos grande y diverso, que incluye 511 casos de 11 centros médicos, y abarca 13 órganos abdominales [28].

Los autores proponen dos medidas de clasificación para evaluar el rendimiento de los métodos participantes: una basada en el coeficiente de similitud de *Dice (DSC)* y otra basada en el tiempo de inferencia y el consumo de memoria *GPU*. Los resultados muestran que los métodos basados en redes neuronales convolucionales *(CNN)* superan a los basados en redes neuronales transformadoras *(TNN)* en términos de precisión, pero tienen una mayor variabilidad en términos de eficiencia. Los autores concluyen que el desafío *FLARE* proporciona un punto de referencia integral y desafiante para la segmentación de órganos abdominales en *CT* y fomenta el desarrollo de métodos más precisos y eficientes para la práctica clínica [28].

El trabajo de Ma et al. (2022) presenta el desafío *FLARE* como una iniciativa para evaluar métodos de segmentación de órganos abdominales, abordando tanto la precisión como la eficiencia. Este enfoque integral puede inspirar al proyecto actual a considerar medidas de evaluación más amplias y a buscar un equilibrio entre la precisión y la eficiencia en la implementación de métodos de segmentación abdominal.

Estos antecedentes muestran avances importantes y desafíos actuales en el campo de la segmentación de imágenes médicas. En este sentido, el actual proyecto pretende contribuir al estado del arte y a la mejora de la precisión y eficiencia en la segmentación de imágenes de tomografía computarizada del área abdominal. La investigación no solo tiene el potencial de avanzar la ciencia de datos aplicada a la medicina, sino que también puede impactar significativamente en la práctica

clínica, proporcionando herramientas más precisas y eficientes para el diagnóstico y tratamiento de enfermedades abdominales. Además, este estudio representa una valiosa oportunidad de formación para los estudiantes involucrados, equipándolos con conocimientos especializados y habilidades prácticas en el desarrollo de tecnologías de vanguardia en el análisis de imágenes médicas.

Capítulo 4

Justificación

El desarrollo del proyecto de investigación propuesto, titulado "Análisis comparativo de Modelos de Segmentación en imágenes de tomografía computarizada (CT) del área abdominal", se presenta como una iniciativa altamente significativa y pertinente en el contexto de la radiología y la medicina moderna. La segmentación de imágenes de tomografía computarizada del área abdominal es fundamental para identificar y delimitar estructuras anatómicas y patológicas en imágenes médicas, desempeñando un papel fundamental en el diagnóstico y tratamiento de diversas enfermedades. En este sentido, la integración de técnicas de *Machine Learning*, particularmente el aprendizaje profundo, se vislumbra como un enfoque vanguardista que puede transformar radicalmente este proceso esencial.

El enfoque tradicional de la segmentación manual, llevado a cabo por radiólogos expertos, presenta limitaciones significativas en términos de tiempo, precisión y consistencia. La automatización de este proceso mediante técnicas de *Machine Learning* no solo ahorra tiempo y recursos, sino que también minimiza los errores humanos inherentes al proceso manual. La fatiga, el sesgo y la variabilidad Inter observador, factores recurrentes en la segmentación manual, se eliminan con la implementación de modelos de *Machine Learning*, lo que resulta en resultados más precisos y consistentes.

Cabe destacar que los equipos de *CT* modernos han experimentado mejoras significativas en la resolución de imagen, esto significa que las imágenes de *CT* actuales pueden capturar detalles anatómicos mucho más pequeños y sutiles que en el pasado, la segmentación automatizada se vuelve esencial para analizar estas imágenes de alta resolución de manera eficiente y precisa, ya que sería extremadamente laborioso realizar una segmentación manual detallada en estructuras tan minuciosas.

Uno de los pilares fundamentales que respalda la viabilidad de este proyecto es la disponibilidad de un conjunto de datos extenso y diverso. La calidad y diversidad de los datos son esenciales para garantizar que los modelos de *Machine Learning* pueda generalizar y segmentar una amplia variedad de tejidos, órganos y estructuras anatómicas presentes en las imágenes de tomografía computarizada. La representatividad del conjunto de datos es importante para evitar sesgos y garantizar la robustez y la precisión del modelo. La capacidad de obtener resultados precisos en diferentes tipos de casos clínicos es esencial para avanzar hacia una medicina de precisión, permitiendo diagnósticos más precisos y tratamientos más efectivos para las enfermedades abdominales.

El impacto potencial de este proyecto es significativo y multifacético. En el ámbito de la investigación médica, el proyecto contribuirá al avance general de la ciencia médica al mejorar los métodos de segmentación y análisis de imágenes. Este avance no solo beneficiará a los investigadores, sino que también proporcionará herramientas más precisas y confiables para los profesionales de la salud, mejorando así la calidad de la atención médica en general.

En el ámbito clínico, la implementación de técnicas de segmentación mejoradas tendrá un impacto directo en el diagnóstico y tratamiento de enfermedades abdominales. La obtención de resultados más precisos y confiables permitirá a los médicos tomar decisiones fundamentadas sobre los tratamientos, lo que podría conducir a una mejora significativa en la calidad de vida de los pacientes. Además, la eficiencia en la atención médica se verá notablemente incrementada, ya que los profesionales de la salud podrán acceder a información detallada de manera más rápida y precisa.

Desde una perspectiva académica, el desarrollo de este proyecto no solo demuestra la excelencia académica de los estudiantes involucrados, sino que también demuestra su capacidad para implementar tecnologías de vanguardia en un contexto médico. Las habilidades adquiridas en este proyecto serán importantes para la formación profesional de los estudiantes, equipándolos con conocimientos especializados en ciencia de datos y medicina, preparándolos para abordar desafíos futuros en estos campos interdisciplinarios.

El proceso de realizar un análisis comparativo de arquitecturas de *Deep Learning* proporciona una oportunidad valorable para la formación y capacitación de estudiantes e investigadores, a través de este ejercicio, se adquiere una experiencia práctica en la implementación y evaluación de modelos de aprendizaje profundo, así como una comprensión más profunda de los desafíos y consideraciones específicas en el contexto médico, dicho análisis comparativo impulsa el avance tanto en la investigación médica como en el desarrollo tecnológico, al tiempo que proporciona una valiosa experiencia de aprendizaje para los involucrados en el proyecto.

La segmentación de imágenes de tomografía computarizada *(CT)* del área abdominal es una tarea importante y desafiante, ya que permite obtener información anatómica y funcional de los órganos internos, que puede ser útil para el diagnóstico, el tratamiento y el seguimiento de diversas enfermedades [3]. Sin embargo, la segmentación de estas imágenes presenta varios problemas, como la baja resolución, el ruido, el contraste, la variabilidad y la complejidad de las estructuras, y la presencia de artefactos y patologías [41].

Para resolver estos problemas, se han propuesto diferentes modelos de segmentación basados en *Deep Learning*, que son capaces de aprender de forma automática las características relevantes de las imágenes, y de generar segmentaciones precisas y confiables [31]. Entre estos modelos, se pueden destacar los siguientes:

U-Net, es un modelo de segmentación basado en una arquitectura de codificador-decodificador,

que utiliza conexiones de salto entre las capas para conservar la información espacial y mejorar la precisión de la segmentación. *U-Net* ha demostrado ser muy efectivo para la segmentación de imágenes médicas, especialmente para el hígado, el páncreas y los riñones [41]

Una de las razones para seleccionar la arquitectura *U-Net* en el desarrollo del proyecto es que este ha sido ampliamente adoptado en la comunidad científica y médica debido a su efectividad en la segmentación de imágenes médicas, al implementar *U-Net* en el proyecto, se estaría utilizando una arquitectura establecida y validada, lo que facilitaría la comparación con otros enfoques y contribuiría al avance general en el campo de la segmentación de imágenes médicas, contribuyendo así a la calidad y precisión de los resultados, así como al avance continuo en el campo de la medicina de imágenes [34].

Además, la incorporación de *Backbone VGG16* en la arquitectura *U-Net* se justifica por varias razones. Primero, *VGG16* es una red pre entrenada en un gran conjunto de datos *(ImageNet)*, lo que le permite extraer características robustas y generales que son útiles para la segmentación de imágenes médicas [36]. Este preentrenamiento ayuda a la red a aprender representaciones útiles más rápidamente y puede mejorar significativamente la precisión del modelo. Segundo, *VGG16* ha demostrado ser eficaz en la captura de características a diferentes niveles de abstracción, lo que es importante para segmentar estructuras complejas en imágenes de tomografía computarizada *(CT)*. Tercero, el uso de *VGG16* puede reducir el riesgo de *overfitting*, especialmente cuando se dispone de un conjunto de datos limitado, al aprovechar las características ya aprendidas en su entrenamiento previo [8].

Por último, la combinación de *U-Net* con *VGG16* permite beneficiarse de una arquitectura bien establecida y una potente red de extracción de características, proporcionando así una base sólida para el desarrollo y la comparación de modelos de segmentación de imágenes médicas.

De igual manera, utilizar *ResNet50* como *Backbone* en la arquitectura *U-Net* también presenta importantes ventajas. *ResNet50* es conocido por su innovador uso de bloques residuales, lo que permite un entrenamiento más profundo y efectivo de redes neuronales sin los problemas de degradación que sufren otras arquitecturas profundas [15]. Al integrar *ResNet50* como *Backbone*, *U-Net* puede aprovechar esta capacidad para extraer características más detalladas y complejas de las imágenes médicas, lo que puede mejorar la precisión y la resolución de la segmentación. Además, *ResNet50* ha sido preentrenado en conjuntos de datos extensivos, proporcionando una base sólida para la transferencia de aprendizaje y mejorando la eficiencia del entrenamiento del modelo [8].

La robustez de *ResNet50* en la extracción de características y su capacidad para generalizar a nuevos conjuntos de datos lo hacen ideal para aplicaciones en la segmentación de imágenes *CT*, donde se requiere una detección precisa y confiable de estructuras anatómicas y patológicas. Por estas razones, la combinación de *U-Net* con *ResNet50* como *Backbone* representa una elección estratégica que puede potenciar significativamente los resultados del proyecto.

SegNet, es otro modelo de segmentación basado en una arquitectura de codificador-decodificador, que se diferencia de *U-Net* en que utiliza un método de agrupación máxima con índices, que reduce el número de parámetros y mejora la eficiencia del modelo. *SegNet* ha sido aplicado con éxito para la segmentación de imágenes de *CT* del abdomen, logrando resultados comparables o superiores a otros métodos [39].

Otra arquitectura importante a considerar es la *Feature Pyramid Network (FPN)* con *Backbone ResNet50*. La *FPN* es especialmente útil para tareas de detección y segmentación debido a su capacidad para construir una pirámide de características que combina información de diferentes niveles de resolución. Al utilizar *ResNet50* como *Backbone*, la *FPN* puede beneficiarse de los bloques residuales de *ResNet50*, permitiendo una extracción de características eficaz y profunda a múltiples escalas. Esta combinación es particularmente ventajosa en la segmentación de imágenes médicas, donde es fundamental el poder capturar tanto los detalles finos como las estructuras más grandes y complejas.

La *FPN* con *ResNet50* puede mejorar la precisión y la consistencia de la segmentación al proporcionar una representación más rica y detallada de las características de las imágenes *CT*. Este enfoque no solo optimiza la capacidad de la red para manejar la variabilidad anatómica y la complejidad de las imágenes médicas, sino que también facilita una segmentación más robusta y precisa. Por lo tanto, la inclusión de *FPN* con *Backbone ResNet50* en el proyecto representa una estrategia avanzada que puede elevar significativamente la calidad de los resultados obtenidos.

Finalmente, la utilización de *LinkNet* con *Backbone VGG16* también aporta ventajas significativas al proyecto. *LinkNet* es una arquitectura ligera y eficiente que está diseñada para la segmentación semántica, y cuando se combina con *Backbone VGG16*, se aprovechan las capacidades de extracción de características robustas y profundas de *VGG16*. Esto resulta en una red que puede segmentar eficientemente imágenes médicas manteniendo un buen balance entre precisión y velocidad de procesamiento.

La ligereza de *LinkNet* es especialmente útil cuando se requiere segmentar grandes volúmenes de datos o se necesita una respuesta rápida en aplicaciones clínicas. Además, al utilizar *VGG16*, se beneficia de los pesos preentrenados, lo que facilita el entrenamiento y mejora la precisión, incluso con conjuntos de datos limitados. Por estas razones, la combinación de *LinkNet* con *VGG16* como *Backbone* es una opción estratégica que puede proporcionar resultados precisos y rápidos en la segmentación de imágenes de tomografía computarizada *(CT)*, contribuyendo de manera significativa al éxito del proyecto [6].

Estos modelos se han seleccionado por ser algunos de los más populares y efectivos para la segmentación de imágenes de *CT* del abdomen, según la literatura científica disponible. Estos modelos han demostrado tener un alto rendimiento y una buena precisión para segmentar diferentes

estructuras y órganos abdominales, como el hígado, el páncreas y los riñones [43]; [5]. Además, estos modelos tienen la ventaja de ser flexibles y adaptables a diferentes condiciones y escenarios, como la variabilidad anatómica, la calidad de las imágenes, la presencia de artefactos o patologías [41];[5].

Capítulo 5

Metodología

5.1. Preprocesamiento de la base de datos

El preprocesamiento de datos constituye una etapa fundamental en el proceso de análisis y modelado de datos, especialmente en el ámbito de la segmentación de imágenes médicas. En el contexto de las imágenes de tomografía computarizada *(CT)* del área abdominal, una adecuada preparación de los datos es indispensable para garantizar la precisión y la eficacia de los modelos de segmentación empleados. Esta fase inicial involucra una serie de procedimientos meticulosos que buscan mejorar la calidad y la estructura de los datos, facilitando así su posterior análisis. Entre estos procedimientos se incluyen la corrección de artefactos, la normalización, la estandarización, la segmentación de regiones de interés y la ampliación de datos.

La corrección de artefactos es fundamental para eliminar o minimizar los errores inherentes o las anomalías presentes en las imágenes, como el ruido o las distorsiones, que pueden obstaculizar la interpretación correcta de la información contenida. Por otro lado, la normalización y la estandarización son procedimientos esenciales que permiten homogenizar las escalas de intensidad y las unidades de medida, proporcionando una base consistente para el análisis y la comparación entre diferentes conjuntos de datos o imágenes.

La segmentación de las regiones de interés, por su parte, se enfoca en aislar y destacar las áreas específicas dentro de las imágenes que son relevantes para el estudio, permitiendo una concentración efectiva de los recursos computacionales y analíticos en los elementos relevantes. Además, la ampliación de datos mediante técnicas como la rotación, el escalamiento o la transformación, contribuye a expandir y diversificar el conjunto de datos disponible, enriqueciendo así el entrenamiento y la validación de los modelos, y promoviendo una mejor generalización de estos.

A través del preprocesamiento, se busca no solo optimizar la calidad de los datos, sino también estructurarlos de manera que faciliten la extracción de características y la aplicación de técnicas avanzadas de aprendizaje automático y *deep learning*. La correcta ejecución de esta etapa preparatoria es vital para maximizar el rendimiento y la precisión de los modelos de segmentación en la tarea de identificación y delimitación de órganos y estructuras en las imágenes de tomografía computarizada del área abdominal, lo cual es esencial para un análisis diagnóstico confiable y una interpretación clínica precisa.

5.1.1. Descripción de la base de datos

El conjunto de datos utilizado en este trabajo de grado se adquirió a través de la plataforma *Grand Challenge*, una iniciativa dedicada al fomento del desarrollo integral de soluciones de aprendizaje automático aplicadas a imágenes biomédicas. Específicamente, los datos provienen del desafío *MICCAI 2021 FLARE Challenge: Fast and Low GPU memory Abdominal oRgan sEgmentation*, organizado en el año 2021. La documentación proporcionada en la plataforma detalla que el conjunto de datos es amplio y diversificado, compuesto por imágenes proporcionadas por 11 centros médicos distintos [29], [30].

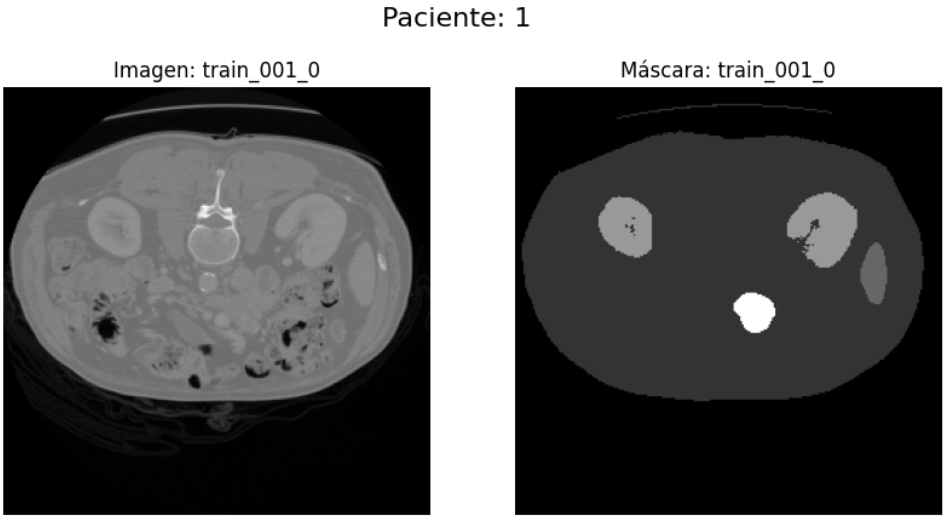

Figura 5.1: Imagen y mascara original del paciente cero. Fuente: *Autores.*

La figura adjunta ilustra la imagen y la máscara correspondiente al paciente No 1 del conjunto de entrenamiento *(train)* y su corte *(slice)* inicial. Para tener una comprensión más profunda acerca del conjunto de datos, se analizó detenidamente la información contenida en sus dos subconjuntos, denominados *train* y *test*. El conjunto *train* se organiza en dos directorios: imágenes y máscaras, albergando cada uno 2725 archivos en formato *numpy* ($*.npy$). Este conjunto incorpora datos de 48 pacientes, cada uno con un número variado de cortes *(slices)*. En promedio, se registraron alrededor de 57 *slices* por paciente, con una desviación estándar de 15.32.

Por otro lado, el conjunto *test* también se estructura en dos directorios: imágenes y máscaras, conteniendo cada uno 513 archivos en formato $*.npy$. Este subconjunto comprende datos de 10 pacientes, cada uno con un número distintivo de *slices*. En promedio, se observaron alrededor de 51 *slices* por paciente, con una desviación estándar de 17.12.

En términos generales como información y meta data, cada imagen posee una resolución de 256 (ancho) x 256 (alto) píxeles. Cada archivo $*.npy$ de las imágenes se halla en formato *Float32*, mientras que el tipo de dato de cada mascara es *uint8*, con un orden en memoria descrito como fila principal orden Fortran: *false*. El conjunto de datos, en su totalidad, tiene 3,238 imágenes, de las cuales 2,725 corresponden al conjunto de entrenamiento, representando el 84 % del total, y 513 al conjunto de prueba, equivalente al 16 % del total. A nivel de distribución de pacientes, el 82.76 % se asignó al conjunto de entrenamiento y el 17.24 % al conjunto de prueba.

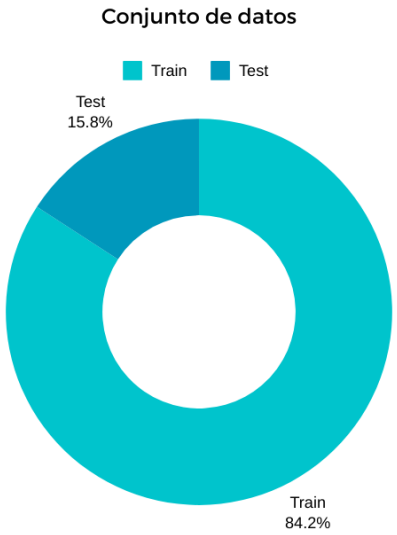

Figura 5.2: División porcentual del conjunto de datos. Fuente: *Autores*.

5.2. Propuestas de preprocesamiento para las imágenes

La tomografía computarizada *(CT)* es una técnica de imagen médica fundamental que proporciona vistas detalladas de estructuras internas del cuerpo humano. Estas imágenes son necesarias para el diagnóstico y seguimiento de diversas condiciones médicas. Sin embargo, para aprovechar al máximo la información que estas imágenes proporcionan, es esencial realizar un preprocesamiento adecuado y una eliminación eficiente del ruido.

En este trabajo de grado, se aplican varias técnicas de preprocesamiento de imágenes para preparar los datos de tomografía computarizada *(CT)* del área abdominal antes de utilizarlos en los modelos de segmentación. Estas técnicas son esenciales para garantizar la consistencia y calidad de los datos de entrada para los algoritmos de segmentación. A continuación, se describen

detalladamente las técnicas de preprocesamiento utilizadas:

5.2.1. Preprocesamiento y Eliminación de Ruido en Imágenes de *CT*

El propósito del preprocesamiento en imágenes de *CT* es múltiple. En primer lugar, busca mejorar la calidad de la imagen, que puede verse comprometida por diversos factores como el ruido inherente al proceso de adquisición de imágenes o las interferencias causadas por elementos externos, como la camilla en la que se acuesta el paciente. Además, el preprocesamiento tiene como objetivo preparar las imágenes para su posterior análisis y procesamiento, en particular para las técnicas de segmentación avanzadas utilizando modelos de aprendizaje profundo como *U-Net*, *U-Net* con *Backbone VGG16*, *U-Net* con *Backbone ResNet50*, *SegNet*, *FPN* con *Backbone ResNet50* y *LinkNet* con *Backbone VGG16*. Estos modelos requieren imágenes de alta calidad y uniformidad para funcionar con precisión, lo que hace que el preprocesamiento sea un paso indispensable en el flujo de trabajo de análisis de imágenes médicas.

Dentro de esta subsección, abordaremos específicamente la eliminación de la camilla en las imágenes de *CT* y residuos adicionales como cobijas, mantas, etc. La presencia de estos elementos puede introducir artefactos que distorsionan los resultados de la segmentación, llevando a interpretaciones erróneas o inexactas. Para resolver este problema se planteó un código en *Python* para la modificación total del *dataset* agregando una máscara que analiza el 50 % de las filas de la matriz (parte superior de todas las máscaras).

La Función *clean_noise_from_mask()*: Esta función tiene como objetivo limpiar ruido de una matriz de máscara. La matriz de la máscara es una matriz bidimensional que donde la camilla tiene el valor 1 y el fondo *(background)* tiene el valor de 0, esta función recibe a `mask_data` que es la matriz de la máscara representada como un *array* de *NumPy* y el porcentaje de filas iniciales a considera con `m_percentage` este es un valor flotante que indica el porcentaje de la máscara que se procesará para la eliminación de ruido.

La función itera sobre las filas de la matriz de máscara desde un índice calculado en base al porcentaje proporcionado hacia atrás. Luego, llama a la función `process_columns()` para procesar las columnas de cada fila en esa sección de la máscara. Esta función examina las sumas de columnas y reemplaza los valores en columnas específicas si se cumplen ciertas condiciones (columnas con sumas mayores a 0 y menores o iguales a 3, considerando valores iguales a 1).

La función *verify_noise_removal()*, es una duplicación *clean_noise_from_mask()*. Ambas funciones tienen la misma estructura y utilizan la misma lógica para procesar la matriz de máscara. Sin embargo, este proceso realiza una segunda validación para evitar algún ruido adicional.

La función *process_columns()* se encarga de procesar las columnas de la matriz de máscara hasta una fila específica (*row_limit*). Para cada columna, calcula la suma de los elementos que son

iguales a 1 en la sección de la máscara hasta la fila *row_limit*. Luego, identifica las columnas que cumplen ciertas condiciones (suma mayor a 0 y menor o igual a 3) y reemplaza los valores en esas columnas con 0 si originalmente eran 1.

Una vez resuelto el ruido del 50 % superior de la máscara usamos la función *apply_mask_to_image()* que utiliza todo el fondo de la máscara igual a cero y lo aplica sobre la imagen con un valor de -1042, de esta forma ya no se tiene ningún ruido en la imagen relacionado con camillas, cobijas, mantas, etc. (véase Imagen 5.3). La elección entre guardar imágenes o máscaras se determina por el parámetro *'select_item'*. El código de eliminación de la camilla se encuentra disponible en el repositorio de Github con el nombre `02 - denoise.ipynb` en la carpeta *Notebooks*. A continuación, se visualiza la imagen del paciente cero sin camilla.

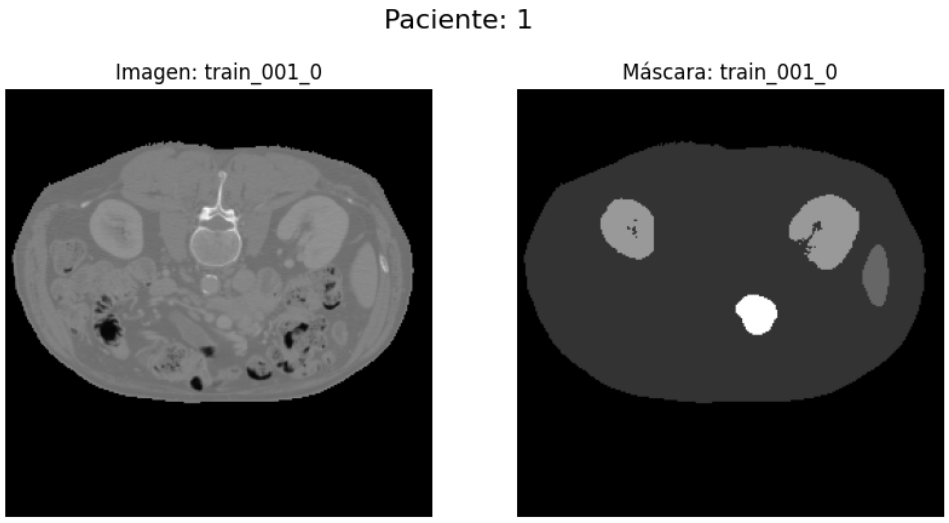

Figura 5.3: Imagen y mascara sin camilla del paciente cero. Fuente: *Autores*.

5.2.2. Ampliación de Datos (*Data Augmentation*)

En el ámbito del aprendizaje profundo (*Deep Learning*), la técnica conocida como *data augmentation* se convierte en un componente esencial para enriquecer tanto la cantidad como la variabilidad de los conjuntos de datos utilizados en el entrenamiento de modelos. Este aspecto es particularmente crítico en campos donde la adquisición de datos se presenta como una tarea costosa o restringida, como sucede comúnmente en el ámbito de las imágenes médicas. En nuestra investigación, la implementación de *Data Augmentation* se lleva a cabo haciendo uso de la librería *imgaug* de *Python*, la cual proporciona una amplia gama de técnicas para la mejora de imágenes.

Dentro del contexto específico de nuestro proyecto, nos centramos principalmente en las transformaciones geométricas proporcionadas por *imgaug*, tales como rotaciones, traslaciones, escalado y volteo, con el propósito de generar variaciones en las imágenes de tomografía computacional *(CT)*.

La ampliación de datos se conceptualiza como un proceso dinámico que busca crear nuevas instancias de entrenamiento a través de la aplicación de transformaciones específicas a las imágenes existentes. Estas transformaciones, aunque no alteran la información de la clase, introducen variabilidad, desempeñando un papel relevante al mejorar la capacidad del modelo para generalizar. Este enriquecimiento artificial de la diversidad de datos contribuye significativamente a la mejora de la generalización de los modelos resultantes.

Es importante destacar que el proceso de *data augmentation* debe ser ejecutado con cautela. Es necesario encontrar un equilibrio óptimo en términos de cantidad y tipo de aumentación para evitar el *sobreajuste* del modelo. Este cuidado garantiza que las imágenes modificadas mantengan su relevancia y realismo en relación con la problemática médica que se pretende abordar.

El código implementado para la *data augmentation* se encuentra disponible para su revisión en el repositorio de *GitHub*, cuyo enlace se proporciona en el siguiente *link*: GitHub. Para proporcionar una visión tangible de los resultados obtenidos a través de esta técnica, se presenta a continuación la visualización de la imagen correspondiente al paciente inicial en nuestro conjunto de datos.

Las técnicas de *Data Augmentation* que se pueden utilizar para la segmentación de imágenes de tomografía computacional son:

1. Rotación:
   - Las imágenes de tomografía pueden rotarse en diferentes ángulos para simular variaciones en la posición del paciente durante la exploración.
   - La rotación puede ser en 2D o 3D, dependiendo de la naturaleza del volumen de datos.
2. Zoom y Escala:
   - La aplicación de zoom y escala simula variaciones en la resolución espacial de las imágenes.
   - Puede ayudar al modelo a adaptarse a diferentes niveles de detalle presentes en diferentes estudios de tomografía.
3. Deformación Elástica:
   - Introduce deformaciones locales en la imagen para simular posibles distorsiones anatómicas o artefactos en las imágenes de tomografía.
4. Flip Horizontal/Vertical:

- Reflejar horizontal o verticalmente las imágenes ayuda a enseñar al modelo sobre simetrías y mejora su capacidad para reconocer estructuras desde diferentes perspectivas.

5. Ajustes de Contraste y Brillo:

   - Variaciones en el contraste y el brillo replican las diferencias en la iluminación durante la adquisición de imágenes.

6. Corte Aleatorio:

   - Realizar cortes aleatorios en las imágenes en 3D para imitar variaciones en la orientación y posición del objeto escaneado.

7. Adición de Ruido:

   - Introducir ruido artificial en las imágenes ayuda al modelo a ser más robusto frente a posibles imperfecciones en los datos reales.

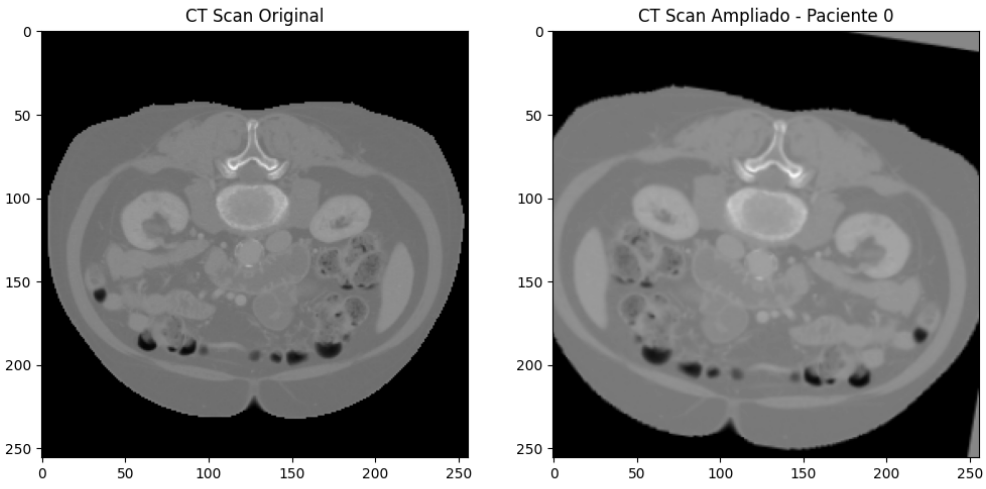

Figura 5.4: Imagen del paciente cero, aumentada. Fuente: *Autores*.

Como se observa en la imagen, en la parte izquierda tenemos la imagen original sin ruido y en la parte derecha tenemos la imagen desplazada y escalada. Este código en Python, basado en la librería *imgaug*, ofrece una solución efectiva para aplicar transformaciones geométricas a las imágenes de tomografía, generando así variaciones artificiales que mejoran la capacidad de generalización

del modelo.

La función `apply_data_augmentation()` toma como entrada una lista de rutas de archivos que contienen imágenes de tomografía en formato numpy (`.npy`). En la sección de anexos, se proporciona una explicación detallada del código 8.1.

La función comienza por crear un directorio de salida, denominado *augmented_images*, donde se almacenarán las imágenes ampliadas. Luego, itera sobre la lista de imágenes originales de tomografía, aplicando una secuencia de transformaciones definida por el objeto *augmenter*. Estas transformaciones incluyen rotaciones en el rango de -10 a 10 grados, volteos horizontales con una probabilidad del 50 %, y recortes de hasta el 10 % de la imagen.

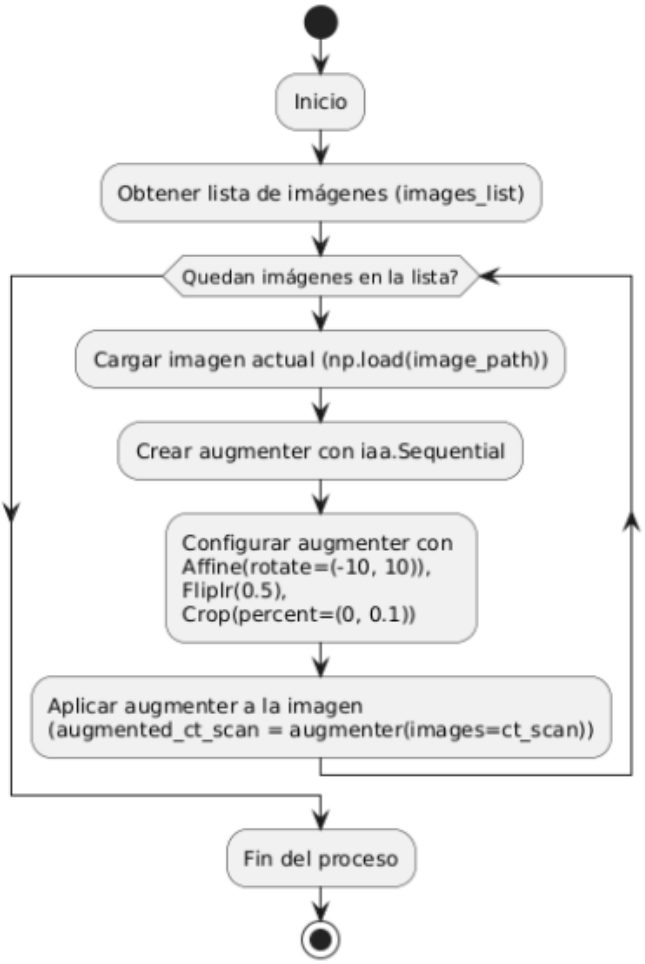

Figura 5.5: Diagrama de flujo - *Data Augmentation*. Fuente: *Autores*.

Posteriormente, la imagen resultante de la ampliación se guarda en formato *.npy* en el directorio de salida con un nombre de archivo específico. Además, se genera una visualización comparativa entre la imagen original y la imagen ampliada, mostrándolas lado a lado. Estas visualizaciones se guardan como archivos de imagen en formato *.png*.

Esta implementación facilita la generación eficiente y automatizada de un conjunto de datos ampliado, enriqueciendo así la diversidad de muestras disponibles para el entrenamiento de modelos de segmentación en tomografía computacional. Este enfoque se integra de manera efectiva en la sección de *Data Augmentation*, contribuyendo a mejorar la robustez y generalización del modelo resultante.

5.3. *Backbones*

En el campo de la segmentación semántica, los *backbones* juegan un papel vital como base para la extracción de características de las imágenes. Estas redes preentrenadas, como *VGG16* y *ResNet50*, se encargan de capturar patrones y características de bajo y alto nivel de las imágenes, lo que facilita la tarea de segmentación al proporcionar una representación robusta y detallada.

Utilizar *backbones* preentrenados permite aprovechar el conocimiento adquirido de grandes conjuntos de datos como *ImageNet*, mejorando la precisión y eficiencia de los modelos de segmentación semántica al adaptarlos a tareas específicas con menor esfuerzo de entrenamiento. Además, la capacidad de estos *backbones* para manejar estructuras complejas y profundas en las imágenes contribuye significativamente a la precisión y efectividad de las arquitecturas de segmentación modernas.

5.3.1. VGG16

VGG16 es un modelo de aprendizaje profundo muy utilizado para tareas de reconocimiento de imágenes *(véase imagen 5.6)*. Pertenece a la familia de *VGG*, conocida por su estructura simple y profunda. *VGG16* contiene 16 capas de convolución, lo que permite una extracción de características robusta y detallada de las imágenes [36].

Este modelo se entrena previamente en el conjunto de datos *ImageNet*, que incluye millones de imágenes etiquetadas, y puede usarse para *transfer learning*. Esto permite reutilizar el modelo preentrenado y ajustarlo a tareas específicas con datos adicionales .

Algunos parámetros importantes para trabajar con *VGG16* incluyen `include_top`, que indica si se deben incluir las tres capas totalmente conectadas en la parte superior de la red, `weights`, que especifica si se deben utilizar pesos preentrenados o inicialización aleatoria, `input_tensor`, que es un tensor de *Keras* opcional para usar como entrada de imagen para el modelo, `input_shape`, que define la forma de entrada cuando `include_top` es `False`, y `pooling`, que especifica el modo de agrupamiento para la extracción de características. `pooling` puede ser `None`, `avg` (agrupamiento

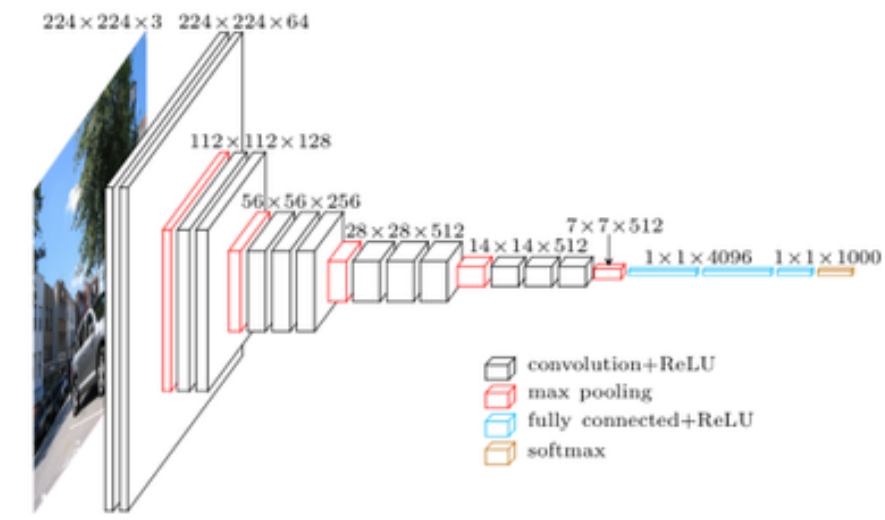

Figura 5.6: Backbone VGG16. Fuente: *[17]*.

promedio global) o `max` (agrupamiento máximo global).

5.3.2. Resnet50

ResNet50 es un modelo de aprendizaje profundo muy utilizado para tareas de reconocimiento de imágenes. Pertenece a la familia de *ResNet*, conocida por su marco de aprendizaje residual, que permite entrenar redes neuronales muy profundas al resolver el problema del gradiente desaparecido. *ResNet50* cuenta con 50 capas de convolución, lo que le permite extraer características de alto nivel de las imágenes [15].

Este modelo se entrena previamente en el conjunto de datos *ImageNet*, que contiene millones de imágenes etiquetadas, y puede utilizarse para aprendizaje por transferencia, lo que significa que se puede reutilizar el modelo preentrenado y adaptarlo a nuevas tareas específicas con datos adicionales.

Algunos parámetros importantes para trabajar con *ResNet50* incluyen `include_top`, que indica si se debe incluir la capa totalmente conectada en la parte superior de la red, `weights`, que especifica si se deben utilizar pesos preentrenados o inicialización aleatoria, `input_shape`, que define la forma de entrada cuando `include_top` es `False`, y `pooling`, que especifica el modo de agrupamiento para la extracción de características.

5.4. Arquitecturas de segmentación de *Deep Learning*

Las arquitecturas de *Deep Learning* para la segmentación de imágenes de *CT* abdominal han demostrado ser herramientas poderosas en la práctica médica. Estas arquitecturas aportan precisión, eficiencia y consistencia al proceso de segmentación, lo que tiene un impacto significativo en el diagnóstico, el tratamiento y la investigación médica en el campo de la radiología y la medicina moderna. En esta sección, se explorarán diversas arquitecturas de segmentación basadas en aprendizaje profundo, analizando sus componentes clave y las ventajas que ofrecen para la segmentación de imágenes médicas.

5.4.1. Modelo *U-Net*

La arquitectura *U-Net* fue introducida por Olaf Ronneberger, Philipp Fischer y Thomas Brox del Departamento de Ciencias de la Computación y el Centro BIOSS para Estudios de Señalización Biológica de la Universidad de Freiburg, Alemania. Esta red se diseñó específicamente para tareas de segmentación de imágenes biomédicas y se basa en el uso intensivo de la ampliación de datos para aprovechar más eficazmente las muestras anotadas disponibles [34].

La arquitectura consta de un camino de contracción para capturar el contexto y un camino de expansión simétrico que permite una localización precisa. La red *U-Net* está compuesta por dos partes principales: el camino de contracción y el camino de expansión.

El camino de contracción sigue la arquitectura típica de una red convolucional. Consiste en la aplicación repetida de dos convoluciones 3×3 (convoluciones sin relleno), cada una seguida de una unidad lineal rectificada ($ReLU$) y una operación de agrupamiento máximo 2×2 con un paso de 2 para la reducción de dimensionalidad. En cada paso de reducción de dimensionalidad, se duplica el número de canales de características *(véase imagen 5.7)*.

Cada paso en el camino de expansión consiste en una ampliación del mapa de características seguida de una convolución 2×2 ("up-convolution") que reduce a la mitad el número de canales de características. Luego, se concatena con el mapa de características recortado correspondiente del camino de contracción y se aplican dos convoluciones 3×3, cada una seguida de una $ReLU$. El recorte es necesario debido a la pérdida de píxeles en el borde en cada convolución. En la capa final, se utiliza una convolución 1×1 para mapear cada vector de características de 64 componentes al número deseado de clases. En total, la red tiene 23 capas convolucionales [34].

Para permitir un mosaico sin costuras del mapa de segmentación de salida, es importante seleccionar el tamaño de mosaico de entrada de manera que todas las operaciones de agrupamiento máximo 2×2 se apliquen a una capa con un tamaño x e y par. La red no tiene capas completamente conectadas y solo utiliza la parte válida de cada convolución, es decir, el mapa de segmentación solo contiene los píxeles para los que el contexto completo está disponible en la imagen de entrada.

Dado que en nuestras tareas hay muy pocos datos de entrenamiento disponibles, utilizamos una amplia ampliación de datos mediante la aplicación de deformaciones elásticas a las imágenes de entrenamiento disponibles. Esto permite que la red aprenda invariancia a tales deformaciones, sin necesidad de ver estas transformaciones en el corpus de imágenes anotadas. Esto es particularmente importante en la segmentación biomédica, ya que la deformación solía ser la variación más común en el tejido y las deformaciones realistas se pueden simular de manera eficiente.

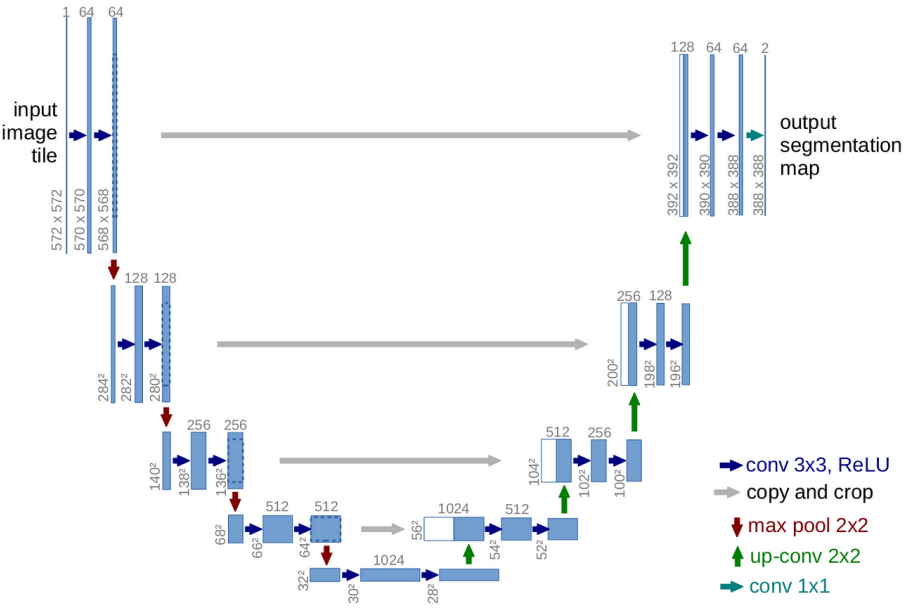

Figura 5.7: Arquitectura *U-net*. Fuente: *[34]*.

La función de energía se calcula mediante un *softmax* por píxel sobre el mapa de características final combinado con la función de pérdida de entropía cruzada. Para permitir la segmentación de bordes en tareas de segmentación celular, proponemos el uso de una pérdida ponderada, donde las etiquetas de fondo separadoras entre células que tocan obtienen un gran peso en la función de pérdida.

La red se entrena con la implementación de descenso de gradiente estocástico de *Caffe*. Debido a las convoluciones sin relleno, la imagen de salida es más pequeña que la entrada por un ancho de borde constante. Para minimizar la sobrecarga y hacer el máximo uso de la memoria de la *GPU*, favorecemos los mosaicos de entrada grandes sobre un tamaño de lote grande y, por lo tanto, reducimos el lote a una sola imagen. En consecuencia, utilizamos un momento alto (0.99) para que un gran número de muestras de entrenamiento vistas previamente determinen la actualización en el paso de optimización actual.

La arquitectura *U-Net* logra un rendimiento muy bueno en aplicaciones de segmentación biomédica muy diferentes. Gracias a la ampliación de datos con deformaciones elásticas, solo necesita muy pocas imágenes anotadas y tiene un tiempo de entrenamiento muy razonable de solo 10 horas en una *GPU NVidia Titan* (6 GB). Proporcionamos la implementación completa basada en *Caffe* y las redes entrenadas para facilitar su aplicación en muchas otras tareas.

5.4.2. Modelo *SegNet*

La segmentación semántica es una tarea fundamental en la visión por computador, con aplicaciones que van desde la comprensión de escenas y la inferencia de relaciones de soporte entre objetos hasta la conducción autónoma. Métodos tempranos que dependían de pistas de visión de bajo nivel han sido rápidamente superados por algoritmos populares de aprendizaje automático.

En particular, el aprendizaje profundo ha visto un enorme éxito últimamente en el reconocimiento de dígitos manuscritos, el habla, la categorización de imágenes completas y la detección de objetos en imágenes. Ahora, hay un interés activo en el etiquetado semántico de píxeles. Nuestra motivación para diseñar *SegNet* surge de la necesidad de mapear características de baja resolución a la resolución de entrada para la clasificación de píxeles. Este mapeo debe producir características que sean útiles para la localización precisa de los bordes.

SegNet es una arquitectura de red neuronal convolucional profunda y completamente entrenable para la segmentación semántica píxel a píxel. Esta arquitectura consta de una red codificadora, una red decodificadora correspondiente y una capa de clasificación píxel a píxel *(véase imagen 5.8)*.

La red codificadora de *SegNet* es topológicamente idéntica a las 13 capas convolucionales de la red *VGG16*, que ha sido ampliamente adoptada para la clasificación de imágenes. En *SegNet*, se eliminan las capas completamente conectadas de *VGG16*, lo que hace que la red codificadora de *SegNet* sea significativamente más pequeña y más fácil de entrenar que muchas otras arquitecturas recientes [4].

El componente clave de *SegNet* es la red decodificadora, que consiste en una jerarquía de decodificadores, uno correspondiente a cada codificador. Los decodificadores usan los índices de *max-pooling* computados en el paso de *max-pooling* del codificador correspondiente para realizar una interpolación no lineal. Este proceso elimina la necesidad de aprender a realizar la interpolación. Los mapas resultantes son dispersos y luego se convolucionan con filtros entrenables para producir mapas de características densos.

Después de la red decodificadora, *SegNet* usa una capa de *softmax* para clasificar cada píxel en una de las clases de segmentación. Esta capa convierte las características en probabilidades de

pertenencia a cada clase, asignando una etiqueta de clase a cada píxel de la imagen.

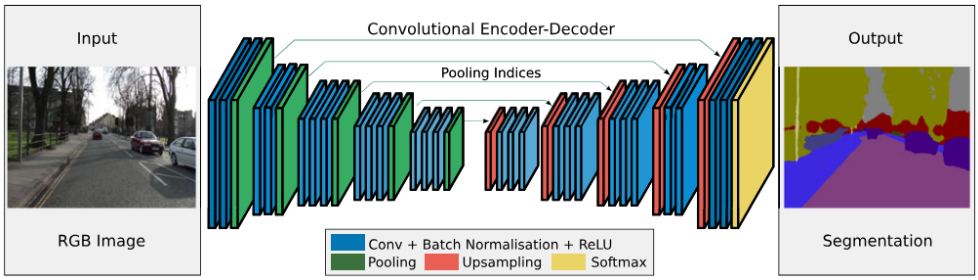

Figura 5.8: Arquitectura *SegNet*. Fuente: *[4]*.

La novedad de *SegNet* radica en la forma en que la red decodificadora realiza la interpolación de su mapa de características de entrada de menor resolución. Específicamente, la decodificadora utiliza índices de *pooling* calculados en el paso de *max-pooling* del codificador correspondiente para realizar una interpolación no lineal. Esto elimina la necesidad de aprender a interpolar. Los mapas interpolados son dispersos y luego se convolucionan con filtros entrenables para producir mapas de características densos.

Comparado con arquitecturas como la *FCN* (Fully Convolutional Network) y *DeepLab-LargeFOV*, *SegNet* muestra una eficiencia significativa en términos de memoria y tiempo computacional durante la inferencia. Aunque todas estas arquitecturas tienen redes codificadoras idénticas, basadas en *VGG16*, difieren en la forma de la red decodificadora, el entrenamiento y la inferencia. *SegNet* fue diseñado para aplicaciones de comprensión de escenas, siendo eficiente tanto en términos de memoria como de tiempo de cómputo durante la inferencia. Además, *SegNet* es considerablemente más pequeño en el número de parámetros entrenables que otras arquitecturas competidoras y puede ser entrenado de extremo a extremo utilizando descenso de gradiente estocástico.

SegNet ha sido evaluado en tareas de segmentación de escenas de carretera y segmentación de escenas interiores *SUN RGB-D*. Estas evaluaciones cuantitativas muestran que *SegNet* ofrece un buen rendimiento con tiempos de inferencia competitivos y el uso de memoria más eficiente en comparación con otras arquitecturas. *SegNet* también proporciona una implementación en *Caffe* y una demostración web.

SegNet representa una solución práctica y eficiente para la segmentación semántica píxel a píxel. Su diseño se enfoca en la eficiencia tanto en términos de memoria como de tiempo computacional, lo que lo hace adecuado para aplicaciones de comprensión de escenas. Además, la eliminación de la necesidad de aprender a realizar la interpolación simplifica significativamente el proceso de entrenamiento, haciendo que *SegNet* sea una opción atractiva para tareas de segmentación en tiempo real.

5.4.3. Modelo *FPN (Feature Pyramid Network)*

Las pirámides de características son un componente básico en los sistemas de reconocimiento para detectar objetos a diferentes escalas. Sin embargo, los detectores de objetos recientes basados en *deep learning* han evitado las representaciones piramidales en parte porque son intensivas en computación y memoria [24].

Este artículo explora la jerarquía piramidal multi-escala inherente a las redes neuronales convolucionales profundas para construir pirámides de características con un costo marginal adicional. Se desarrolla una arquitectura de arriba hacia abajo con conexiones laterales para construir mapas de características semánticas de alto nivel a todas las escalas. Esta arquitectura, denominada *Feature Pyramid Network* (FPN), muestra una mejora significativa como extractor de características genéricas en varias aplicaciones [24].

El objetivo de la *Feature Pyramid Network* es aprovechar la jerarquía piramidal de características de una *ConvNet*, que tiene semántica desde niveles bajos hasta altos, y construir una pirámide de características con semántica de alto nivel en todas las escalas. La construcción de la pirámide implica una vía de abajo hacia arriba, una vía de arriba hacia abajo y conexiones laterales.

La vía de abajo hacia arriba es la computación *feed-forward* de la *ConvNet* base, que calcula una jerarquía de características que consiste en mapas de características a varias escalas con un paso de escala de 2. Hay muchas capas que producen mapas de salida del mismo tamaño y estas capas se consideran en la misma etapa de red.

Para la pirámide de características, se define un nivel de pirámide para cada etapa. Se elige la salida de la última capa de cada etapa como el conjunto de referencia de mapas de características, que se enriquecerá para crear la pirámide. Esta elección es natural ya que la capa más profunda de cada etapa debería tener las características más fuertes *(véase imagen 5.9)*.

Específicamente, para los *ResNets* se utilizan las activaciones de características producidas por el último bloque residual de cada etapa. Se denotan las salidas de estos últimos bloques residuales como {C2, C3, C4, C5} para las salidas de *conv2*, *conv3*, *conv4* y *conv5*, y se observa que tienen pasos de {4, 8, 16, 32} píxeles con respecto a la imagen de entrada.

La vía de arriba hacia abajo genera características de mayor resolución mediante el muestreo espacial de mapas de características más burdos pero semánticamente más fuertes de niveles de pirámide superiores. Estas características se mejoran con características de la vía de abajo hacia arriba a través de conexiones laterales. Cada conexión lateral fusiona mapas de características del mismo tamaño espacial de la vía de abajo hacia arriba y la vía de arriba hacia abajo. El mapa de características de la vía de abajo hacia arriba es de semántica de nivel más bajo, pero sus activaciones están más precisamente localizadas ya que fue subsampleado menos veces.

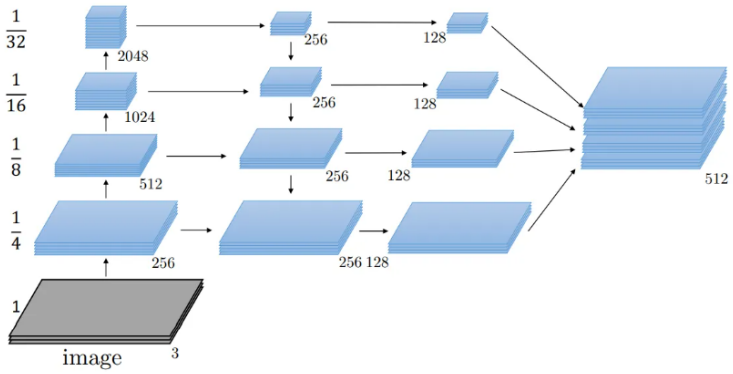

Figura 5.9: Arquitectura *FPN*. Fuente: *[9]*.

La Figura 5.9 muestra el bloque de construcción que construye los mapas de características de arriba hacia abajo. Con un mapa de características de resolución más burda, se aumenta la resolución espacial por un factor de 2 (usando el muestreo más cercano por simplicidad). El mapa de características aumentado se fusiona con el mapa de características correspondiente de la vía de abajo hacia arriba (que pasa por una capa de convolución 1×1 para reducir las dimensiones del canal) mediante una adición elemento a elemento. Este proceso se itera hasta que se genera el mapa de resolución más fina.

Para iniciar la iteración, simplemente se adjunta una capa de convolución 1×1 en C5 para producir el mapa de resolución más burda. Finalmente, se adjunta una convolución 3×3 en cada mapa fusionado para generar el mapa de características final, lo que reduce el efecto de aliasing del muestreo. Este conjunto final de mapas de características se llama {P2, P3, P4, P5}, correspondiente a {C2, C3, C4, C5} que son respectivamente del mismo tamaño espacial.

Debido a que todos los niveles de la pirámide usan clasificadores/regresores compartidos como en una pirámide de imágenes tradicional, se fija la dimensión de las características (número de canales, denotado como d) en todos los mapas de características. En este artículo se fija $d = 256$ y así todas las capas de convolución adicionales tienen salidas de 256 canales. No hay no-linealidades en estas capas adicionales, las cuales se ha encontrado empíricamente que tienen impactos menores.

La simplicidad es central en el diseño y se ha encontrado que el modelo es robusto a muchas opciones de diseño. Se ha experimentado con bloques más sofisticados (por ejemplo, usando bloques residuales multinivel como las conexiones) y se ha observado mejoras marginales. El diseño de mejores módulos de conexión no es el foco de este artículo, por lo que se opta por el diseño simple descrito anteriormente.

El método es una solución genérica para construir pirámides de características dentro de *ConvNets*. Se adopta el método en *RPN* para la generación de propuestas de cuadros delimitadores y

en *Fast R-CNN* para la detección de objetos.

RPN es un detector de objetos agnóstico a clases basado en ventanas deslizantes. En el diseño original de *RPN*, una pequeña subred se evalúa en ventanas deslizantes densas de 3×3, sobre un mapa de características convolucional de una sola escala, realizando la clasificación binaria de objeto/no-objeto y la regresión de cuadros delimitadores. Esto se realiza mediante una capa de convolución 3×3 seguida de dos convoluciones 1×1 hermanas para la clasificación y la regresión, a lo que se refiere como una cabeza de red.

El criterio de objeto/no-objeto y el objetivo de regresión del cuadro delimitador se definen con respecto a un conjunto de cuadros de referencia llamados *anchors*. Se adapta *RPN* reemplazando el mapa de características de una sola escala con la FPN. Se adjunta una cabeza de diseño similar (convolución de 3×3 y dos convoluciones 1×1 hermanas) a cada nivel en la pirámide de características.

Fast R-CNN es un detector de objetos basado en regiones en el cual se utiliza *RoI pooling* para extraer características. Fast R-CNN se realiza más comúnmente en un mapa de características de una sola escala. Para usarlo con la FPN, se necesita asignar *RoIs* de diferentes escalas a los niveles de la pirámide.

Se ha presentado un marco limpio y simple para construir pirámides de características dentro de *ConvNets*. El método muestra mejoras significativas sobre varios puntos de referencia fuertes y ganadores de competencias. Proporciona una solución práctica para la investigación y aplicaciones de pirámides de características, sin la necesidad de computar pirámides de imágenes.

5.4.4. Modelo *LinkNet*

A pesar de los avances en las redes neuronales profundas, muchas de las arquitecturas actuales son demasiado grandes y lentas para su uso en dispositivos embebidos. En este contexto, se presenta *LinkNet*, una arquitectura de red neuronal profunda diseñada para aprender de manera eficiente sin un incremento significativo en el número de parámetros.

La arquitectura de *LinkNet* se basa en un par *encoder-decoder*, similar a las técnicas inspiradas en autoencoders. La innovación principal de *LinkNet* radica en la forma en que se enlaza cada capa del *encoder* con su correspondiente capa del *decoder*, preservando la información espacial que generalmente se pierde durante el proceso de *downsampling* en el *encoder (véase imagen 5.10)*.

LinkNet utiliza *ResNet18* como *encoder*, una red relativamente ligera con solo 11.5 millones de parámetros y 21.2 GFLOPs para procesar una imagen de resolución $3 \times 640 \times 360$. Esta elección de

encoder permite que la red sea más eficiente en comparación con otras arquitecturas de segmentación como *VGG16* y *ResNet101*, que tienen muchos más parámetros y operaciones.

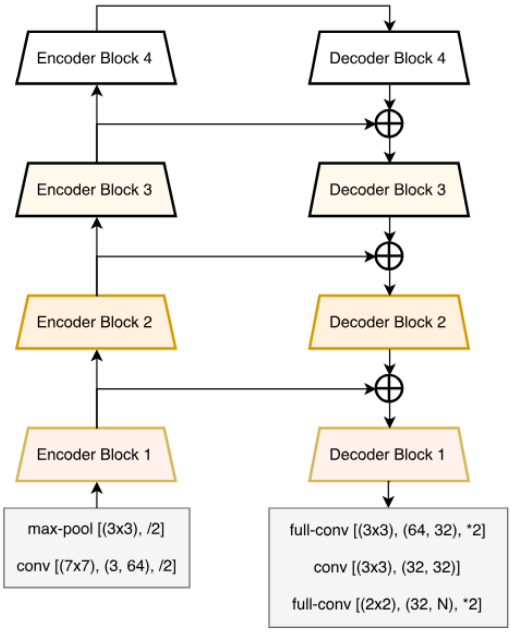

Figura 5.10: Arquitectura LinkNet. Fuente: *[6]*.

LinkNet está compuesta por varios bloques *encoder* y *decoder*. Cada bloque del *encoder* realiza una convolución inicial con un tamaño de kernel de 7×7 y un *stride* de 2, seguida de un *max-pooling* espacial en un área de 3×3 con un *stride* de 2. Posteriormente, utiliza bloques residuales para extraer características.

Cuadro 5.1: Mapas de características de entrada y salida

Bloque	*Encoder*	*Decoder*	m	n
1	64	64	64	64
2	64	128	128	64
3	128	256	256	128
4	256	512	512	256

La información de entrada de cada capa del *encoder* se pasa directamente a la salida de su correspondiente capa del *decoder*, mejorando así la precisión y reduciendo el tiempo de procesamiento. La Tabla 5.1 detalla los mapas de características utilizados en cada uno de estos bloques.

LinkNet muestra un rendimiento superior en términos de precisión y velocidad en comparación con otras arquitecturas de segmentación. En el conjunto de datos *Cityscapes*, *LinkNet* logra una IoU de clase de 76.4 % y una $iIoU$ de 58.6 %, superando a modelos como *SegNet*, *ENet* y *Deep-Lab CRF*. Los resultados en el conjunto de datos *CamVid* también demuestran la eficacia de *LinkNet*, con un rendimiento superior en las métricas de IoU y $iIoU$.

Capítulo 6

Resultados

6.1. Validación de los modelos obtenidos

Se realizó una exhaustiva comparación de las arquitecturas *U-Net*, *SegNet*, *FPN* y *LinkNet*, empleando diversos *Backbones* como *VGG16* y *ResNet50*. Para evaluar el desempeño de los modelos, se utilizaron múltiples métricas, incluyendo *accuracy*, IoU, *Dice*, $F1$, $F2$, *precision* y *recall*. Inicialmente, se había considerado implementar un criterio de parada temprana (*early stopping*), sin embargo, se optó por establecer un número fijo de 250 épocas para el entrenamiento de todas las arquitecturas.

Además, se generaron dos gráficas para ilustrar el rendimiento de los modelos: una mostrando la *accuracy* tanto en entrenamiento como en validación, y otra representando la función de pérdida en ambas fases. Se llevó a cabo una validación indirecta utilizando las primeras 30 imágenes del conjunto de pruebas, con el fin de evaluar visualmente las predicciones de segmentación semántica en comparación con las máscaras reales. Este análisis permitió verificar la coherencia de las predicciones a nivel de órganos, asegurando su validez frente a las máscaras de referencia.

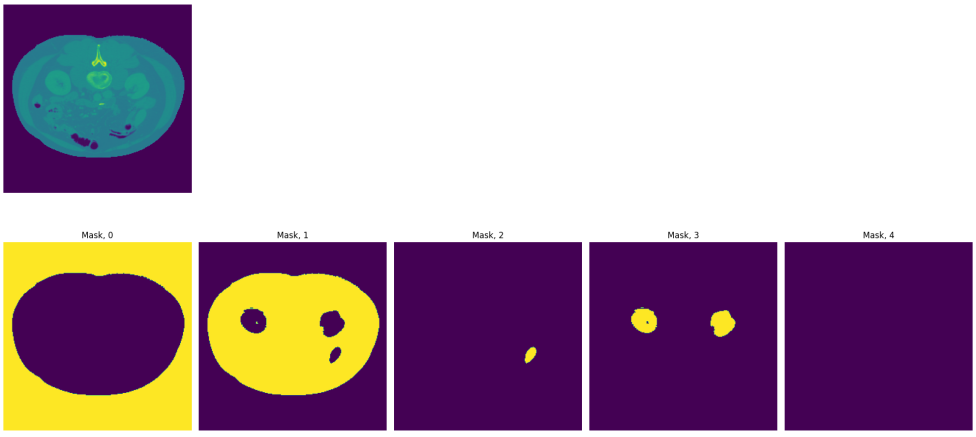

Figura 6.1: Gráfico de imágenes y mascaras en *One-Hot encoding*. Fuente: *Autores*.

Para la construcción del *dataset* se consideró un tamaño de lote *(BATCH SIZE)* de 32, recordando que el dataset contiene 5 clases que van de 0 a 4, donde 0 es el fondo *BACKGROUND* de

la imagen, como se mencionó en la sección 5.2, las imágenes y mascaras se encuentran en formato *numpy (*.npy)* de tamaño $256 \times 256 \times 1$, al tener 1 solo canal se tiene que deshabilitar los pesos de los decodificadores en las arquitecturas que tienen *Backbone*.

Las máscaras son llevadas a una representación *One-Hot encoding* y en general se puede decir que cada imagen tiene un tensor de tamaño (256, 256, 1) y las máscaras con un tensor de tamaño (256, 256, 5) *(véase imagen 6.1)* y el tamaño del tensor resultante del dataset fue [32, 256, 256, 1] para imágenes y [32, 256, 256, 5] para las máscaras incluyendo el tamaño del lote.

Para la seleccionar y configurar diferentes las arquitecturas de segmentación, específicamente *U-Net*, *FPN* y *Linknet*, se utilizó la librería `segmentation_models` [21] y un selector en un condicional el cual dependiendo del valor de la variable escoge uno de estos modelos *(véase código anexo 8.1)*.

Se establece un punto de control *checkpoint* para guardar los archivos `*.Keras`, esta ruta se define utilizando la carpeta de salida de *Kaggle*, el nombre de la arquitectura y una extensión `.keras`. Adicional se configura la lista de métricas que se van a utilizar durante el entrenamiento del modelo.

para la Compilación del Modelo, se hace utilizando el modelo seleccionado y el optimizador *Adam*, que se establece con una tasa de aprendizaje (`learning_rate`) de 0.001. *Adam* es un optimizador basado en el descenso de gradiente estocástico que ajusta automáticamente la tasa de aprendizaje de cada parámetro.

La función de pérdida (`loss`) utilizada es `categorical_crossentropy`, adecuada para problemas de clasificación con múltiples clases. Finalmente, `model.summary()` se llama para imprimir un resumen de la arquitectura del modelo *(véase Sección de anexos 8.1)*. Esto incluye detalles sobre cada capa del modelo, el tipo de capa, la forma de salida de cada capa, el número de parámetros y otros detalles relevantes.

A continuación, se muestra un análisis comparativo de varias arquitecturas de segmentación utilizadas en el entrenamiento de modelos, con información específica sobre el número de épocas, el tiempo por época y el tiempo total de entrenamiento *(véase tabla 6.1)*.

Basándonos exclusivamente en las variables de tiempo por época y tiempo total de entrenamiento, la mejor arquitectura es *U-Net* con *Backbone* ResNet50, ya que tiene el menor tiempo por época (29 segundos) y el menor tiempo total de entrenamiento (2.01 horas). Esto sugiere que es la opción más eficiente en términos de tiempo de entrenamiento, lo cual es fundamental en escenarios donde la rapidez en el desarrollo y la iteración de modelos es importante.

Es fundamental evaluar el rendimiento de las arquitecturas considerando métricas como *accuracy*, IoU_{score}, $dice_{score}$ y calidad de segmentación, para determinar la mejor opción en aplicaciones prácticas. Utilizamos métricas de segmentación y clasificación en un sistema de puntaje no pon-

No.	Arq.	Backbone	Épocas	tiempo por época (s)	tiempo total (h)
1	*U-Net*		250	67	4,65
2	*SegNet*		250	55	3,81
3	*U-Net*	VGG16	250	34	2,36
4	*U-Net*	ResNet50	250	29	2,01
5	*FPN*	ResNet50	250	63	4,37
6	*LinkNet*	VGG16	250	34	2,36

Cuadro 6.1: Características generales y tiempos de entrenamiento de las arquitecturas

derado, donde el puntaje se obtiene sumando todas las métricas, exceptuando la pérdida, que se incluye mediante su complemento *(véase Ec. 6.1)*.

$$Puntaje = \frac{accuracy + dice + F2 + IoU + (1 - loss) + precision + recall}{7} \tag{6.1}$$

Este enfoque es válido ya que todas las métricas tienen valores entre 0 y 1, excepto la función de pérdida CEE, donde se utiliza el complemento $1 - loss$. Para asegurar que la fórmula está correctamente planteada y que los términos no se cancelan, desarrollamos la fórmula del puntaje en el anexo 8.2. Allí se muestra que los denominadores son distintos, evitando agrupaciones o cancelaciones de métricas. Esto confirma que la expresión es irreducible y que corresponde a las métricas referenciadas.

6.1.1. Análisis de resultados y comparación

6.1.1.1. Modelo U-Net

La arquitectura inicial seleccionada para este estudio fue *U-Net*. A continuación, se presenta un análisis exhaustivo de las métricas obtenidas durante las fases de entrenamiento y validación de dicho modelo. Esta arquitectura consta de 31,054,405 parámetros, de los cuales 31,042,629 son entrenables. El modelo entrenado, en formato `*.keras`, se encuentra disponible en nuestro repositorio de GitHub para su consulta y reproducibilidad.

El código de Python *(véase código anexo 8.3)* implementa la arquitectura clásica de *U-Net*. A continuación, se presenta una explicación detallada de cada componente del código, destacando la información más relevante. El bloque convolucional, definido en la función `conv_block`, realiza operaciones convolucionales seguidas de normalización por lotes y activación ReLU.

Esta función toma como entrada la imagen y el número de filtros, y su salida es una característica extraída y normalizada. Específicamente, el bloque realiza dos convoluciones 2D con filtros de tamaño 3x3 y relleno *same* para mantener las dimensiones de la entrada, seguidas de normalización por

lotes para estabilizar y acelerar el entrenamiento, y activación ReLU para introducir no linealidades.

El bloque del codificador, definido en la función `encoder_block`, aplica un bloque convolucional seguido de una operación de *max pooling*. Esta función toma como entrada la imagen y el número de filtros, y su salida incluye tanto las características extraídas mediante convoluciones como la imagen reducida en resolución mediante *max pooling* de tamaño 2x2.

El bloque del decodificador, definido en la función `decoder_block`, reconstruye la imagen a partir de las características extraídas, combinándolas con las características del codificador. Esta función toma como entrada la imagen, las características del codificador (*skip features*), y el número de filtros. La salida es la característica reconstruida. El bloque del decodificador utiliza una convolución transpuesta para aumentar la resolución de la imagen, concatena estas características con las correspondientes del codificador, y luego aplica un bloque convolucional.

La función `build_unet` construye la arquitectura completa del *U-Net*. Esta función toma como entradas el número de clases y las dimensiones de la imagen de entrada (por ejemplo, 256x256). La arquitectura del *U-Net* consta de un codificador, un puente y un decodificador. El codificador incluye cuatro bloques de codificación, cada uno aplicando convoluciones y reduciendo la resolución mediante *max pooling*. El puente actúa como cuello de botella entre el codificador y el decodificador, con una capa convolucional de alta dimensionalidad. El decodificador incluye cuatro bloques de decodificación que aumentan la resolución de la imagen y combinan características del codificador. La capa final del *U-Net* es una convolución 2D con activación *softmax* para producir mapas de probabilidad por clase.

El gráfico titulado "Función de pérdidas - U-Net" *(véase imagen 6.2)* ilustra la evolución de la función de pérdida tanto para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul se mantiene estable desde el inicio, finalizando con valores de 0.0390 (3,9 %) para validación y 0.0210 (2,1 %) para entrenamiento, con una diferencia de 0.018 (1,8 %), lo que indica que no hay sobreajuste.

La estabilidad y los bajos valores de esta curva indican una buena adaptación del modelo a los datos de entrenamiento, minimizando efectivamente el error en la predicción de las segmentaciones. La línea roja exhibe un comportamiento más volátil en comparación con la pérdida de entrenamiento. Aunque la tendencia general muestra una disminución, hay picos significativos observados en varias épocas, por ejemplo, cerca de las épocas 50, 100 y 150.

Las fluctuaciones observadas en la función de pérdida de validación durante el entrenamiento del modelo `U-Net` para la segmentación de imágenes CT del área abdominal pueden atribuirse a varios factores importantes. En primer lugar, la *variabilidad de datos* juega un papel fundamental.

Dado que el conjunto de validación puede contener características o variaciones ausentes en el

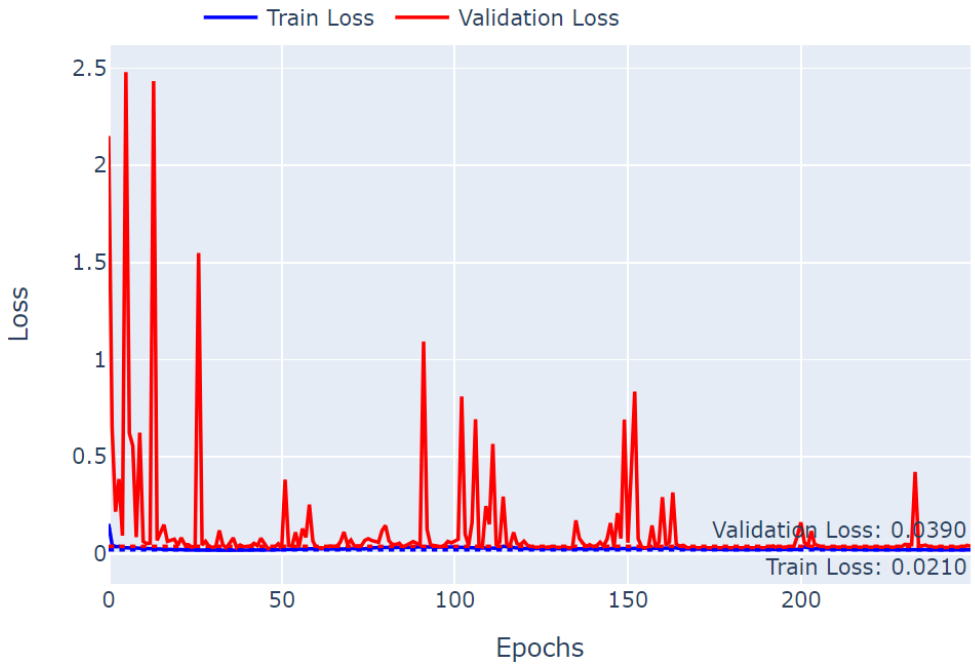

Figura 6.2: Función de perdidas - U-Net. Fuente: *Autores*.

conjunto de entrenamiento, el modelo puede experimentar dificultades para generalizar, lo que se refleja en los picos de la función de pérdida. Analizando estos picos de inestabilidad en la curva de validación, se encuentran y se descartan las siguientes razones:

Se descarta el *overfitting*, ya que a medida que el modelo se entrena, puede aprender patrones específicos del conjunto de entrenamiento y generalizar bien los datos de validación según la curva.

Se considera la variabilidad en el conjunto de validación, dado que este conjunto incluye pacientes distintos que no necesariamente representan bien la distribución de los datos de entrenamiento, la pérdida de validación puede mostrar grandes variaciones.

Se considera el desbalance en las clases. En tareas de segmentación semántica, si algunas clases están subrepresentadas, como es el caso de las máscaras 3 y 4 que tienen pocos píxeles, el modelo puede tener dificultades para aprender a segmentar estas clases correctamente, resultando en una pérdida de validación inestable.

Se descarta un proceso de regularización, ya que el efecto de *dropout* no se consideró debido a que las arquitecturas suelen incluir *backbones* en el *encoder*, y además se utilizaron arquitecturas estándar de acuerdo a lo reportado en el estado del arte. Complementariamente, se realizó una aumentación de datos para obtener un conjunto de datos más robusto a variaciones en los datos de

entrada.

Las estrategias planteadas para mitigar estos problemas incluyen aumentar el conjunto de datos de entrenamiento y validación, es decir, considerar más pacientes para que el conjunto de validación sea suficientemente grande y representativo, lo cual puede ayudar a estabilizar la pérdida de validación.

El uso de otra función de pérdidas que sea adecuada y balanceada, de modo que penalice más las clases subrepresentadas, puede asegurar que el modelo aprenda a segmentar todas las clases correctamente. Por otro lado, se puede considerar la validación cruzada para asegurar que los resultados no dependan de un solo conjunto de validación y obtener una estimación más robusta del rendimiento del modelo.

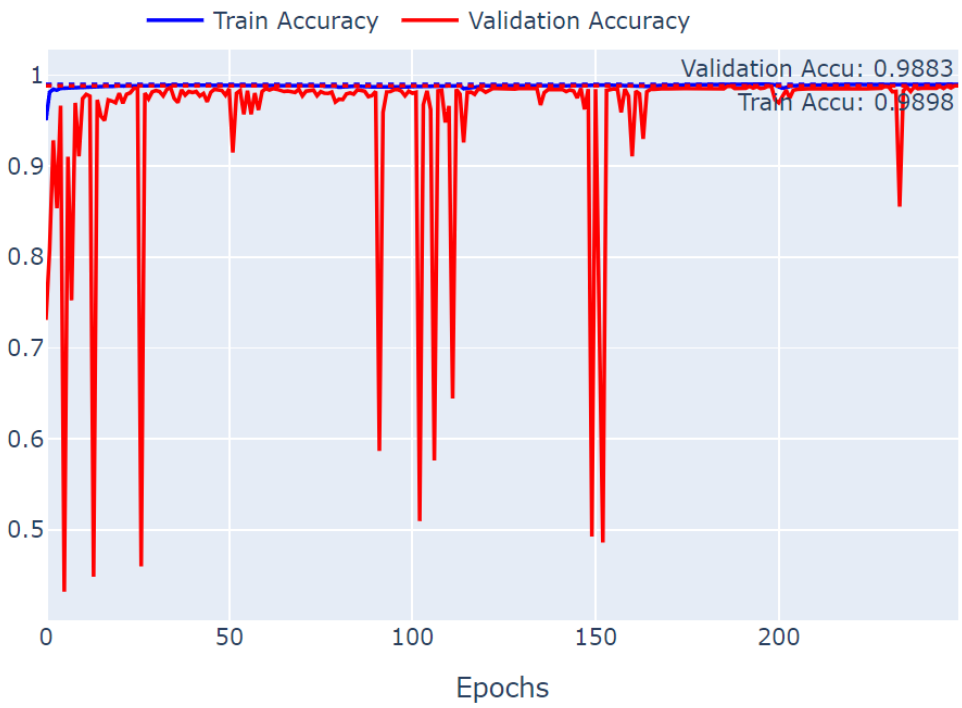

Figura 6.3: Métrica Accuracy - U-Net. Fuente: *Autores.*

El gráfico titulado "Accuracy en Training y Test - U-Net" *(véase imagen 6.3)* ilustra la evolución de la precisión del modelo para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable desde la época 170, al final los valores 0.9883 (98, 83 %) para validación y 0.98 (98, 98 %) para entrenamiento con una diferencia de 0.0015 (0, 15 %), lo que indica que no hay sobreajuste.

La línea azul, que representa la precisión en el conjunto de entrenamiento, muestra un comportamiento relativamente estable y alto, manteniendo un valor cercano a 0.9883 (98, 98 %). Este alto nivel de precisión indica que el modelo es capaz de aprender efectivamente las características relevantes de los datos de entrenamiento, lo cual es un indicativo positivo de su capacidad de aprendizaje.

Por otro lado, la línea roja muestra la precisión en el conjunto de validación. A diferencia de la curva de entrenamiento, esta línea exhibe una serie de caídas significativas y picos a lo largo de las épocas, especialmente notorios en torno a las épocas 20, 100, 150. La precisión máxima alcanzada es de aproximadamente 0.9883 (98, 83 %), lo cual es comparativamente alto, pero sugiere ciertas fluctuaciones en el rendimiento del modelo cuando se enfrenta a nuevos datos.

El análisis de la precisión del modelo *U-Net* en la tarea de segmentación de imágenes CT del área abdominal revela un alto grado de aprendizaje y capacidad de adaptación a los datos de entrenamiento. Sin embargo, las variaciones observadas en la precisión de validación requieren una revisión detallada de los parámetros de entrenamiento y posiblemente la implementación de técnicas de regularización para mejorar la generalización del modelo.

Adicional, hemos evaluado la eficacia del modelo *U-Net* para la segmentación de imágenes médicas utilizando diversas métricas de desempeño. Los resultados obtenidos en las fases de entrenamiento y prueba se resumen en las Tablas 6.2 y 6.3 respectivamente.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	*Puntaje*
0.9898	0.928	0.9381	0.8734	0.021	0.9144	0.9461	0.9383

Cuadro 6.2: Tabla de resultados de U-Net en Train

La Tabla 6.2 muestra que el modelo alcanzó un puntaje de 0.9383, una *accuracy* de 0.9898, lo que indica una alta tasa de clasificaciones correctas. El Coeficiente DICE, que mide la similitud entre los objetos segmentados por el modelo y los objetos reales, fue de 0.928, destacando la precisión del modelo en la segmentación. El *IoU*, o *Intersection over Union*, alcanzó un valor de 0.8734, lo que refuerza la eficacia del modelo en la identificación precisa de áreas de interés. Los valores de *precision* y *recall* fueron altos, 0.9144 y 0.9461 respectivamente, lo que indica que el modelo es muy competente en detectar y clasificar píxeles pertinentes. Las pérdidas son de 0.021 muestra que el modelo tiene un bajo error al predecir las segmentaciones.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9883	0.9063	0.9134	0.849	0.039	0.8997	0.9315	0.9213

Cuadro 6.3: Tabla de resultados de U-Net en Test

Los resultados de la fase de prueba, presentados en la Tabla 6.3, confirman la robustez del modelo. Con un alcanzó un puntaje de 0.9213, *accuracy* de 0.9883, ligeramente inferior a la del entrenamiento, pero aún destacada. El coeficiente DICE y el *IoU* presentaron valores elevados (0.9063

y 0.849 respectivamente), lo que indica una buena generalización del modelo a nuevos datos. La *precision* y *recall* también se mantuvieron altas en la fase de prueba, siendo de 0.8997 y 0.9315, respectivamente. La función de perdidas fue de 0.039, reflejando un incremento respecto al entrenamiento, pero manteniendo un nivel aceptable de error.

El análisis de las métricas revela que el modelo *U-Net* ha demostrado ser altamente efectivo y robusto para la tarea de segmentación de imágenes médicas. Estos resultados son prometedores para aplicaciones futuras en entornos clínicos reales donde la precisión y la confiabilidad son fundamentales.

6.1.1.2. Modelo SegNet

La segunda arquitectura seleccionada para este estudio fue *SegNet* [4]. A continuación, se presenta un análisis exhaustivo de las métricas obtenidas durante las fases de entrenamiento y validación de dicho modelo. Esta arquitectura consta de 11,742,853 parámetros, de los cuales 11,735,173 son entrenables. El modelo entrenado, en formato `*.keras`, se encuentra disponible en nuestro repositorio de GitHub para su consulta y reproducibilidad.

El código de Python *(véase código anexo 8.4)* implementa la arquitectura clásica de *SegNet*. A continuación, se presenta una explicación detallada de cada componente del código, destacando la información más relevante.

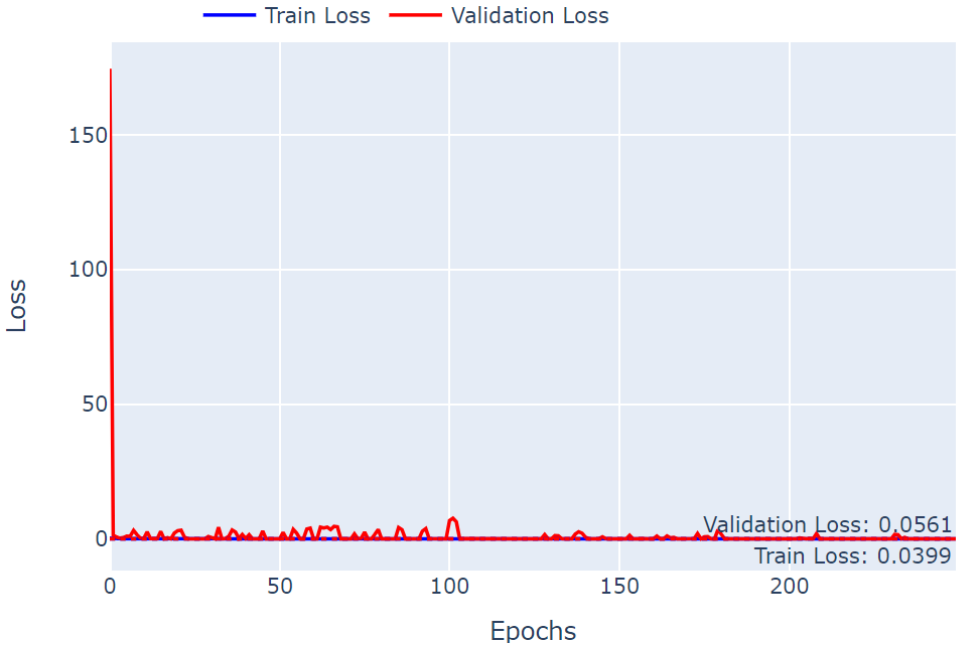

Figura 6.4: Función de perdidas - SegNet. Fuente: *Autores*.

El gráfico titulado "Función de perdidas - SegNet" *(véase imagen 6.4)* ilustra la evolución de la función de pérdida tanto para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul se mantiene un nivel estable desde la época 236, al final los valores 0.0561 ($5,61\,\%$) para validación y 0.039 ($3,9\,\%$) para entrenamiento con una diferencia de 0.0171 ($1,71\,\%$), lo que indica que no hay sobreajuste.

La estabilidad y bajos valores de esta curva indican una buena adaptación del modelo a los datos de entrenamiento, minimizando de manera efectiva el error en la predicción de las segmentaciones. La línea azul indica muestra una reducción significativa desde el inicio del entrenamiento estabilizándose rápidamente cerca de un valor bajo de 0.0399 hacia el final del entrenamiento. Esta rápida estabilización sugiere una convergencia efectiva del modelo al minimizar el error de las predicciones en el conjunto de entrenamiento.

La línea roja representa la pérdida en el conjunto de validación. A diferencia de la curva de entrenamiento, esta curva muestra fluctuaciones menores a lo largo del proceso de entrenamiento, pero mantiene una tendencia consistente y baja, con un valor final aproximado de 0.0561. La relativa estabilidad y el bajo nivel de esta curva son indicativos de una buena generalización del modelo sobre nuevos datos, lo que es crítico para su aplicación en entornos clínicos.

La rápida disminución y estabilización temprana de la pérdida de *entrenamiento* indica que el modelo es capaz de aprender de manera eficiente las características relevantes de los datos de *entrenamiento*. Esta convergencia rápida es un indicio prometedor de la capacidad del modelo para capturar los patrones subyacentes en los datos.

Aunque se observan fluctuaciones en la curva de *validación*, estas son comparativamente menores y no indican problemas significativos de *sobreajuste*. Este aspecto es fundamental, porque un modelo que se sobreajusta a los datos de *entrenamiento* puede tener un desempeño deficiente en datos nuevos, lo cual es indeseable en aplicaciones clínicas.

El análisis detallado de la función de pérdidas en el *entrenamiento* y la *validación* del modelo *SegNet* aplicado a la segmentación de imágenes CT del área abdominal revela un rendimiento prometedor tanto en el aprendizaje como en la generalización. La rápida convergencia durante el *entrenamiento* y la capacidad de generalización demostrada en la *validación* posicionan a *SegNet* como un candidato viable para aplicaciones clínicas donde la precisión y la estabilidad son críticas.

El gráfico titulado "Métrica Accuracy - SegNet" *(véase imagen 6.5)* ilustra la evolución de la precisión del modelo para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable desde la época 170, al final los valores 0.9772 ($97,72\,\%$) para validación y 0.982 ($98,2\,\%$) para entrenamiento con una diferencia de 0.0015 ($0,48\,\%$), lo que indica que no hay sobreajuste.

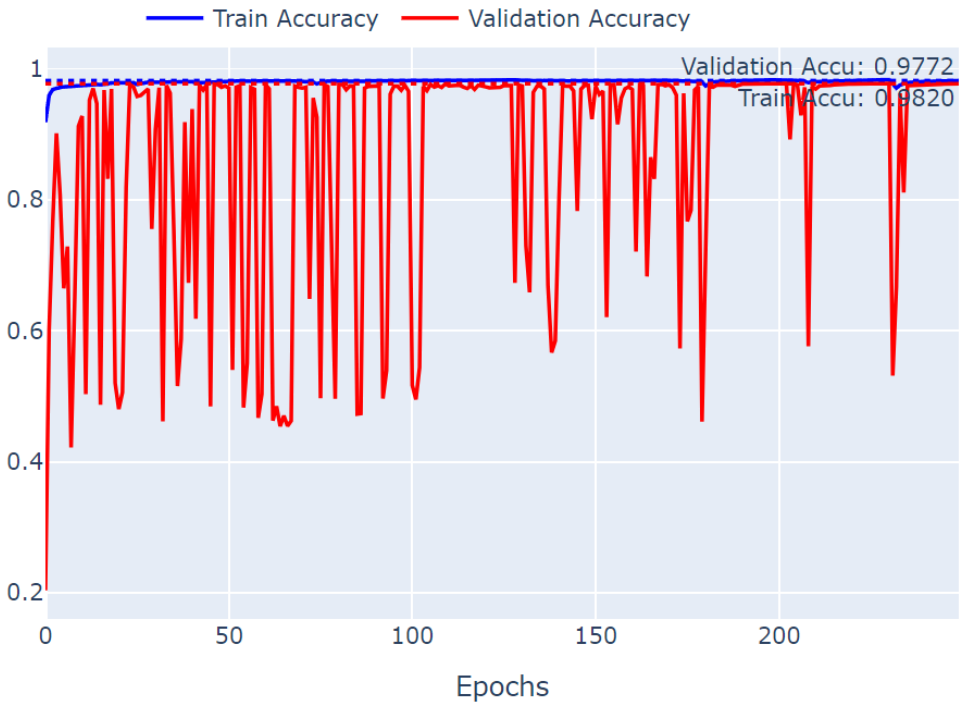

Figura 6.5: Métrica Accuracy - SegNet. Fuente: *Autores*.

La línea azul muestra la precisión en el conjunto de entrenamiento, que se mantiene constante y elevada, cerca del 0.982 ($98,2\,\%$) durante la mayor parte del entrenamiento. Esta alta y estable precisión en entrenamiento indica que el modelo *SegNet* es capaz de aprender eficazmente las características de los datos de entrenamiento.

La línea roja representa la precisión en el conjunto de validación. Aunque esta línea comienza cerca de los niveles de la precisión de entrenamiento, presenta fluctuaciones significativas a lo largo de todas las épocas. Estas fluctuaciones son marcadas y repetitivas, especialmente notables en torno a las épocas 50, 100, 150 y 200, aunque la precisión alcanza un valor máximo de aproximadamente 0.9772 ($97,72\,\%$).

Las fluctuaciones observadas en la precisión de validación durante el entrenamiento del modelo *SegNet* para la segmentación de imágenes CT del área abdominal pueden ser atribuidas a diversos factores. Uno de los principales aspectos a considerar es la estabilidad del modelo es la configuración de los parámetros de entrenamiento, como la tasa de aprendizaje (`learning_rate`) y el tamaño del lote (`batch_size`). Una tasa de aprendizaje inadecuada o un tamaño de lote subóptimo pueden conducir a un rendimiento inconsistente del modelo en el conjunto de validación.

El análisis exhaustivo de la precisión del modelo *SegNet* en la tarea de segmentación de imágenes

CT del área abdominal ha revelado su eficacia en el aprendizaje de los datos de entrenamiento. Sin embargo, se han identificado desafíos en la capacidad del modelo para generalizar a datos nuevos, evidenciados por la alta variabilidad en la precisión de validación.

La arquitectura *SegNet* ha sido evaluada en este estudio para determinar su desempeño en la segmentación semántica de imágenes. Las métricas clave de rendimiento, han sido calculadas tanto para los datos de entrenamiento como de prueba, y se presentan en las Tablas 6.4 y 6.5.

La arquitectura *SegNet* ha sido evaluada en este estudio para determinar su desempeño en la segmentación semántica de imágenes. Las métricas clave de rendimiento, han sido calculadas tanto para los datos de entrenamiento como de prueba, y se presentan en las Tablas 6.4 y 6.5.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.982	0.8994	0.9091	0.8265	0.0399	0.8856	0.9163	0.9112

Cuadro 6.4: Tabla de resultados de SegNet en Train

La Tabla 6.5 muestra que el modelo alcanzó un puntaje de 0.9112, un *accuracy* de 0.982, indicando una alta tasa de acierto en la clasificación de píxeles. El coeficiente DICE de 0.8994 y el *IoU* de 0.8265 una buena superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.8856 y 0.9163 respectivamente, destacan la eficacia del modelo en la detección de píxeles relevantes y la minimización de falsos negativos. El *loss* registrado fue de 0.0422, lo cual es relativamente bajo, indicando un buen desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de SegNet en entrenamiento se encuentran por debajo de lo resultados obtenidos con *U-Net*.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9772	0.8679	0.8785	0.7893	0.0561	0.8543	0.8988	0.8871

Cuadro 6.5: Tabla de resultados de SegNet en Test

Los resultados obtenidos en la evaluación de SegNet en test, se observa que el modelo alcanzó un puntaje de 0.8871, un *accuracy* de 0.9772, indicando una alta tasa de acierto en la clasificación de píxeles. El coeficiente DICE de 0.8679 y el *IoU* de 0.7893, indican una buena superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.8543 y 0.8988 respectivamente, destacan la eficacia del modelo en la detección de píxeles relevantes y la minimización de falsos negativos. El *loss* registrado fue de 0.0561, lo cual es relativamente bajo, indicando un buen desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de SegNet en test se encuentran por

debajo de lo resultados obtenidos con *U-Net*.

6.1.1.3. Modelo U-Net con VGG16

La tercera arquitectura seleccionada para este estudio fue *U-Net* con *Backbone VGG16*. A continuación, se presenta un análisis exhaustivo de las métricas obtenidas durante las fases de entrenamiento y validación de dicho modelo. Esta arquitectura consta de 23,751,701 parámetros, de los cuales 23,747,669 son entrenables. El modelo entrenado, en formato `*.keras`, se encuentra disponible en nuestro repositorio de GitHub para su consulta y reproducibilidad.

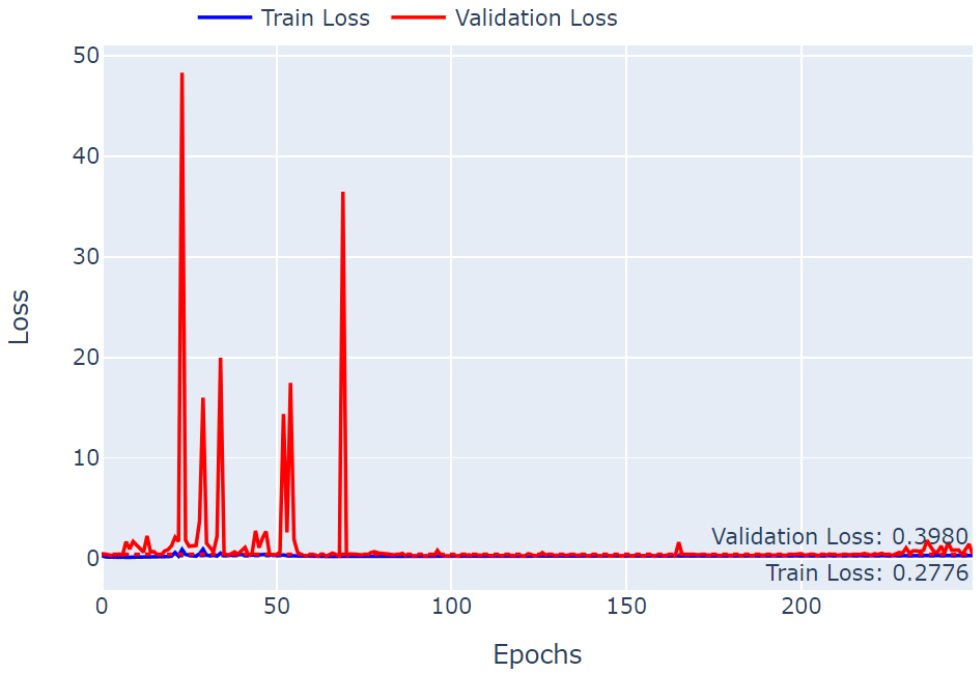

Figura 6.6: Función de perdidas - U-Net con VGG16. Fuente: *Autores*.

El gráfico titulado "Función de perdidas - U-Net con VGG16" *(véase imagen 6.6)* ilustra la evolución de la función de pérdida tanto para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable desde la época 70, al final los valores 0.0380 (3,8 %) para validación y 0.0276 (2,76 %) para entrenamiento con una diferencia de 0.0104 (1,04 %), lo que indica que no hay sobreajuste. La estabilidad y bajos valores de esta curva indican una buena adaptación del modelo a los datos de entrenamiento, minimizando de manera efectiva el error en la predicción de las segmentaciones.

El gráfico titulado "Métrica *Accuracy* - U-Net con VGG16" *(véase imagen 6.7)* ilustra la evolución de la precisión del modelo para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable

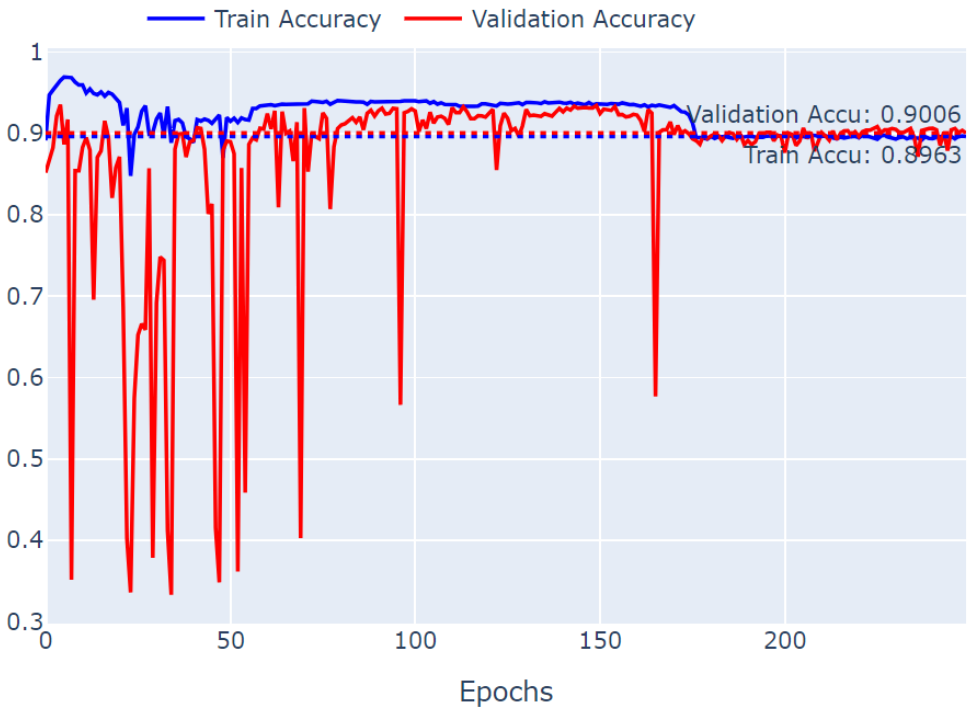

Figura 6.7: Métrica *Accuracy* - U-Net con VGG16. Fuente: *Autores*.

desde la época 170, al final los valores 0.9006 (90, 06 %) para validación y 0.8963 (89, 63 %) para entrenamiento con una diferencia de 0.0015 (0, 43 %), lo que indica que no hay sobreajuste pero si un estancamiento de la métrica sobre el (90 %).

La arquitectura *U-Net con VGG16* ha sido evaluada en este estudio para determinar su desempeño en la segmentación semántica de imágenes. Las métricas clave de rendimiento, han sido calculadas tanto para los datos de entrenamiento como de prueba, y se presentan en las Tablas 6.6 y 6.7.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.8963	0.4206	0.4221	0.364	0.2776	0.4229	0.4242	0.5246

Cuadro 6.6: Tabla de resultados de U-Net con VGG16 en Train

La Tabla 6.7 muestra que el modelo alcanzó un puntaje de 0.5246, un *accuracy* de 0.8963, indicando una baja tasa de acierto en la clasificación de píxeles comparado con arquitecturas anteriores. El coeficiente DICE de 0.4206 y el *IoU* de 0.364 con una mala superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la poca capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.4229 y 0.4242 respectivamente, destacan la ineficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.2776, lo cual es relativamente alto, indicando un mal desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de U-Net con VGG16 en entrenamiento se encuentran por debajo de lo resultados obtenidos con *U-Net*.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9006	0.4083	0.4126	0.364	0.398	0.415	0.4398	0.5060

Cuadro 6.7: Tabla de resultados de U-Net con VGG16 en Test

Los resultados obtenidos en la evaluación de U-Net con VGG16 en test, se encuentran por debajo de los resultados obtenidos con *U-Net*. El modelo alcanzó un puntaje de 0.5060, un *accuracy* de 0.9006, indicando una baja tasa de acierto en la clasificación de píxeles comparado con arquitecturas anteriores. El Coeficiente DICE de 0.4083 y el *iou* de 0.364 con una mala superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la poca capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.415 y 0.4398 respectivamente, destacan la ineficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.398, lo cual es relativamente alto, indicando un mal desempeño general del modelo durante el entrenamiento.

6.1.1.4. Modelo U-Net con ResNet50

La cuarta arquitectura seleccionada para este estudio fue *U-Net* con *Backbone ResNet50*. A continuación, se presenta un análisis exhaustivo de las métricas obtenidas durante las fases de entrenamiento y validación de dicho modelo. Esta arquitectura consta de 32,555,416 parámetros, de los cuales 32,507,862 son entrenables. El modelo entrenado, en formato `*.keras`, se encuentra disponible en nuestro repositorio de GitHub para su consulta y reproducibilidad.

El gráfico titulado "Función de perdidas - U-Net con ResNet50" *(véase imagen 6.8)* ilustra la evolución de la función de pérdida tanto para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable desde las primeras épocas, con pequeños picos que no superan al 0.5 y final los valores 0.0616 (6, 16 %) para validación y 0.0490 (4, 9 %) para entrenamiento con una diferencia de 0.018 (1, 26 %), lo que indica que no hay sobreajuste. La estabilidad y bajos valores de esta curva indican una buena adaptación del modelo a los datos de entrenamiento, minimizando de manera efectiva el error en la predicción de las segmentaciones.

El gráfico titulado "Métrica *Accuracy* - U-Net con ResNet50" *(véase imagen 6.9)* ilustra la evolución de la precisión del modelo para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable

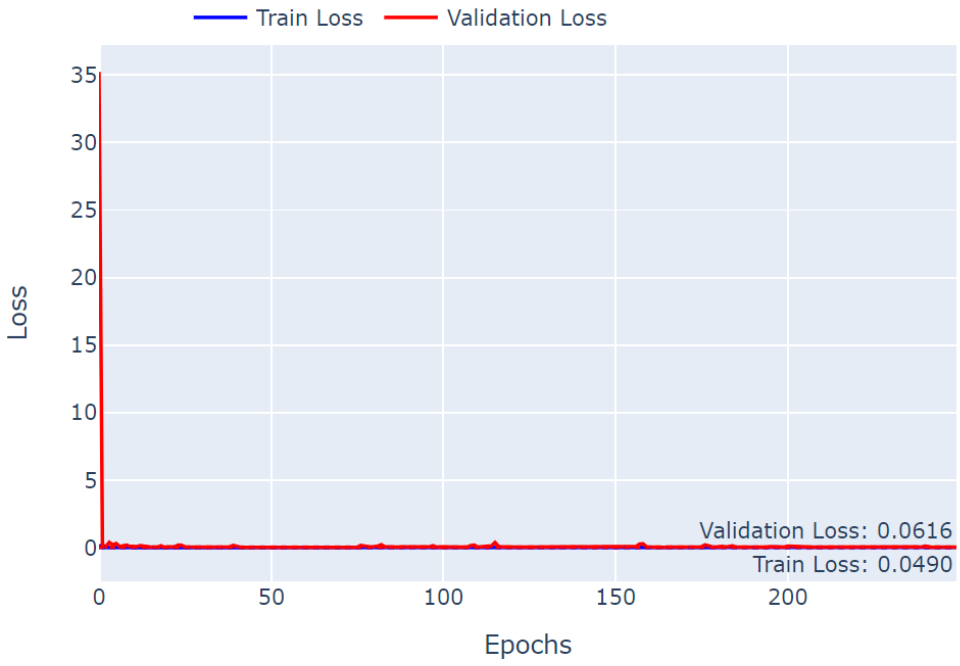

Figura 6.8: Función de perdidas - U-Net con ResNet50. Fuente: *Autores*.

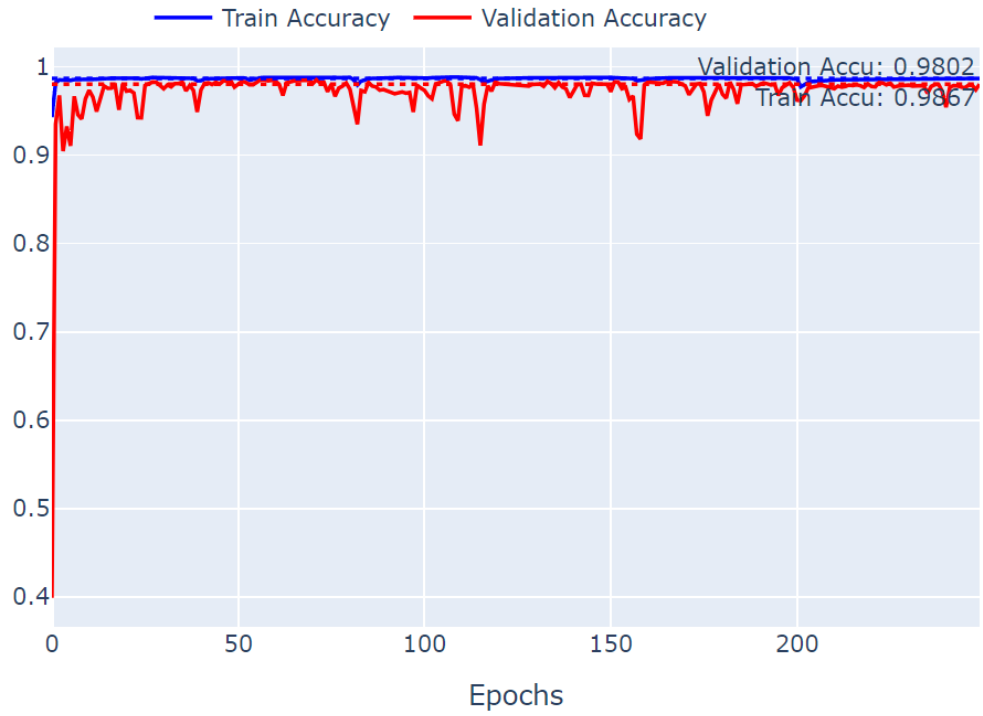

Figura 6.9: Métrica *Accuracy* - U-Net con ResNet50. Fuente: *Autores*.

desde las primeras épocas con oscilaciones que varían ±0,1, al final los valores son 0.9802 (98, 02 %) para validación y 0.9867 (98, 67 %) para entrenamiento con una diferencia de 0.0065 (0, 65 %), de todas las arquitecturas planteadas *U-Net* con *Backbone ResNet50* es la que presenta mayor estabilidad en las funciones de pérdidas y la métrica de *Accuracy*, que la ubica como una gran candidata en el problema de segmentación.

La arquitectura *U-Net con ResNet50* ha sido evaluada en este estudio para determinar su desempeño en la segmentación semántica de imágenes. Las métricas clave de rendimiento, han sido calculadas tanto para los datos de entrenamiento como de prueba, y se presentan en las Tablas 6.8 y 6.9.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9867	0.8573	0.8824	0.768	0.049	0.8246	0.9024	0.8817

Cuadro 6.8: Tabla de resultados de U-Net con ResNet50 en Train

La Tabla 6.9 muestra que el modelo alcanzó un puntaje de 0.8817, un *accuracy* de 0.9867, indicando una alta tasa de acierto en la clasificación de píxeles. El coeficiente DICE de 0.8573 y el *IoU* de 0.8824 con una aceptable superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.8246 y 0.9024 respectivamente, destacan la eficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.049, lo cual es relativamente bajo, indicando un buen desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de U-Net con ResNet50 en entrenamiento se encuentran por debajo de lo resultados obtenidos con *U-Net*

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9802	0.7944	0.8198	0.7011	0.0616	0.7658	0.8572	0.8367

Cuadro 6.9: Tabla de resultados de U-Net con ResNet50 en Test

Los resultados obtenidos en la evaluación de U-Net con ResNet50 en test, muestran un puntaje de 0.8367, un *accuracy* de 0.9802, indicando una alta tasa de acierto en la clasificación de píxeles. El Coeficiente DICE de 0.7944 y el *IoU* de 0.8198 con una aceptable superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.7658 y 0.8572 respectivamente, destacan la eficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.0616, lo cual es relativamente bajo, indicando un buen desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de U-Net con ResNet50 en test se encuentran por debajo de lo resultados obtenidos con *U-Net*

6.1.1.5. Modelo FPN con ResNet50

La quinta arquitectura inicial seleccionada para este estudio fue *FPN* con *Backbone ResNet50*. A continuación, se presenta un análisis exhaustivo de las métricas obtenidas durante las fases de entrenamiento y validación de dicho modelo. Esta arquitectura consta de 26,910,152 parámetros, de los cuales 26,862,278 son entrenables. El modelo entrenado, en formato `*.keras`, se encuentra disponible en nuestro repositorio de GitHub para su consulta y reproducibilidad.

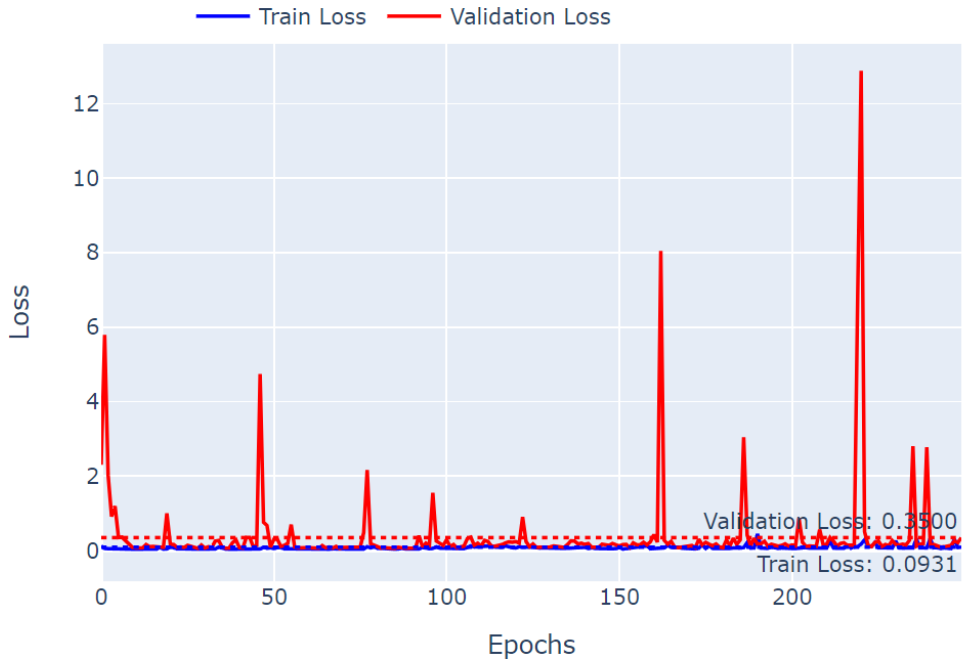

Figura 6.10: Función de perdidas - FPN con ResNet50. Fuente: *Autores*.

El gráfico titulado "Función de perdidas - FPN con ResNet50" *(véase imagen 6.10)* ilustra la evolución de la función de pérdida tanto para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul es constante y la línea roja presenta varios picos a lo largo del entrenamiento, al final los valores 0.35 (35 %) para validación y 0.0931 (9, 31 %) para entrenamiento con una diferencia de 0.2569 (25, 69 %), lo que indica que hay sobreajuste. La inestabilidad y picos en los valores de esta curva indican una mala adaptación del modelo a los datos de entrenamiento, lo que puede llevar a errores en la predicción de las segmentaciones.

El gráfico titulado "Métrica *Accuracy* - FPN con ResNet50" *(véase imagen 6.11)* ilustra la evolución de la precisión del modelo para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul y roja se mantienen un nivel estable desde la época 170, al final los valores 0.9551 (95, 51 %) para validación y 0.9796 (97, 96 %)

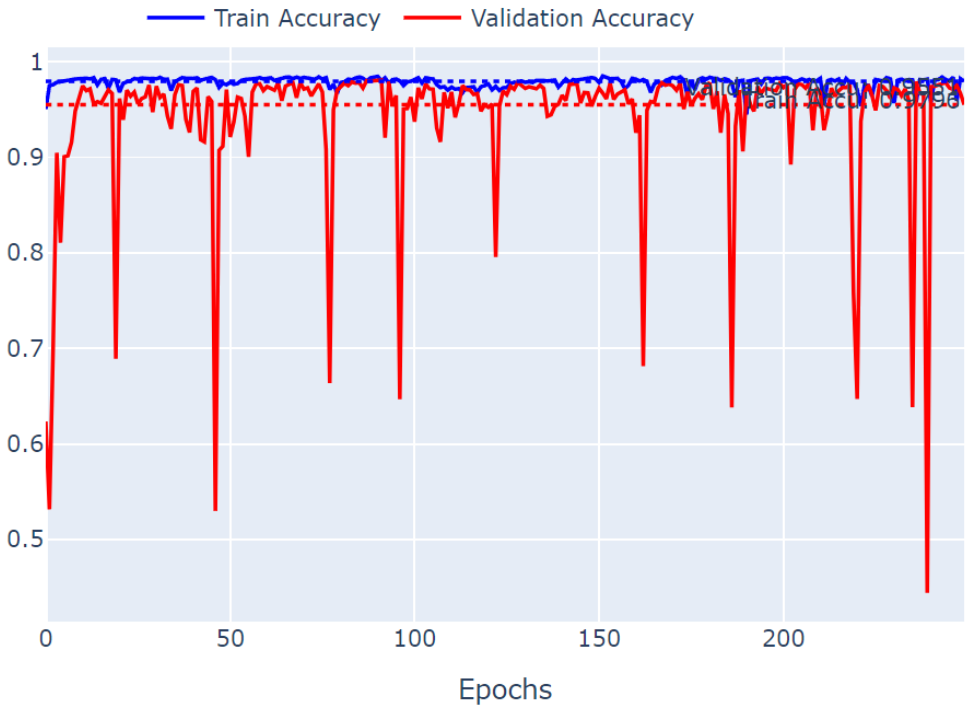

Figura 6.11: Métrica *Accuracy* - FPN con ResNet50. Fuente: *Autores*.

para entrenamiento con una diferencia de 0.0245 (2, 45 %), lo que indica que hay un leve sobreajuste.

La línea azul, muestra un comportamiento relativamente estable y alto, manteniendo un valor cercano a 0.9796 (97, 96 %). Este alto nivel de precisión indica que el modelo es capaz de aprender efectivamente las características relevantes de los datos de entrenamiento, lo cual es un indicativo positivo de su capacidad de aprendizaje.

Por otro lado, la línea roja muestra la precisión en el conjunto de validación. A diferencia de la curva de entrenamiento, esta línea exhibe una serie de caídas significativas y picos a lo largo de todas las épocas. La precisión máxima alcanzada es de aproximadamente 0.9551 (95, 51 %), lo cual es comparativamente alto, pero sugiere ciertas fluctuaciones en el rendimiento del modelo cuando se enfrenta a nuevos datos.

La arquitectura *FPN con ResNet50* ha sido evaluada en este estudio para determinar su desempeño en la segmentación semántica de imágenes. Las métricas clave de rendimiento, han sido calculadas tanto para los datos de entrenamiento como de prueba, y se presentan en las Tablas 6.10 y 6.11.

La Tabla 6.11 muestra que el modelo alcanzó un puntaje de 0.9195, un *accuracy* de 0.9796, indicando una alta tasa de acierto en la clasificación de píxeles. El Coeficiente DICE de 0.9162 y

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9796	0.9162	0.9177	0.8606	0.0931	0.933	0.9226	0.9195

Cuadro 6.10: Tabla de resultados de FPN con ResNet50 en Train

el *IoU* de 0.8606 con una aceptable superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.933 y 0.9226 respectivamente, destacan la eficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.0931, lo cual es relativamente alto, indicando un buen desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de FPN con ResNet50 en entrenamiento se encuentran por debajo de lo resultados obtenidos con *U-Net*

accuracy	*DICE*	*f1-score*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.9551	0.7616	0.7616	0.7181	0.6673	0.35	0.9187	0.709	0.7685

Cuadro 6.11: Tabla de resultados de FPN con ResNet50 en Test

Los resultados obtenidos en la evaluación de FPN con ResNet50 en test se encuentran por debajo de lo resultados obtenidos con *U-Net*. El modelo alcanzó un puntaje de 0.7685, un *accuracy* de 0.9551, indicando una alta tasa de acierto en la clasificación de píxeles. El Coeficiente DICE de 0.7616 y el *IoU* de 0.6673 con una aceptable superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la capacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.9187 y 0.709 respectivamente, destacan la eficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.35, lo cual es relativamente alto, indicando un buen desempeño general del modelo durante el entrenamiento.

6.1.1.6. Modelo LinkNet con VGG16

La sexta arquitectura seleccionada para este estudio fue *LinkNet* con *Backbone VGG16*. A continuación, se presenta un análisis exhaustivo de las métricas obtenidas durante las fases de entrenamiento y validación de dicho modelo. Esta arquitectura consta de 20,324,565 parámetros, de los cuales 20,317,749 son entrenables. El modelo entrenado, en formato `*.keras`, se encuentra disponible en nuestro repositorio de GitHub para su consulta y reproducibilidad.

El gráfico titulado “Función de perdidas - LinkNet con VGG16” *(véase imagen 6.12)* ilustra la evolución de la función de pérdida tanto para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul es constante y la línea roja presenta varios picos a lo largo del entrenamiento, al final los valores 0.2311 $(23,11\,\%)$ para

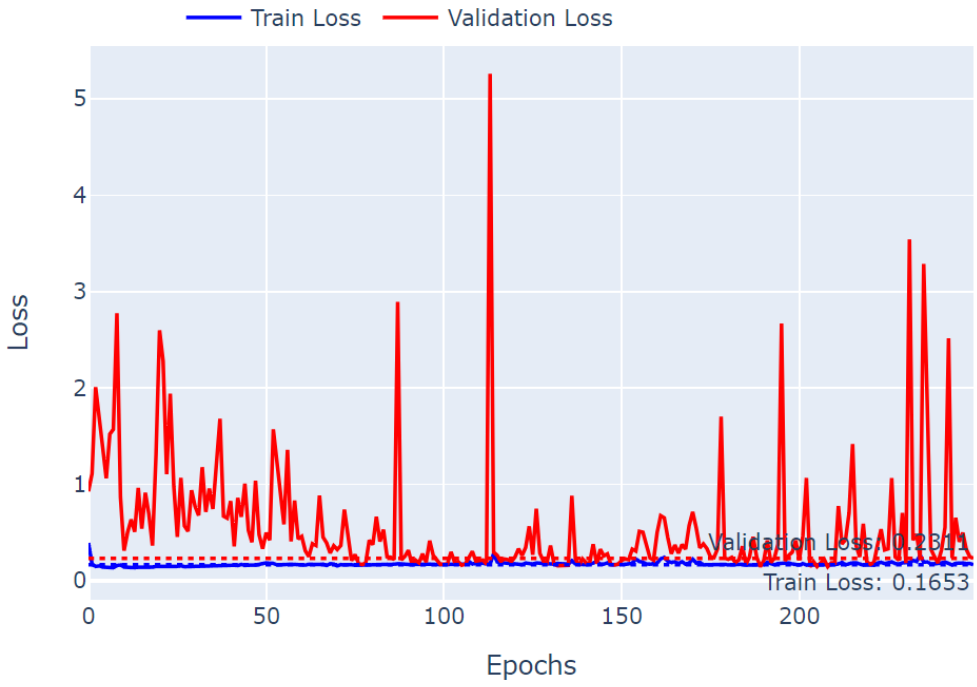

Figura 6.12: Función de perdidas - LinkNet con VGG16. Fuente: *Autores.*

validación y 0.1653 (16, 53 %) para entrenamiento con una diferencia de 0.0658 (6, 58 %), lo que indica que hay sobreajuste. La inestabilidad y picos en los valores de esta curva indican una mala adaptación del modelo a los datos de entrenamiento, lo que puede llevar a errores en la predicción de las segmentaciones.

El gráfico titulado "Métrica *Accuracy* - LinkNet con VGG16" *(véase imagen 6.13)* ilustra la evolución de la precisión del modelo para el entrenamiento (línea azul) como para la validación (línea roja) a lo largo de 250 épocas. Observamos que la línea azul se mantienen un nivel estable desde las primeras épocas y la línea roja presenta fluctuaciones a lo largo del entrenamiento, al final los valores son 0.935 (93, 5 %) para validación y 0.943 (94, 3 %) para entrenamiento con una diferencia de 0.008 (0, 8 %),

La arquitectura *LinkNet con VGG16* ha sido evaluada en este estudio para determinar su desempeño en la segmentación semántica de imágenes. Las métricas clave de rendimiento, han sido calculadas tanto para los datos de entrenamiento como de prueba, y se presentan en las Tablas 6.12 y 6.13.

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.943	0.5613	0.5656	0.4866	0.1653	0.5674	0.5729	0.6473

Cuadro 6.12: Tabla de resultados de LinkNet con VGG16 en Train

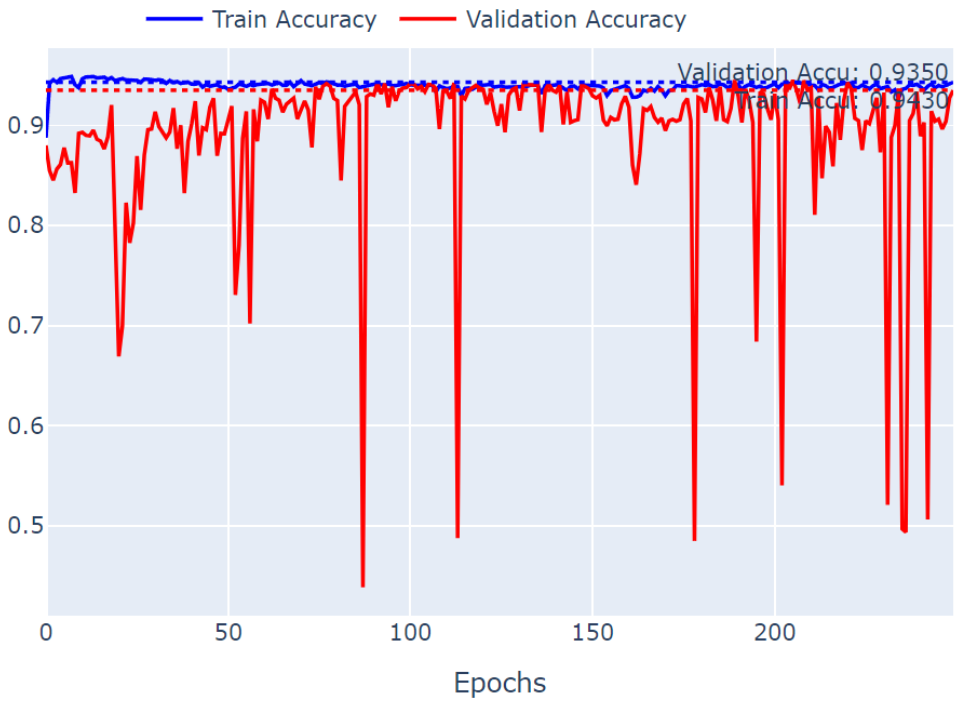

Figura 6.13: Métrica *Accuracy* - LinkNet con VGG16. Fuente: *Autores*.

La Tabla 6.13 muestra que el modelo alcanzó un puntaje de 0.6473, un *accuracy* de 0.943, indicando una alta tasa de acierto en la clasificación de píxeles. El Coeficiente DICE de 0.5613 y el *IoU* de 0.4866 con una pobre superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la incapacidad del modelo para identificar con precisión las regiones de interés.

Los valores de *precision* y *recall*, 0.5674 y 0.5729 respectivamente, destacan la ineficacia del modelo en la detección de píxeles relevantes. El *loss* registrado fue de 0.5674, lo cual es relativamente alto, indicando un mal desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de LinkNet con VGG16 en entrenamiento se encuentran por debajo de lo resultados obtenidos con *U-Net*

accuracy	*DICE*	*f2-score*	*IoU score*	*loss*	*precision*	*recall*	**puntaje**
0.935	0.4935	0.4879	0.4391	0.2311	0.562	0.4981	0.5977

Cuadro 6.13: Tabla de resultados de LinkNet con VGG16 en Test

Los resultados obtenidos en la evaluación de LinkNet con VGG16 en test, muestran que el modelo alcanzó un puntaje de 0.5977, un *accuracy* de 0.935, indicando una alta tasa de acierto en la clasificación de píxeles. El Coeficiente DICE de 0.4935 y el *IoU* de 0.4391 con una pobre superposición entre las predicciones del modelo y las etiquetas verdaderas, demostrando la incapacidad del

modelo para identificar con precisión las regiones de interés. Los valores de *precision* y *recall*, 0.562 y 0.4981 respectivamente, destacan la ineficacia del modelo en la detección de píxeles relevantes.

El *loss* registrado fue de 0.2311, lo cual es relativamente alto, indicando un mal desempeño general del modelo durante el entrenamiento. Los resultados obtenidos en la evaluación de LinkNet con VGG16 en test se encuentran por debajo de lo resultados obtenidos con *U-Net*

Los códigos de las arquitecturas se ejecutaron en una Instancia de *Kaggle* usando Python 3,10,13, esta instancia cuenta con un procesador Intel(R) Xeon(R) CPU @ 2.00GHz, Memoria RAM 32 GB, 73 GB de Almacenamiento. Sistema Operativo Ubuntu 20.04.6 LTS, tarjeta gráfica Tesla P100 - PCIE de 16GB GPU y CUDA en la versión 12.2.

6.1.2. Tabla general de puntajes

En el ámbito de la segmentación semántica de imágenes de tomografía computacional, es fundamental evaluar y comparar el desempeño de diversas arquitecturas de redes neuronales. Este proceso de evaluación permite determinar cuáles modelos ofrecen los mejores resultados en términos de las métricas, lo cual es fundamental para aplicaciones médicas donde la exactitud en la segmentación puede tener un impacto significativo en el diagnóstico y tratamiento de los pacientes. La tabla general de puntajes *(véase tabla 6.14)* presentada a continuación resume los resultados obtenidos por diferentes arquitecturas y sus respectivos *Backbones* durante las fases de entrenamiento y validación.

La tabla incluye modelos bien conocidos como *U-Net*, *SegNet*, *FPN* y *LinkNet*, utilizando diferentes *Backbones* como *VGG16* y *ResNet50*. Cada modelo fue evaluado en función de su desempeño durante el entrenamiento y la validación, proporcionando una visión clara de su efectividad y capacidad de generalización. Los puntajes reflejan la capacidad de cada modelo para segmentar con precisión las imágenes de tomografía computacional, y se presentan con el objetivo de identificar las arquitecturas más prometedoras para futuras investigaciones y aplicaciones clínicas.

No.	Arq.	Backbone	Punt. Train	Punt. Val
1	*U-Net*		0.9383	0.9213
2	*SegNet*		0.9112	0.8871
3	*U-Net*	VGG16	0.5246	0.5060
4	*U-Net*	ResNet50	0.8817	0.8367
5	*FPN*	ResNet50	0.9195	0.7685
6	*LinkNet*	VGG16	0.6473	0.5977

Cuadro 6.14: Tabla general de puntajes

De acuerdo con los resultados presentados en la tabla general de puntajes, las tres mejores arquitecturas en términos del puntaje de entrenamiento fueron *U-Net* con un puntaje de 0.9383,

seguida de *SegNet* con un puntaje de 0.9112, y *FPN* con *ResNet50* ocupando el tercer lugar con un puntaje de 0.9195. Estos resultados indican que *U-Net* sin *Backbone* mostró el mejor desempeño general durante el proceso de entrenamiento, seguida de cerca por *SegNet* y *FPN* con *ResNet50*.

En cuanto al puntaje de validación, las tres mejores arquitecturas fueron *U-Net* con un puntaje de 0.9213, *SegNet* con un puntaje de 0.8871, y *U-Net* con *ResNet50* con un puntaje de 0.8367. Estos resultados reflejan la capacidad de los modelos para generalizar sobre datos no vistos durante el entrenamiento, destacando nuevamente a *U-Net* y *SegNet* como las arquitecturas más consistentes en términos de desempeño tanto en entrenamiento como en validación.

Para finalizar hacemos una validación indirecta, esta consiste en validar el proceso de inferencia de los modelos *U-Net*, *SegNet*, *U-Net* con *ResNet50* y *FPN* con *ResNet50* en la misma imagen aleatoria del conjunto de validación, esto nos permite ver como fue el proceso de segmentación sobre cada clase de forma visual. A continuación, se anexan las imágenes donde cada una contiene la imagen en escala de grises de una sección transversal del cuerpo de un paciente, la máscara *(ground truth)* y la predicción de cada clase.

La arquitectura *U-Net*, es una red neuronal convolucional diseñada específicamente para la segmentación de imágenes biomédicas, que ha demostrado un rendimiento destacado en la segmentación de múltiples clases en el conjunto de datos analizado *(véase imagen 6.14)*. Para la clase 0, la red logró segmentar la región general correspondiente, aunque se observaron imprecisiones en los bordes y la inclusión de áreas adicionales no presentes en la máscara de referencia. Se detectó una falsa detección relacionada con un órgano correspondiente a la clase 1, lo que sugiere cierta confusión entre clases adyacentes o similares.

Figura 6.14: Inferencia No. 29 con *U-Net*. Fuente: *Autores*.

En cuanto a la clase 1, la U-Net identificó y segmentó estructuras detalladas con relativa precisión. Sin embargo, se notaron leves discrepancias en las formas y áreas segmentadas en comparación con la máscara de referencia. Esta observación indica que, si bien la red es capaz de capturar detalles finos, aún existe margen de mejora en la exactitud de la segmentación.

La predicción para la clase 2 mostró la segmentación de una estructura específica, aunque con pequeñas diferencias e imprecisiones en comparación con la máscara de referencia. Esto sugiere que la red es capaz de identificar correctamente la estructura, pero puede tener dificultades para delinear con precisión sus límites exactos.

Las predicciones para las clases 3 y 4 presentaron resultados similares a los anteriores, con la red identificando las estructuras correspondientes, pero mostrando algunas diferencias en los bordes y detalles. Estas observaciones son consistentes con el rendimiento general de la red en las otras clases.

En general, se observó que la precisión en la segmentación varía entre las diferentes clases. Las predicciones para las clases 1, 2 y 3 mostraron un alto grado de similitud con las máscaras de referencia, lo que indica un buen rendimiento de la U-Net en estas categorías. Sin embargo, la predicción para la clase 0 presentó más errores, sugiriendo que esta clase puede ser más desafiante para la red o que puede requerir ajustes adicionales en el entrenamiento o en la arquitectura.

Los resultados obtenidos mediante la aplicación de *SegNet* muestran una segmentación efectiva, aunque con variaciones en la precisión para las diferentes clases analizadas *(véase imagen 6.15)*. En la predicción de la clase 0, se observa que la red ha logrado segmentar la región general correspondiente a la máscara de referencia. Sin embargo, se aprecian diferencias en la definición de los bordes y la inclusión de áreas adicionales no presentes en la máscara original, un fenómeno similar al reportado en la arquitectura *U-Net*.

Para la clase 1, *SegNet* ha demostrado su capacidad para identificar y segmentar estructuras detalladas. No obstante, al comparar con la máscara de referencia, se evidencian discrepancias en las formas y áreas segmentadas. Es notable la ausencia de un órgano específico en esta predicción, el cual aparece erróneamente en la predicción de la clase 0, indicando una posible confusión en la clasificación entre estas dos clases.

En cuanto a la predicción de la clase 2, la red ha logrado segmentar una estructura específica, aunque se observan pequeñas imprecisiones en comparación con la máscara de referencia. La predicción para la clase 3 muestra resultados similares a los anteriores, con la particularidad de la adición de un pequeño órgano no presente en la máscara original. Por último, la predicción de la clase 4 demuestra la capacidad de *SegNet* para identificar la estructura correspondiente, presentando algunas diferencias en los bordes y detalles finos.

La arquitectura *U-Net* con *Backbone ResNet50* ha demostrado resultados variados en la seg-

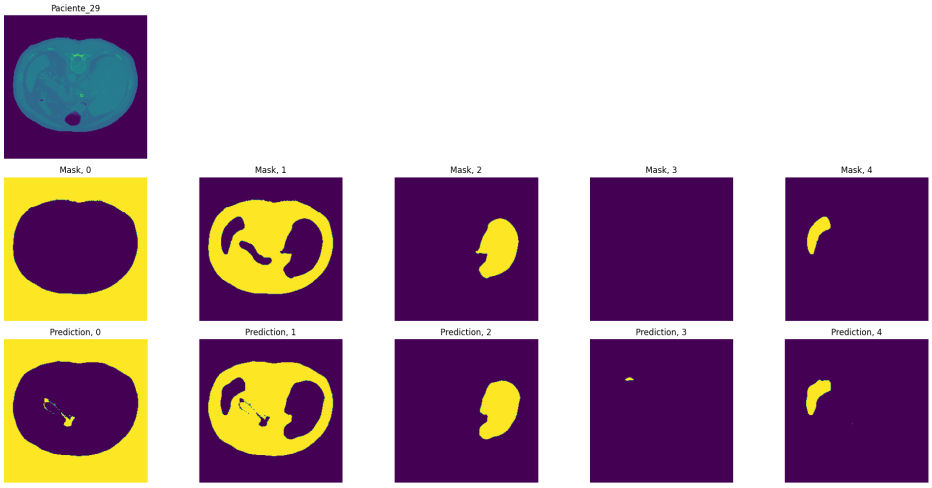

Figura 6.15: Inferencia No. 29 con *SegNet*. Fuente: *Autores*.

mentación de diferentes clases *(véase imagen 6.16)*. Para la clase 0, se observa una segmentación general de la región correspondiente, con una definición más fina y precisa de las áreas internas en comparación con la máscara de referencia. Sin embargo, se detecta una zona difusa que podría estar relacionada con un órgano de la clase 1, lo que sugiere cierta confusión en la delimitación entre estas dos clases.

Figura 6.16: Inferencia No. 29 con *U-Net* con *ResNet50*. Fuente: *Autores*.

En el caso de la clase 1, la red ha logrado identificar y segmentar estructuras detalladas con mayor precisión que la máscara de referencia. Esta mejora en la definición de los bordes y la captura de detalles finos destaca la capacidad de la arquitectura para extraer características complejas con

un órgano difusamente visible. La predicción para la clase 2 muestra una segmentación precisa y detallada de una estructura específica, manteniendo una alta similitud con la máscara de referencia. Esto indica un rendimiento robusto de la red en la identificación de esta clase en particular.

Para la clase 3, la comparación directa con la máscara de referencia revela que la predicción ha identificado dos puntos pequeños. Esta discrepancia sugiere que la red puede ser sensible a detalles mínimos o que podría estar detectando estructuras que no están presentes en la máscara original. La clase 4 presenta una segmentación detallada y precisa de la estructura correspondiente, demostrando la capacidad de la *U-Net* con *ResNet50* para capturar y reproducir con fidelidad las características específicas de esta clase.

La arquitectura *U-Net* con *ResNet50* fusiona las capacidades de segmentación de la *U-Net* tradicional con las potentes características de extracción de la *ResNet50*. Esta combinación mantiene la estructura fundamental de la *U-Net*, pero incorpora la *ResNet50* como codificador (encoder), lo que mejora significativamente la capacidad de la red para extraer características de alto nivel.

La precisión en la segmentación varía entre las diferentes clases. Se observan desafíos particulares en las clases 0, 1 y 3, similares a los reportados en arquitecturas como SegNet. Para las clases restantes, la *U-Net* con *ResNet50* muestra un alto grado de similitud con las máscaras de referencia, lo que indica un rendimiento sólido en estas categorías.

La implementación de una Red de Pirámide de Características (*FPN*) con un backbone ResNet50 ha demostrado resultados variados en la segmentación de imágenes médicas *(véase imagen 6.17)*. Esta arquitectura, diseñada para mejorar la capacidad de segmentación en imágenes a diferentes escalas, crea una pirámide de características que permite a la red manejar objetos de diversos tamaños de manera efectiva.

En el análisis de las predicciones para las diferentes clases, se observan resultados heterogéneos. Para la clase 0, la red ha logrado segmentar la región general correspondiente con una precisión notable, mostrando una segmentación más fina y detallada de las áreas internas en comparación con la máscara de referencia. De manera similar, la predicción para la clase 3 exhibe una segmentación precisa y bien delineada al compararla directamente con su máscara correspondiente.

Sin embargo, se han identificado desafíos en la segmentación de otras clases. La predicción para la clase 1 muestra que la *FPN* con ResNet50 ha identificado y segmentado las estructuras grandes, pero con una precisión limitada, evidenciando dificultades en la segmentación de órganos específicos. Para la clase 2, la red produce una segmentación borrosa y con presencia de ruido. La clase 4 presenta un caso similar, donde la predicción muestra una identificación imprecisa de la estructura específica, acompañada de ruido adicional.

Estas discrepancias entre las predicciones y las máscaras de referencia señalan áreas donde la

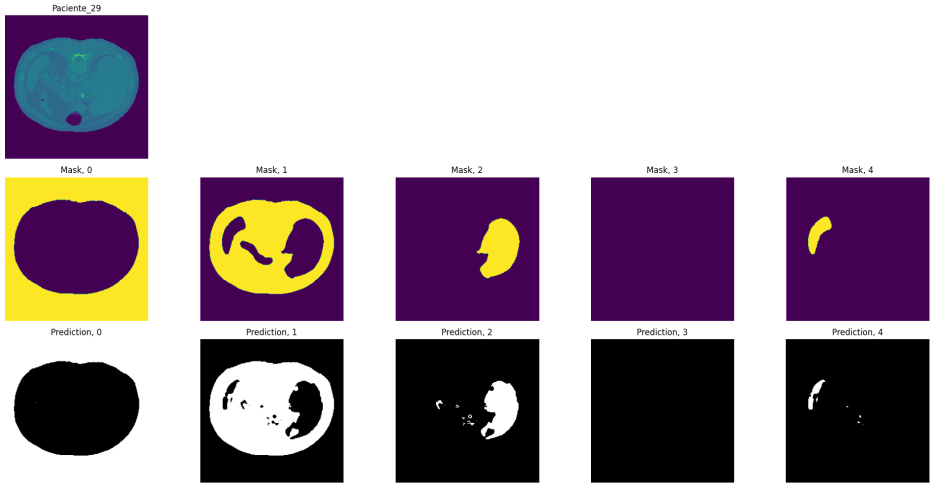

Figura 6.17: Inferencia No. 29 con *FPN* con *ResNet50*. Fuente: *Autores*.

red requiere mejoras. Las dificultades observadas, particularmente en las clases 1, 2 y 4, sugieren la necesidad de ajustes en la arquitectura, la incorporación de más datos de entrenamiento o la implementación de técnicas de post-procesamiento más avanzadas.

La precisión en la segmentación varía significativamente entre las diferentes clases. Mientras que las predicciones para las clases 0 y 3 muestran un alto grado de similitud con las máscaras de referencia, las clases 1, 2 y 4 presentan mayores errores y desafíos.

A la pregunta problema planteada en la sección 1.2 ¿Cuál modelo de segmentación demostró ser más eficiente para lograr una segmentación precisa y confiable en imágenes de tomografía computarizada *(CT)* del área abdominal, al exhibir los mejores resultados en términos de *precisión*, *recall*, y otras métricas pertinentes, a través de un análisis comparativo de diferentes modelos?

En los resultados encontramos que el modelo de segmentación que demostró ser más eficiente para lograr una segmentación precisa y confiable en imágenes de tomografía computarizada *(CT)* del área abdominal fue *U-Net* sin *Backbone*. Este modelo sobresalió en términos de precisión (*accuracy*), *DICE*, *F1 score*, *IoU score*, *precision* y *recall*, comparado con otras arquitecturas evaluadas, incluyendo *SegNet*, *U-Net* con *VGG16*, *U-Net* con *ResNet50*, *FPN* con *ResNet50* y *LinkNet* con *VGG16*.

Los resultados obtenidos para *U-Net* sin *Backbone* mostraron los siguientes puntajes: en entrenamiento, alcanzó un puntaje de 0.9383, y en validación, un puntaje de 0.9213. Estos puntajes reflejan una alta capacidad del modelo para generalizar sobre datos no vistos durante el entrenamiento, destacando su eficiencia y confiabilidad en la segmentación de imágenes médicas.

El entrenamiento de *U-Net* sin *Backbone* se realizó durante 250 épocas con un tiempo promedio por época de 67 segundos, resultando en un tiempo total de entrenamiento de aproximadamente 4.65 horas. Este tiempo de entrenamiento es razonable considerando el rendimiento del modelo y la precisión alcanzada en las tareas de segmentación.

La evolución de la función de pérdida y *accuracy* del modelo *U-Net* sin *Backbone* durante las 250 épocas se muestra en las figuras 6.2 y 6.3, respectivamente. La curva de pérdida para el entrenamiento (línea azul) y la validación (línea roja) mostraron una tendencia estable desde el inicio, con valores finales de 0.0210 (2.1 %) para el entrenamiento y 0.0390 (3.9 %) para la validación, indicando una diferencia mínima de 0.018 (1.8 %), lo que sugiere que no hay sobreajuste. La estabilidad y los bajos valores de esta curva indican una buena adaptación del modelo a los datos de entrenamiento, minimizando de manera efectiva el error en la predicción de las segmentaciones.

El gráfico de *accuracy* muestra que tanto la precisión en el conjunto de entrenamiento (línea azul) como en el de validación (línea roja) se mantienen estables desde la época 170, con valores finales de 0.9898 (98.98 %) para el entrenamiento y 0.9883 (98.83 %) para la validación, con una diferencia mínima de 0.0015 (0.15 %). Esta pequeña diferencia sugiere que el modelo no sufre de sobreajuste y tiene una gran capacidad de generalización.

Para complementar el análisis cuantitativo, se realizó una validación indirecta utilizando las primeras 30 imágenes del conjunto de pruebas, con el fin de evaluar visualmente las predicciones de segmentación semántica en comparación con las máscaras reales. Este análisis permitió verificar la coherencia de las predicciones a nivel de órganos, asegurando su validez frente a las máscaras de referencia. En la figura 6.14, se muestra un ejemplo de esta validación indirecta, donde se puede observar la imagen en escala de grises de una sección transversal del cuerpo de un paciente, la máscara *(ground truth)* y la predicción de cada clase realizada por el modelo *U-Net*.

La *U-Net* logró segmentar con precisión las diferentes clases, aunque se observaron algunas imprecisiones en los bordes y la inclusión de áreas adicionales no presentes en la máscara de referencia. Para la clase 0, se observó una falsa detección relacionada con un órgano correspondiente a la clase 1, lo que sugiere cierta confusión entre clases adyacentes o similares. No obstante, las predicciones para las clases 1, 2, 3 y 4 mostraron un alto grado de similitud con las máscaras de referencia, indicando un buen rendimiento del modelo en estas categorías.

En general, la *U-Net* sin *Backbone* demostró ser altamente efectiva y robusta para la tarea de segmentación de imágenes médicas, destacándose en todas las métricas evaluadas y mostrando una gran capacidad de generalización y precisión en la segmentación de órganos en imágenes de tomografía computarizada del área abdominal. Estos resultados son prometedores para aplicaciones futuras en entornos clínicos reales donde la precisión y la confiabilidad son fundamentales.

Para la pregunta de sistematización de la sección 1.2: ¿Cuáles fueron los tres modelos de *Deep*

Learning más eficaces y apropiados para la segmentación de las imágenes de tomografía computarizada *(CT)* del área abdominal, tomando en consideración la precisión y confiabilidad en los resultados de segmentación?

Para determinar cuáles fueron los tres modelos de *Deep Learning* más eficaces y apropiados para la segmentación de imágenes de tomografía computarizada (CT) del área abdominal, se consideraron varias métricas de desempeño, incluyendo *accuracy*, *IoU*, *Dice*, *F1*, *F2*, *precision* y *recall*, así como la estabilidad y confiabilidad en los resultados de segmentación. Además, se evaluaron las funciones de pérdida y *accuracy* tanto en las fases de entrenamiento como de validación, así como el tiempo total de entrenamiento de cada modelo. Con base en estas consideraciones, los tres modelos más destacados fueron:**U-Net**, **SegNet**, **U-Net con *Backbone* ResNet50**.

La arquitectura *U-Net* sin *Backbone* se destacó como el modelo más eficaz y confiable para la segmentación de imágenes CT del área abdominal. Este modelo mostró un desempeño sobresaliente en términos de precisión y confiabilidad de los resultados de segmentación, alcanzando un puntaje de entrenamiento de 0.9383 y un puntaje de validación de 0.9213. La *U-Net* logró mantener un alto *accuracy* de 0.9883 (98.83 %) en la fase de validación y 0.9898 (98.98 %) en la fase de entrenamiento, con una diferencia mínima de 0.15 %, lo que indica una buena capacidad de generalización y ausencia de sobreajuste.

El análisis de las funciones de pérdida y *accuracy* mostró una curva de pérdida estable y baja tanto en entrenamiento (0.0210) como en validación (0.0390), con una diferencia de apenas 1.8 %. Este comportamiento indica una buena adaptación del modelo a los datos de entrenamiento y una minimización efectiva del error en la predicción de las segmentaciones, como se observa en la Figura 6.2.

En cuanto a la validación indirecta, la *U-Net* mostró una segmentación precisa y detallada en las diferentes clases analizadas. Las predicciones para las clases 1, 2 y 3 presentaron un alto grado de similitud con las máscaras de referencia, mientras que la predicción para la clase 0 mostró más errores, sugiriendo que esta clase puede ser más desafiante para la red.

El segundo modelo más eficaz fue *SegNet*, que obtuvo un puntaje de entrenamiento de 0.9112 y un puntaje de validación de 0.8871. La *SegNet* demostró una *accuracy* de 0.9772 (97.72 %) en la fase de validación y 0.982 (98.2 %) en la fase de entrenamiento, con una diferencia de 0.48 %, indicando un buen equilibrio entre el aprendizaje y la generalización.

La curva de pérdida para *SegNet* mostró una disminución significativa y estabilización temprana en el entrenamiento, con valores finales de 0.0399 (3.99 %) para entrenamiento y 0.0561 (5.61 %) para validación. Esta estabilidad indica una buena adaptación del modelo a los datos de entrenamiento y una generalización aceptable en datos nuevos, como se muestra en la Figura 6.4.

En la validación indirecta, *SegNet* mostró una segmentación efectiva, aunque con variaciones en la precisión para las diferentes clases. Las predicciones para la clase 0 presentaron diferencias en la definición de los bordes y la inclusión de áreas adicionales no presentes en la máscara original, mientras que las predicciones para las clases 1, 2 y 3 mostraron un rendimiento robusto con ligeras imprecisiones en comparación con las máscaras de referencia.

La *U-Net* con *Backbone* ResNet50 se posicionó como el tercer modelo más eficaz, con un puntaje de entrenamiento de 0.8817 y un puntaje de validación de 0.8367. Este modelo alcanzó una *accuracy* de 0.9802 (98.02 %) en la fase de validación y 0.9867 (98.67 %) en la fase de entrenamiento, con una diferencia de 0.65 %, lo que indica una buena capacidad de generalización y estabilidad en los resultados de segmentación.

El análisis de la función de pérdida mostró una curva estable y baja, con valores finales de 0.049 (4.9 %) para entrenamiento y 0.0616 (6.16 %) para validación, lo que indica una buena adaptación del modelo a los datos de entrenamiento y una minimización efectiva del error en la predicción de las segmentaciones, como se observa en la Figura 6.8.

En la validación indirecta, la *U-Net* con *ResNet50* mostró resultados variados en la segmentación de diferentes clases. Las predicciones para las clases 1 y 2 presentaron una segmentación precisa y detallada con alta similitud con las máscaras de referencia, mientras que las predicciones para la clase 0 mostraron imprecisiones en los bordes y la inclusión de áreas adicionales no presentes en la máscara original. Las clases 3 y 4 también mostraron una segmentación precisa, aunque con pequeñas diferencias en los bordes y detalles finos.

Además de la precisión y confiabilidad en los resultados de segmentación, se consideraron los tiempos de entrenamiento y las funciones de pérdida y *accuracy*. La *U-Net* con *ResNet50* presentó el menor tiempo por época (29 segundos) y el menor tiempo total de entrenamiento (2.01 horas), lo que la hace una opción eficiente en términos de tiempo de entrenamiento. La *U-Net* sin *Backbone* tuvo un tiempo por época de 67 segundos y un tiempo total de entrenamiento de 4.65 horas, mientras que *SegNet* tuvo un tiempo por época de 55 segundos y un tiempo total de entrenamiento de 3.81 horas.

Los tres modelos más eficaces y apropiados para la segmentación de imágenes CT del área abdominal fueron *U-Net*, *SegNet* y *U-Net* con *Backbone* ResNet50. Estos modelos demostraron una alta precisión y confiabilidad en los resultados de segmentación, una buena capacidad de generalización y tiempos de entrenamiento razonables, lo que los hace adecuados para aplicaciones prácticas en el ámbito de la segmentación médica.

Para la pregunta de sistematización de la sección 1.2: ¿Cómo se efectuó la validación del modelo de segmentación obtenido, evaluando su desempeño en términos de *precisión*, *recall*, *coeficiente Dice*, *accuracy*, *IoU* y *loss*, garantizando su robustez y generalización en diferentes conjuntos de datos?

La validación del modelo de segmentación se realizó mediante un enfoque integral que incluyó la evaluación de tiempos de entrenamiento, el análisis de gráficas de *loss* y *accuracy*, el cálculo de múltiples métricas de desempeño y una validación indirecta visual. Este proceso garantizó una evaluación robusta y exhaustiva del modelo, asegurando su capacidad de generalización y precisión en diferentes conjuntos de datos.

Se llevó a cabo una comparación exhaustiva de las arquitecturas *U-Net*, *SegNet*, *FPN* y *LinkNet*, utilizando diferentes *Backbones* como *VGG16* y *ResNet50*. Cada modelo fue entrenado durante 250 épocas, y se registraron los tiempos de entrenamiento por época y el tiempo total para cada arquitectura, como se muestra en la Tabla 6.1.

Los resultados indican que la arquitectura *U-Net* con *Backbone ResNet50* es la más eficiente en términos de tiempo de entrenamiento, con un menor tiempo por época (29 segundos) y el menor tiempo total (2,01 horas). Para evaluar el comportamiento de los modelos durante el entrenamiento, se generaron dos gráficas clave: una mostrando la *accuracy* tanto en entrenamiento como en validación, y otra representando la función de pérdida en ambas fases. A continuación, se describe el análisis de estas gráficas para las arquitecturas principales.

Para evaluar el desempeño de los modelos, se utilizaron múltiples métricas, incluyendo *accuracy*, IoU, *Dice*, $F1$, $F2$, *precision* y *recall*. Estos indicadores permiten una evaluación integral de la calidad de las segmentaciones. Además de las evaluaciones cuantitativas, se realizó una validación indirecta utilizando las primeras 30 imágenes del conjunto de pruebas. Este análisis visual comparó las predicciones de segmentación semántica con las máscaras reales, permitiendo verificar la coherencia de las predicciones a nivel de órganos.

Al final, se compararon los puntajes generales de todas las arquitecturas, considerando tanto el entrenamiento como la validación. La Tabla 6.14 muestra como estos resultados, la arquitectura *U-Net* sin *Backbone* mostró el mejor desempeño tanto en entrenamiento como en validación, destacándose como la opción más robusta y eficiente para la tarea de segmentación semántica en las condiciones evaluadas.

¿Qué técnicas y algoritmos se evidenciaron como los más adecuados para llevar a cabo un preprocesamiento exhaustivo de las imágenes de tomografía computarizada *(CT)* del área abdominal, con la finalidad de eliminar el ruido y optimizar la calidad de las imágenes?

Este enfoque integral de preprocesamiento, que incluye la corrección de artefactos, la normalización y estandarización, y la ampliación de datos, resulta en la optimización de la calidad de las imágenes de CT del área abdominal. Esto garantiza que los modelos de segmentación reciban datos de alta calidad y uniformidad, mejorando así la precisión y eficacia de los resultados obtenidos.

La corrección de artefactos es un paso crucial en el preprocesamiento de imágenes médicas, ya que los artefactos pueden distorsionar significativamente los resultados de la segmentación y el aná-

lisis subsiguiente. En el contexto de las imágenes *CT*, los artefactos pueden incluir ruido de fondo, distorsiones debidas a movimientos del paciente, y otros elementos no deseados como la camilla o mantas.

Se desarrolló una función en *Python* denominada *clean_noise_from_mask()* que tiene como objetivo limpiar el ruido de la matriz de máscara. Esta función identifica y elimina elementos no deseados, como la camilla, que suelen aparecer en la parte superior de las imágenes. La función itera sobre las filas de la matriz de máscara desde un índice calculado en base a un porcentaje proporcionado hacia atrás, procesando las columnas de cada fila en esa sección de la máscara. Las columnas que cumplen ciertas condiciones (suma mayor a 0 y menor o igual a 3) son limpiadas, reemplazando los valores de 1 por 0.

Para asegurar una eliminación eficiente del ruido, se realizó una segunda validación mediante la función *verify_noise_removal()*, la cual repite el proceso de *clean_noise_from_mask()*. Finalmente, se utilizó la función *apply_mask_to_image()* para aplicar la máscara limpia sobre la imagen, asignando un valor de -1042 a las áreas de fondo (background), garantizando así que no quede ningún residuo no deseado en la imagen.

La técnica de *data augmentation* se implementó utilizando la librería *imgaug* de *Python*, la cual proporciona una amplia gama de transformaciones geométricas y ajustes para mejorar y diversificar el conjunto de datos de entrenamiento. Se aplicaron rotaciones en un rango de -10 a 10 grados para simular variaciones en la posición del paciente durante la adquisición de imágenes.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

El estudio realizado se centró en evaluar y comparar diferentes modelos de segmentación en imágenes de tomografía computarizada (*CT*) del área abdominal para determinar el modelo más efectivo en términos de precisión, *recall*, coeficiente *Dice*, *accuracy*, *IoU* y *loss*. El análisis se realizó considerando un procesamiento previo exhaustivo de las imágenes, la implementación y entrenamiento de múltiples modelos de *Deep Learning*, y una validación integral del modelo seleccionado.

Para facilitar el proceso de segmentación, se realizó un preprocesamiento detallado de las imágenes de *CT*. Se desarrollaron funciones específicas en Python para eliminar el ruido de las imágenes, incluyendo la identificación y eliminación de artefactos no deseados como camillas y objetos de fondo.

Se aplicaron técnicas de normalización para ajustar la escala de las imágenes y estandarización para uniformar los datos, asegurando así la consistencia y calidad de las imágenes. Además, se implementaron transformaciones geométricas y ajustes para aumentar la variabilidad del conjunto de datos de entrenamiento. Las transformaciones incluyeron rotaciones y cambios en el brillo y contraste, lo que ayudó a mejorar la robustez y generalización de los modelos.

Se implementaron y entrenaron seis modelos definidos de *Deep Learning* para la segmentación de las imágenes de *CT* del área abdominal. Los modelos evaluados fueron: *U-Net*, *SegNet*, *FPN* y *LinkNet*, utilizando diversos *backbones* como *VGG16* y *ResNet50*. Cada modelo se entrenó durante 250 épocas, y se registraron los tiempos de entrenamiento y los resultados de las métricas clave. Los hallazgos específicos fueron:

- *U-Net*: Demostró ser el modelo más eficiente y confiable, con una alta precisión y capacidad de generalización. La *U-Net* sin *Backbone* obtuvo un puntaje de entrenamiento de 0.9383 y un puntaje de validación de 0.9213, destacándose en todas las métricas evaluadas.
- *SegNet*: Fue el segundo modelo más eficaz, con un puntaje de entrenamiento de 0.9112 y un puntaje de validación de 0.8871. Aunque mostró un buen equilibrio entre el aprendizaje y la generalización, presentó algunas variaciones en la precisión de las clases segmentadas.
- *U-Net* con *ResNet50*: Se posicionó como el tercer modelo más efectivo, con un puntaje de entrenamiento de 0.8817 y un puntaje de validación de 0.8367. Este modelo logró un buen

desempeño general, aunque con algunas imprecisiones en ciertas clases.

- *FPN* con *ResNet50* y *LinkNet* con *VGG16*: Aunque estas arquitecturas mostraron un desempeño aceptable, sus puntajes fueron inferiores a los de *U-Net* y *SegNet*, indicando una menor efectividad en la segmentación de las imágenes evaluadas.

La validación del modelo de segmentación se realizó mediante un enfoque integral que incluyó: análisis de gráficas de *Loss* y *Accuracy* donde se generaron gráficos para evaluar el comportamiento de los modelos durante el entrenamiento y la validación. Las curvas de pérdida y precisión se utilizaron para identificar el sobreajuste y la capacidad de generalización de los modelos. Se calcularon múltiples métricas, incluyendo precisión, *IoU*, *Dice*, *F2*, precisión y *recall*, para una evaluación integral de la calidad de las segmentaciones.

Finalmente, se realizó una validación indirecta visual donde se compararon las predicciones de segmentación con las máscaras reales utilizando las primeras 30 imágenes del conjunto de pruebas. Este análisis visual permitió verificar la coherencia de las predicciones a nivel de órganos y evaluar la efectividad del modelo en condiciones prácticas.

El estudio concluye que el modelo *U-Net* sin *Backbone* es el más eficiente y confiable para la segmentación de imágenes de *CT* del área abdominal, seguido de cerca por *SegNet* y *U-Net* con *ResNet50*. Este modelo demostró un desempeño sobresaliente en todas las métricas evaluadas y mostró una gran capacidad de generalización y precisión en la segmentación de órganos en imágenes médicas. Los resultados obtenidos son prometedores para aplicaciones futuras en entornos clínicos reales, donde la precisión y la confiabilidad son fundamentales.

El proceso de validación integral y el preprocesamiento exhaustivo de las imágenes fueron vitales para garantizar la robustez y efectividad de los modelos evaluados. Las técnicas de corrección de artefactos, normalización, estandarización y *data augmentation* fueron clave para mejorar la calidad de los datos y optimizar el rendimiento de los modelos de segmentación.

7.2. Trabajos Futuros

En esta sección se plantean diversas líneas de investigación y desarrollo que podrían expandir y mejorar los resultados obtenidos en este trabajo. Estas líneas se enfocan en la exploración de arquitecturas avanzadas de segmentación, el uso de *backbones* más adecuados, y la integración de nuevas técnicas y modelos para optimizar la segmentación de imágenes de tomografía computarizada (*CT*) del área abdominal.

Un área prometedora para futuros trabajos es la implementación de arquitecturas de *instance segmentation*. La segmentación de instancias es una extensión de la segmentación semántica que no solo clasifica cada píxel de una imagen en una categoría, sino que también distingue entre diferentes

objetos de la misma clase.

En el contexto de la segmentación de imágenes médicas, esto podría ser útil para diferenciar entre múltiples lesiones o tumores en una sola imagen. Modelos como *Mask R-CNN*, que extiende *Faster R-CNN* añadiendo una rama para predecir una máscara de segmentación para cada objeto detectado, y *SOLOv2 (Segmenting Objects by Locations)*, que predice las máscaras directamente a partir de ubicaciones espaciales, han demostrado ser efectivos en diversas aplicaciones, incluyendo imágenes médicas [40].

Además, es importante considerar la exploración de arquitecturas más avanzadas. Las arquitecturas actuales utilizadas en este estudio, como *U-Net* y *SegNet*, han mostrado un rendimiento sólido. Sin embargo, existen modelos más recientes y avanzados que podrían ofrecer mejoras significativas en precisión y eficiencia. Por ejemplo, los *Swin Transformer*, una arquitectura eficiente y escalable basada en transformadores, ha mostrado un rendimiento superior en tareas de segmentación de imágenes [25]. Asimismo, *nnU-Net* es una extensión automatizada de *U-Net* que adapta automáticamente su configuración para diferentes tareas de segmentación médica, ofreciendo un rendimiento líder en múltiples desafíos de segmentación.

El uso de *backbones* más poderosos puede mejorar significativamente el rendimiento de los modelos de segmentación. Los *backbones* más avanzados pueden extraer características más ricas y discriminativas de las imágenes de entrada. *EfficientNet*, una familia de modelos diseñados mediante una búsqueda automatizada de arquitectura (*NAS*), ha mostrado ser altamente eficiente en términos de precisión y uso de recursos [38]. *RegNet*, basado en principios de diseño que permiten un escalado uniforme, ha demostrado un excelente rendimiento en varias tareas de visión por computador.

La aplicación de técnicas de *data augmentation* avanzadas es necesaria para mejorar la robustez y generalización de los modelos de segmentación. Técnicas más avanzadas y específicas pueden simular mejor las variaciones presentes en las imágenes médicas. Por ejemplo, el uso de *Generative Adversarial Networks (GANs)* para generar nuevas imágenes sintéticas que aumenten el conjunto de datos de entrenamiento [13], y técnicas como *MixUp* y *CutMix*, que combinan múltiples imágenes para crear nuevos ejemplos de entrenamiento, pueden ayudar a mejorar la robustez del modelo.

Para una evaluación más integral del rendimiento del modelo, se pueden considerar métricas adicionales que capturen diferentes aspectos de la calidad de la segmentación. Métricas como el *Pixel Accuracy (PA)*, la proporción de píxeles correctamente predichos sobre el total de píxeles, y el *Average Precision (AP)*, utilizada comúnmente en la evaluación de modelos de detección de objetos, podría adaptarse para evaluar la segmentación de instancias.

Una dirección adicional y altamente innovadora para futuros trabajos es la implementación del *Segment Anything Model (SAM)*, un modelo de segmentación universal desarrollado recientemente.

SAM está diseñado para segmentar cualquier objeto en cualquier imagen sin necesidad de ajustes específicos para cada tarea.

La integración de *SAM* en la segmentación de imágenes de *CT* abdominal podría simplificar significativamente el proceso de segmentación, mejorando la eficiencia y reduciendo la necesidad de ajustes manuales. Su capacidad para segmentar cualquier objeto con precisión podría ser particularmente útil para identificar estructuras anatómicas y patológicas con mayor exactitud.

Además, si se obtiene un *dataset* de tomografía computacional del área abdominal de tres canales, se podría realizar un reentrenamiento de las arquitecturas aprovechando los pesos de los *backbones* de *ImageNet*, dando paso al *transfer learning*. Esta técnica permitiría adaptar los modelos preentrenados a las características específicas del nuevo conjunto de datos, mejorando así la precisión y la robustez de las segmentaciones.

Explorar estas direcciones futuras no solo fortalecerá los hallazgos de este estudio, sino que también abrirá nuevas oportunidades para mejorar la precisión y eficiencia de los modelos de segmentación de imágenes médicas. La integración de arquitecturas avanzadas, *backbones* más poderosos, y técnicas innovadoras de *data augmentation* promete mejorar significativamente los resultados obtenidos, beneficiando así tanto a la investigación médica como a la práctica clínica.

Capítulo 8

Anexos

8.1. Repositorio de GitHub

Para consultar los distintos códigos de python, funciones y archivos de Keras (`*.Keras`) del trabajo de grado de maestría "Análisis comparativo de Modelos de Segmentación en imágenes de tomografía computarizada (CT) del área abdominal" revisar el siguiente enlace de GitHub, es un repositorio público, revisar el documento *README.md* para mas información.

8.2. Formula de Puntaje

$$Puntaje = \frac{accuracy + dice + F2 + IoU + (1 - loss) + precision + recall}{7}$$

$$Puntaje = \frac{TP + TN}{TP + TN + FP + FN} + \frac{2TP}{2TP + FP + FN} + \frac{5TP}{5TP + 4FN + FP} + \\ + \frac{TP}{TP + FP + FN} + (1 - loss) + \frac{TP}{TP + FP} + \frac{TP}{TP + FN}$$

8.3. Códigos

Listing 8.1: Implementación de *Data Augmentation* con *imgaug*

```
def apply_data_augmentation(images_list, output_folder="augmented_images"):
    os.makedirs(output_folder, exist_ok=True)

    for i, image_path in enumerate(images_list):
        ct_scan = np.load(image_path)

        augmenter = iaa.Sequential([
            iaa.Affine(rotate=(-10, 10)),    # Rotación de la imagen
            iaa.Fliplr(0.5),                 # Volteos horizontales
            iaa.Crop(percent=(0, 0.1))       # Recortar imágenes
        ])

        # Aplicar la ampliación de datos a la imagen de tomografía actual
        augmented_ct_scan = augmenter(images=ct_scan)
```

```
15
16          output_path = os.path.join(output_folder, ...
                f'train_{str(i).zfill(3)}_augmented.npy')
17          np.save(output_path, augmented_ct_scan)
18
19          # Mostrar la imagen original y la imagen ampliada
20          fig, ax = plt.subplots(1, 2, figsize=(12, 6))
21          ax[0].imshow(ct_scan, cmap='gray')
22          ax[0].set_title(f'CT Scan Original - Paciente {i}')
23          ax[1].imshow(augmented_ct_scan, cmap='gray')
24          ax[1].set_title(f'CT Scan Ampliado - Paciente {i}')
25          plt.savefig(f'train_{str(i).zfill(3)}_augmented.png')
26          plt.show()
```

Listing 8.2: Codigo de las arquitecturas usando librería `segmentation_models` [21]

```
1
2  if selector == 0:
3      model = sm.Unet(BACKBONE, input_shape=(IMG_HEIGHT, IMG_WIDTH, 1), ...
           classes=n_classes, activation='softmax',encoder_weights=None)
4      name_model = "unet"
5  elif selector == 1:
6      model = sm.PSPNet(BACKBONE, input_shape=(IMG_HEIGHT, IMG_WIDTH, 1), ...
           classes=n_classes, activation='softmax',encoder_weights=None)
7      name_model = "PSPNet"
8  elif selector == 2:
9      model = sm.FPN(BACKBONE, input_shape=(IMG_HEIGHT, IMG_WIDTH, 1), ...
           classes=n_classes, activation='softmax',encoder_weights=None)
10     name_model = "FPN"
11 elif selector == 3:
12     model = sm.Linknet(BACKBONE, input_shape=(IMG_HEIGHT, IMG_WIDTH, 1), ...
           classes=n_classes, activation='softmax',encoder_weights=None)
13     name_model = "Linknet"
14
15 name_checkpoint = f'{current_path}/{name_model}_5_classes.keras'
16
17 keras_metrics = ['accuracy',
18                  sm.metrics.iou_score,
19                  sm.metrics.dice_score,
20                  sm.metrics.f1_score,
21                  sm.metrics.f2_score,
22                  sm.metrics.precision,
23                  sm.metrics.recall,
24                  ]
25
26 model.compile(optimizer=Adam(learning_rate=1e-3),
27               loss='categorical_crossentropy',
28               metrics=keras_metrics)
29 model.summary()
```

Listing 8.3: Arquitectura *U-Net* [34]

```
1
2  #Definir bloques convolucionales.
3  def conv_block(input, num_filters):
4      x = Conv2D(num_filters, 3, padding="same")(input)
5      x = BatchNormalization()(x) # Se usa para normalizar las salidas de las capas
6      x = Activation("relu")(x)
7
8      x = Conv2D(num_filters, 3, padding="same")(x)
9      x = BatchNormalization()(x)
10     x = Activation("relu")(x)
11
12     return x
13
14 #Definir bloques convolucionales + max pooling.
15 def encoder_block(input, num_filters):
16     x = conv_block(input, num_filters)
17     p = MaxPool2D((2, 2))(x)
18     return x, p
19
20 # Definir bloques del decodificador.
21 def decoder_block(input, skip_features, num_filters):
22     x = Conv2DTranspose(num_filters, (2, 2), strides=2, padding="same")(input)
23     x = Concatenate()([x, skip_features])
24     x = conv_block(x, num_filters)
25     return x
26
27 #UNet
28 def build_unet(n_classes,input_shape):
29     inputs = Input(input_shape) #256x256
30
31     # Codificador
32     s1, p1 = encoder_block(inputs, 64)  # Salida: 128x128x64
33     s2, p2 = encoder_block(p1, 128)     # Salida: 64x64x128
34     s3, p3 = encoder_block(p2, 256)     # Salida: 32x32x256
35     s4, p4 = encoder_block(p3, 512)     # Salida: 16x16x512
36
37     # Puente (cuello de botella)
38     b1 = conv_block(p4, 1024)           # Salida: 16x16x1024
39
40     # Decodificador
41     d1 = decoder_block(b1, s4, 512)
42     d2 = decoder_block(d1, s3, 256)
43     d3 = decoder_block(d2, s2, 128)
44     d4 = decoder_block(d3, s1, 64)
45
46     outputs = Conv2D(n_classes, (1, 1), padding="same", activation="softmax")(d4) ...
           #Multiclase
47
48     model = Model(inputs=[inputs], outputs=[outputs], name="U-Net")
```

```
49      return model
```

Listing 8.4: Arquitectura *SegNet* [4]

```
1
2  def conv_block(input, num_filters):
3      x = Conv2D(num_filters, 3, padding="same")(input)
4      x = BatchNormalization()(x)
5      x = Activation("relu")(x)
6
7      x = Conv2D(num_filters, 3, padding="same")(x)
8      x = BatchNormalization()(x)
9      x = Activation("relu")(x)
10
11     return x
12
13 # Definir bloques convolucionales + max pooling para el codificador.
14 def encoder_block(input, num_filters):
15     x = conv_block(input, num_filters)
16     p = MaxPooling2D((2, 2))(x)
17     return x, p
18
19 # Definir bloques de upsample para el decodificador.
20 def decoder_block(input, num_filters):
21     x = UpSampling2D((2, 2))(input)
22     x = conv_block(x, num_filters)
23     return x
24
25 # Construir la arquitectura SegNet.
26 def build_segnet(n_classes, input_shape):
27     inputs = Input(input_shape)
28
29     # Codificador
30     s1, p1 = encoder_block(inputs, 64)
31     s2, p2 = encoder_block(p1, 128)
32     s3, p3 = encoder_block(p2, 256)
33     s4, p4 = encoder_block(p3, 512)
34
35     # Decodificador
36     d1 = decoder_block(p4, 512)
37     d2 = decoder_block(d1, 256)
38     d3 = decoder_block(d2, 128)
39     d4 = decoder_block(d3, 64)
40
41     outputs = Conv2D(n_classes, (1, 1), padding="same", activation="softmax")(d4)
42
43     model = Model(inputs=[inputs], outputs=[outputs], name="SegNet")
44     return model
```

Bibliografía

[1] PENG AN, YUROU XU, and PANPAN WU. Miccai flare23 challenge-attention mechanism-based deep supervision network for abdominal multi-organ segmentation. In *MICCAI 2023 FLARE Challenge*, 2023.

[2] José Denes Lima Araújo, Luana Batista da Cruz, Jonnison Lima Ferreira, Otilio Paulo da Silva Neto, Aristófanes Corrêa Silva, Anselmo Cardoso de Paiva, and Marcelo Gattass. An automatic method for segmentation of liver lesions in computed tomography images using deep neural networks. *Expert Systems with Applications*, 180:115064, 2021.

[3] Yodit Abebe Ayalew, Kinde Anlay Fante, and Mohammed Aliy Mohammed. Modified u-net for liver cancer segmentation from computed tomography images with a new class balancing method. *BMC Biomedical Engineering*, 3:1–13, 2021.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.

[5] Jinzheng Cai, Le Lu, Zizhao Zhang, Fuyong Xing, Lin Yang, and Qian Yin. Pancreas segmentation in mri using graph-based decision fusion on convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 442–450. Springer, 2016.

[6] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, December 2017.

[7] Kuan-bing Chen, Ying Xuan, Ai-jun Lin, and Shao-hua Guo. Lung computed tomography image segmentation based on u-net network fused with dilated convolution. *Computer Methods and Programs in Biomedicine*, 207:106170, 2021.

[8] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[9] CloudFactory. Feature pyramid network (fpn), 2024. Accessed: 2024-06-25.

[10] Albert Comelli, Claudia Coronnello, Navdeep Dahiya, Viviana Benfante, Stefano Palmucci, Antonio Basile, Carlo Vancheri, Giorgio Russo, Anthony Yezzi, and Alessandro Stefano. Lung segmentation on high-resolution computerized tomography images using deep learning: a preliminary step for radiomics studies. *Journal of Imaging*, 6(11):125, 2020.

[11] Luana Batista da Cruz, Jose Denes Lima Araujo, Jonnison Lima Ferreira, Joao Otavio Bandeira Diniz, Aristofanes Correa Silva, João Dallyson Sousa de Almeida, Anselmo Cardoso de Paiva, and Marcelo Gattass. Kidney segmentation from computed tomography images using deep neural network. *Computers in Biology and Medicine*, 123:103906, 2020.

[12] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[14] Intisar Rizwan I Haque and Jeremiah Neubert. Deep learning approaches to biomedical image segmentation. *Informatics in Medicine Unlocked*, 18:100297, 2020.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Sheng He, Rina Bao, Jingpeng Li, P Ellen Grant, and Yangming Ou. Accuracy of segment-anything model (sam) in medical image segmentation tasks. *arXiv preprint arXiv:2304.09324*, 2023.

[17] Mauricio Hidalgo, Bryan Ortiz, Ignacio Vera, and Manuel Escalona. Deep learning aplicado para la detección de hemorragias y tumores cerebrales. *AtoZ: novas práticas em informação e conhecimento*, 10:1, 12 2021.

[18] Godfrey N Hounsfield. Computerized transverse axial scanning (tomography): Part 1. description of system. *The British journal of radiology*, 46(552):1016–1022, 1973.

[19] Chuanfei Hu and Xinde Li. When sam meets medical images: An investigation of segment anything model (sam) on multi-phase liver tumor segmentation. *arXiv preprint arXiv:2304.08506*, 2023.

[20] Yuhao Huang, Xin Yang, Lian Liu, Han Zhou, Ao Chang, Xinrui Zhou, Rusi Chen, Junxuan Yu, Jiongquan Chen, Chaoyu Chen, et al. Segment anything model for medical images? *arXiv preprint arXiv:2304.14660*, 2023.

[21] Pavel Iakubovskii. Segmentation models. `https://github.com/qubvel/segmentation_models`, 2019.

[22] Ali F Khalifa and Eman Badr. Deep learning for image segmentation: A focus on medical imaging. *CMC-COMPUTERS MATERIALS & CONTINUA*, 75(1):1995–2024, 2023.

[23] Chae Eun Lee, Minyoung Chung, and Yeong-Gil Shin. Voxel-level siamese representation learning for abdominal multi-organ segmentation. *Computer methods and programs in biomedicine*, 213:106547, 2022.

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[26] Jun Ma and Bo Wang. *Fast and Low-Resource Semi-supervised Abdominal Organ Segmentation: MICCAI 2022 Challenge, FLARE 2022, Held in Conjunction with MICCAI 2022, Singapore, September 22, 2022, Proceedings*, volume 13816. Springer Nature, 2023.

[27] Jun Ma and Bo Wang. Segment anything in medical images. *arXiv preprint arXiv:2304.12306*, 2023.

[28] Jun Ma, Yao Zhang, Song Gu, Xingle An, Zhihe Wang, Cheng Ge, Congcong Wang, Fan Zhang, Yu Wang, Yinan Xu, et al. Fast and low-gpu-memory abdomen ct organ segmentation: the flare challenge. *Medical Image Analysis*, 82:102616, 2022.

[29] Jun Ma, Yao Zhang, Song Gu, Xingle An, Zhihe Wang, Cheng Ge, Congcong Wang, Fan Zhang, Yu Wang, Yinan Xu, Shuiping Gou, Franz Thaler, Christian Payer, Darko tern, Edward G.A. Henderson, Dónal M. McSweeney, Andrew Green, Price Jackson, Lachlan McIntosh, Quoc-Cuong Nguyen, Abdul Qayyum, Pierre-Henri Conze, Ziyan Huang, Ziqi Zhou, Deng-Ping Fan, Huan Xiong, Guoqiang Dong, Qiongjie Zhu, Jian He, and Xiaoping Yang. Fast and low-gpu-memory abdomen ct organ segmentation: The flare challenge. *Medical Image Analysis*, 82:102616, 2022.

[30] Jun Ma, Yao Zhang, Song Gu, Cheng Zhu, Cheng Ge, Yichi Zhang, Xingle An, Congcong Wang, Qiyuan Wang, Xin Liu, Shucheng Cao, Qi Zhang, Shangqing Liu, Yunpeng Wang, Yuhui Li, Jian He, and Xiaoping Yang. Abdomenct-1k: Is abdominal organ segmentation a solved problem?, 2021.

[31] Arturo Orellana García and Luis Manuel García Portal. Técnicas de segmentación y procesamiento para la detección de carcinomas renales en imágenes de tomografía abdominal. *Revista Cubana de Informática Médica*, 12(2), 2020.

[32] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[33] Pavlo Radiuk. Applying 3d u-net architecture to the task of multi-organ segmentation in computed tomography. *Applied Computer Systems*, 25(1):43–50, 2020.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[35] Peilun Shi, Jianing Qiu, Sai Mu Dalike Abaxi, Hao Wei, Frank P-W Lo, and Wu Yuan. Generalist vision foundation models for medical imaging: A case study of segment anything model on zero-shot medical segmentation. *Diagnostics*, 13(11):1947, 2023.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[37] Abdel Aziz Taha and Allan Hanbury. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC medical imaging*, 15(1):1–28, 2015.

[38] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[39] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and R Summers. Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *IEEE CVPR*, volume 7, page 46. sn, 2017.

[40] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation, 2020.

[41] Melanie Yusta Gómez, Marlen Pérez Díaz, Rubén Orozco Morales, and Xiomara Plasencia Hernández. Segmentación del hígado en imágenes de tomografía computarizada. *MediSur*, 20(2):257–271, 2022.

[42] Fan Zhang, Yu Wang, and Hua Yang. Efficient context-aware network for abdominal multi-organ segmentation. *arXiv preprint arXiv:2109.10601*, 2021.

[43] Zhuotun Zhu, Yingda Xia, Wei Shen, Elliot Fishman, and Alan Yuille. A 3d coarse-to-fine framework for volumetric medical image segmentation. In *2018 International conference on 3D vision (3DV)*, pages 682–690. IEEE, 2018.