# SISTEMA DE RECONOCIMIENTO Y SEGMENTACIÓN DE HERIDAS PARA SUTURA


Trabajo de Grado en Ingeniería Automática Industrial

SERGIO BASTIDAS LOSADA
HUGO NORBEY QUISOBONI IJAJI

Director: Ph.D(c) HERMES FABIÁN VARGAS ROSERO
(Universidad del Cauca)
Codirectora: MSc. ELENA MUÑOZ ESPAÑA
(Universidad del Cauca)

*Universidad del Cauca*
Facultad de Ingeniería Electrónica y Telecomunicaciones
PREGRADO EN INGENIERÍA AUTOMÁTICA INDUSTRIAL
Popayán 2023

# Agradecimientos

*... A Dios por guiarnos, cuidarnos e iluminarnos en todo momento. Por darnos la fortaleza y la sabiduría necesarias para culminar esta etapa de nuestra trayectoria académica.*

*... A nuestras madres, quienes con su amor, esfuerzo, acompañamiento constante, apoyo incondicional y dedicación a nuestras vidas, nos brindaron la oportunidad de acceder a una educación de alta calidad y nos guiaron en cada paso que dimos en vía hacia nuestro desarrollo personal y profesional.*

*... A nuestros familiares y amigos por su constante acompañamiento, sabios consejos y por sus buenos ánimos en cada paso que dimos para llegar aquí.*

*... A nuestro director de proyecto, el ingeniero Hermes Fabián Vargas, por su guía y orientación fundamental para el desarrollo de esta investigación. Su compromiso y apoyo con nuestro trabajo fue una fuente constante de motivación.*

*... A nuestra codirectora de proyecto, la ingeniera Elena Muñoz España, por su apoyo constante y sus valiosas correcciones que con su conocimiento y experiencia enriquecieron nuestra comprensión en nuevas temáticas.*

*... A todos los docentes de nuestra alma mater, quienes a lo largo de nuestra formación académica nos han brindado importantes conocimientos que enriquecen tanto nuestra vida profesional como personal, siendo ejemplos de un constante esfuerzo y compromiso.*

*... A nuestra alma mater, la Universidad del Cauca, por proporcionarnos educación de alta calidad, por su constante lucha en defensa de nuestros derechos y la búsqueda de una educación integral.*

# Resumen

En este proyecto de grado, se aborda el desarrollo de un sistema de visión artificial basado en técnicas de procesamiento digital de imágenes y Machine Learning (ML), con un enfoque específico en el uso de Convolutional Neural Networks (CNN) para el reconocimiento y segmentación de heridas corto punzantes. La investigación se centra en comparar tres arquitecturas de segmentación de imágenes: U-Net2D, SegNet y DeepLabV3+ con ResNet50 como codificador. Un aspecto distintivo de este trabajo radica en su enfoque específico en la segmentación de heridas corto punzantes, un área que ha recibido poca atención previa. Esta investigación busca aportar a esta brecha de conocimiento al abordar específicamente este tipo de heridas, con el propósito de garantizar que el sistema sea robusto ante variaciones de iluminación, rotación y desplazamiento del objetivo, lo que lo hace apto para aplicaciones médicas, como la sutura autónoma, contribuyendo así al campo de la visión artificial en medicina.

Se entrenan los modelos U-Net2D, SegNet y DeepLabV3+ en un conjunto de datos especializado, recopilado de la web y etiquetado usando **Hasty.ai**. Diseñado específicamente para la segmentación de heridas corto-punzantes. Las métricas de entrenamiento y validación como Dice, IoU, Precision y Recall demuestran que el modelo DeepLabV3+ es superior en términos de generalización, mientras que los dos modelos restantes tienden al sobre ajuste.

Se realizan inferencias en tiempo real para determinar la robustez de los modelos ante variaciones o disturbios. Los resultados confirman al modelo DeepLabV3+ como candidato óptimo para aplicaciones médicas de segmentación y resaltan que U-Net2D puede llegar a ser superior en la detección de detalles finos. Respecto a SegNet, se considera poco preciso en la detección de bordes y extremadamente sensible ante la variabilidad de los datos, por lo que no se considera viable para aplicaciones médicas. Este trabajo contribuye al avance del conocimiento en el campo de la visión artificial en medicina y abre nuevas posibilidades para mejorar la precisión y eficacia en el tratamiento de heridas corto punzantes.

**Palabras clave: Visión artificial, heridas corto-punzantes, imágenes médicas 2D, Machine Learning, Convolutional Neural Networks, segmentación semántica.**

# Índice general

# Índice de tablas

## INTRODUCCIÓN

Cuando se consulta a profesionales médicos sobre los factores más críticos para lograr una atención exitosa del paciente, dos conceptos se destacan: conocimiento y experiencia. Estos dos aspectos son esenciales para comprender la inteligencia artificial (IA) y sus implicaciones en la medicina. La IA se destaca por su capacidad de acumular experiencia y recopilar datos a través del análisis de información, lo que permite al sistema tomar decisiones fundamentadas basadas en el conocimiento adquirido a partir de las características significativas extraídas [1]. Por lo tanto, la IA encuentra aplicaciones en el ámbito médico, especialmente en el campo de la visión artificial, donde su objetivo principal es entrenar sistemas computacionales para emular la visión humana y comprender los objetos circundantes. Con este propósito, la visión artificial utiliza algoritmos especializados para procesar imágenes médicas, lo que posibilita realizar diagnósticos más rápidos y precisos que los que podría llevar a cabo un profesional médico [2]. Actualmente, la visión artificial en medicina tiene un campo de aplicación inmenso, ya que aproximadamente el 90 % de los datos médicos se componen de imágenes [3].

Este trabajo de grado se divide en cuatro capítulos que abordan diferentes aspectos de la investigación. En particular, se enfoca en aportar a la creciente área de la visión artificial aplicada a la medicina. Los aportes específicos de esta investigación incluyen la creación de un sistema de visión artificial especializado en la segmentación de heridas corto punzantes, abordando una brecha de conocimiento existente en esta área. Esta investigación también destaca la importancia de la robustez del sistema de visión ante variaciones y su potencial aplicación en el campo médico, específicamente en procedimientos de sutura autónoma.

- Capítulo 1: Generalidades. Este capítulo presenta el planteamiento del problema y los aspectos teóricos necesarios en torno a la inteligencia artificial y redes neuronales convolucionales.
- Capítulo 2: Desarrollo del Sistema de Reconocimiento y Segmentación de Heridas. Este capítulo describe el desarrollo y la implementación de modelos de segmentación de imágenes médicas seleccionados: U-Net2D, SegNet y DeepLabV3+. Este proceso se describe en detalle, abarcando desde la obtención y preparación de los datos hasta la definición, entrenamiento y validación de los modelos.
- Capítulo 3: Evaluación del desempeño de los modelos. En este capítulo, se presenta el análisis de los resultados obtenidos de la segmentación en vivo, además de gráficas y comparaciones entre los modelos seleccionados.

- Capítulo 4: Discusión, Conclusiones y trabajos futuros. En este capítulo, se discuten las conclusiones obtenidas a lo largo del desarrollo de la investigación, además de proponer posibles trabajos futuros a considerar.

Se presenta, a continuación, una introducción teórica para entender algunos conceptos necesarios de la IA y sus aplicaciones en medicina. Posteriormente, se mostrará cómo se ha implementado el sistema de visión artificial para la segmentación de heridas corto punzantes, teniendo para finalizar, la evaluación del desempeño de los modelos y una serie de resultados y conclusiones obtenidos de los algoritmos empleados. Presentando, por último, posibles mejoras o líneas de investigación futuras.

# Capítulo 1

# Generalidades

## 1.1. Planteamiento del Problema.

En la actualidad, los avances tecnológicos han llevado a la convergencia de las aplicaciones en informática y medicina [4]. Uno de los principales campos de aplicación es la cirugía robótica [5], en donde el cirujano dirige el instrumental a voluntad para realizar cada procedimiento. Las incisiones pequeñas que requiere el instrumental permiten una recuperación más rápida, disminuye riesgos asociados, y además genera menor dolor en comparación con la cirugía abierta.

Sin embargo, en la cirugía robótica aún existen muchos desafíos, principalmente por la pérdida de percepción táctil y de profundidad [6]. Así como la falta de ergonomía en la interfaz de manejo del robot [7]. Por este motivo, se prevé que el desarrollo de tareas automatizadas brindará beneficios tanto al paciente como al cirujano a cargo. Un caso particular de procedimientos extenuantes para los cirujanos es la ejecución de suturas por la cantidad de movimientos y acciones repetitivas que requiere pasar una aguja por una herida. Para el planteamiento de una sutura automatizada se requiere como primera instancia identificar y delimitar la herida a tratar, no obstante, es un campo poco explorado a diferencia del seguimiento del instrumental y otros elementos del quirófano. Por este motivo, en esta investigación se propone un sistema de reconocimiento de heridas que adopta un enfoque de aprendizaje profundo para la segmentación adecuada de la herida, cuyos resultados permitan obtener un modelo capaz de delimitar una herida de manera robusta mediante técnicas de visión artificial.

En informática, el procesamiento de imágenes hace uso de computadoras para manipular imágenes digitales a través de algoritmos de segmentación de imagen [8], los cuales realizan operaciones de morfología como: detección, clasificación, bloqueo, criptografía, identificación, localización, estenografía, fragmentación, seguimiento y reconocimiento. La visión por computadora busca duplicar el efecto de la visión humana mediante la percepción y análisis electrónico de las imágenes [9]. Para cumplir este propósito, el procesamiento digital de imágenes utiliza una técnica y algoritmos específicos. Como lo es la visión en robótica. La visión artificial es uno de los tipos de sensores inteligentes que aportan soluciones de vanguardia convirtiéndose en los ojos de un robot, siendo una alternativa que mejora las variaciones en la iluminación y las condiciones que generan disturbios en la imagen, también resalta mucho sobre la presencia de sangre ayudando a identificar de manera precisa la herida, sin permitir el reconocimiento robusto con técnicas de color. [10]. Diversos métodos para análisis de clasificación de imagen han sido implementados para la extracción de características de la misma. El aprendizaje profundo mediante redes neuronales convolucionales (CNN), ha sido aplicado en clasificación de heridas, destacando que dicho procedimiento se ha venido realizando a partir de imágenes bidimensionales a las cuales se les ha procesado por técnicas de selección a color. Ting Sun et al. [11] mencionan que la limitación del uso de imágenes 2D está supeditado al cambio de iluminación. Otro punto débil que tiene trabajar con este tipo de imágenes es que se desconoce la profundidad de campo, ya que la imagen es una matriz bidimensional. Además, el movimiento en la escena quirúrgica es otro inconveniente presente a tener en cuenta.

En la presente propuesta se pretende explorar las CNN para diseñar un sistema robusto que permita el reconocimiento y segmentación de heridas, que sea resistente ante variaciones de iluminación, rotación y desplazamiento. De acuerdo con lo explicado, surge la siguiente pregunta investigativa: En la línea de investigación en la ciencia quirúrgica, específicamente en el contexto de sutura autónoma, ¿Cuáles serán las características de un algoritmo de aprendizaje de máquina y procesamiento de imágenes que permita la segmentación y reconocimiento de heridas corto punzantes?

## 1.2. Estado del Arte

En esta sección, se presenta un marco teórico que sienta las bases para el proyecto de investigación actual. El propósito principal radica en la contextualización de la presente investigación en el contexto más amplio de lo que se ha investigado y publicado en el ámbito de la clasificación de imágenes mediante segmentación semántica. La segmentación semántica, una tarea esencial en la visión artificial, implica asignar una etiqueta semántica a cada píxel de una imagen, y encuentra diversas aplicaciones en el campo biomédico, tales como la detección y el seguimiento de heridas, el diagnóstico y tratamiento de enfermedades, la sutura automatizada, entre otros.

Para llevar a cabo la segmentación semántica, se han propuesto diversos modelos basados en redes neuronales convolucionales (CNN) que se especializan en la extracción y combinación de características visuales de diferentes niveles de abstracción. A continuación, se destacan algunos de los modelos más reconocidos y sus respectivas contribuciones en este ámbito:

- **U-Net** [12], [13], [14], [15], [16]: Es un modelo basado en una estructura de codificador-decodificador que utiliza conexiones de salto entre capas simétricas para preservar la información espacial. El codificador se encarga de extraer características de alto nivel, mientras que el decodificador se encarga de reconstruir la segmentación a partir de las características. U-Net ha demostrado su eficacia en la segmentación de imágenes médicas, especialmente en la segmentación de estructuras celulares y tejidos. Algunos de los resultados o aportes obtenidos con este modelo son:
  - Siddique y otros coautores logran la segmentación de vasos sanguíneos en imágenes de retina con una precisión del 95 % [13].
  - Ronneberger y su equipo, logran la segmentación de células nerviosas en imágenes de microscopía electrónica con una precisión del 93 % [15].
  - Otros investigadores consiguen segmentar tumores cerebrales en imágenes de resonancia magnética con una precisión del 85 % [16].
- **ResNet** [17], [18]: Es un modelo basado en bloques residuales que consisten en una conexión de identidad que salta una o más capas convolucionales. Esta conexión permite al modelo aprender funciones residuales en lugar de funciones directas, lo que facilita el entrenamiento de redes profundas y mejora el rendimiento. ResNet

se ha utilizado como columna vertebral de otros modelos de segmentación, como RefineNet y PSPNet. Algunos de los aportes obtenidos con este modelo son:

- La clasificación de imágenes en 1000 categorías con un error del 22 % [17].
- La clasificación de imágenes en 10 categorías con una precisión del 94 % [18].

- **RefineNet** [19], [20]: Es un modelo basado en una red de refinamiento multi-camino que combina características de diferentes niveles de resolución mediante un módulo de fusión adaptativo. Este módulo utiliza una convolución residual y una unidad de normalización de longitudes de onda para integrar las características de forma coherente. RefineNet ha obtenido buenos resultados en la segmentación de escenas urbanas y naturales. Algunos de los resultados obtenidos con este modelo son:

  - La segmentación de escenas urbanas en 19 categorías con una precisión del 81 % [19].
  - La segmentación de objetos en 20 categorías con una precisión del 83 % [20].

- **FCN** [21]: Es un modelo basado en una red totalmente convolucional que reemplaza las capas totalmente conectadas de una CNN por capas convolucionales, lo que permite procesar imágenes de cualquier tamaño. El modelo utiliza una operación de deconvolución para reconstruir la segmentación a partir de las características extraídas por la CNN. FCN fue uno de los primeros modelos en aplicar el aprendizaje profundo a la segmentación semántica y ha servido de base para otros modelos posteriores. Algunos de los resultados o aportes derivados de la implementación de este modelo son:

  - La segmentación de objetos en 20 categorías con una precisión del 62 % [21].
  - La segmentación de escenas interiores en 40 categorías con una precisión del 29 % [21].

- **Mask-RCNN** [22], [23]: Es un modelo basado en una red de dos etapas que combina la detección y la segmentación de objetos. La primera etapa consiste en una red de propuestas de regiones que genera candidatos a objetos, mientras que la segunda etapa consiste en una red de clasificación, regresión y segmentación que asigna una etiqueta, una caja delimitadora y una máscara a cada candidato. Mask-RCNN ha logrado resultados sobresalientes en la segmentación de instancias, que consiste en segmentar y diferenciar cada objeto de una misma clase. Algunos de los aportes obtenidos con la implementación de este modelo son:

- La segmentación de instancias en 80 categorías con una precisión del 38 % [22].
- Zhang y su equipo de trabajo lograron la segmentación de nódulos pulmonares en imágenes de tomografía computarizada con una precisión del 83 % [23].

- **SegNet** [24], [25]: Es un modelo basado en una estructura de codificador-decodificador que utiliza un mecanismo de indexación para transferir la información de las posiciones de los píxeles desde el codificador al decodificador. El codificador se basa en la arquitectura de VGG-16, mientras que el decodificador utiliza capas de deconvolución y de muestreo ascendente para reconstruir la segmentación. SegNet se ha aplicado principalmente a la segmentación de escenas urbanas con algunos de sus aportes a continuación:

  - La segmentación de escenas interiores en 37 categorías con una precisión del 28 % [24].
  - La segmentación de escenas urbanas en 11 categorías con una precisión del 46 % [25].

- **PSPNet** [16]: Es un modelo basado en una red piramidal de análisis de escenas que utiliza un módulo de agrupación espacial piramidal para capturar información contextual a diferentes escalas. Este módulo consiste en aplicar operaciones de agrupación a diferentes niveles de cuadrícula sobre las características extraídas por una CNN. PSPNet ha mostrado un alto rendimiento en la segmentación de escenas complejas y diversas, como en la investigación realizada por Divam Gupta que consigue la segmentación de escenas en 150 categorías con una precisión del 43 % [16].

- **DeepLab** [26], [27]: Es un modelo basado en una red de campos aleatorios condicionales totalmente conectados (CRF) que utiliza una convolución atrous para controlar la resolución de las características y un módulo de agrupación espacial atrous para capturar información contextual a múltiples tasas de dilatación. El modelo también utiliza una característica de nivel de imagen para incorporar información global y un CRF para refinar los bordes de los objetos. DeepLab actualmente es uno de los modelos más influyentes y exitosos en la segmentación semántica de imágenes. Algunos de los aportes obtenidos con este modelo son:

  - La segmentación de objetos en 20 categorías con una precisión del 89 % [28].
  - La segmentación de escenas en 150 categorías con una precisión del 43 % [27].

## 1.3. Aspectos teóricos

Este trabajo hace referencia a conceptos técnicos tanto del campo de la IA como del procesamiento de imágenes para visión artificial. En esta sección se diferencian dos secciones principalmente, en primer lugar se aborda una introducción a estos conceptos como segmentación, detección, aprendizaje profundo y convolución, entre otros. En segundo lugar, se describen algunos aspectos de implementación necesarios para el desarrollo del trabajo, tanto a nivel de software como de hardware, finalizando con un resumen general de los conceptos abordados.

### 1.3.1. Redes neuronales artificiales (ANN)

Las redes neuronales artificiales, como se menciona anteriormente, son algoritmos o modelos computacionales diseñados según la estructura neuronal del cerebro humano. La propiedad de interés es su capacidad de entender a partir de los datos. Actualmente, existen numerosos tipos y arquitecturas de redes neuronales artificiales, pero los algoritmos de perceptrón simple y multicapa son un buen punto de partida para entender su funcionamiento y su aplicación en IA.

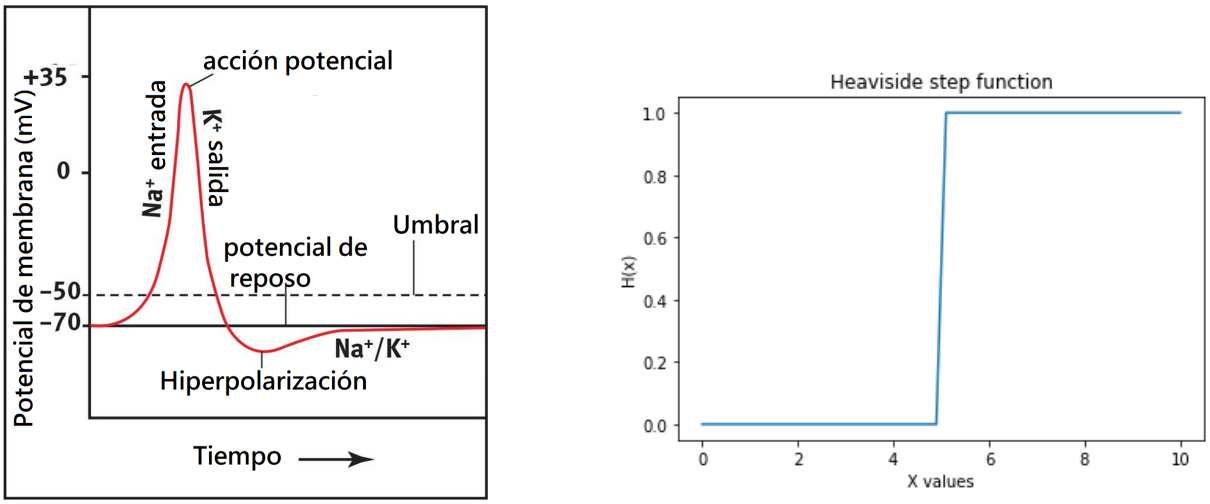
La mayoría de las neuronas son capaces de generar un tren de potenciales de acción, propagando pulsos de actividad electroquímica cuando el potencial medio a través de su membrana se mantiene muy por encima de su valor normal de reposo. La tasa media a la que se generan los potenciales de acción es una función uniforme del potencial medio de la membrana [29], con la forma general mostrada en la figura 1.18a ha sido modelo base para la traslación matemática. Cuando los pulsos de actividad neuronal son propagados, se habla de una red neuronal, que no es más que una interconexión de múltiples perceptrones o neuronas.

En esta sección del documento se abordan los fundamentos del modelo perceptrón (representación matemática de una neurona), su funcionamiento y algunos aspectos esenciales del mismo. Además, se abordan también las redes neuronales profundas (DNN) y las redes neuronales convolucionales (CNN), así como aspectos teóricos importantes de cada tipo de red.

#### 1.3.1.1. Perceptrón

La unidad básica de una red neuronal artificial es un procesador elemental llamado neurona (Perceptrón) que posee la capacidad limitada de calcular, en general, una suma ponderada de sus entradas que luego le aplica una función de activación para obtener una señal que será transmitida a la próxima neurona. Estas neuronas artificiales se agrupan en capas o niveles y poseen un alto grado de conectividad entre ellas, conectividad que es ponderada por valores numéricos denominados como pesos [30].

El modelo de una neurona artificial fue creado utilizando la lógica de operación de las células nerviosas en el cerebro humano, modelo de neurona artificial que se ha desarrollado a lo largo del tiempo y se ha utilizado con frecuencia en el aprendizaje automático. Este método se utilizó por primera vez en 1943, cuando se generó la primera neurona artificial: la Unidad Lógica de Umbral (TLU), o Unidad de Umbral Lineal, un modelo matemático de la estructura de las células en el cerebro humano (McCulloch y Pitts, 1943) cuyo modelo empleó un umbral, equivalente a usar la función de paso de Heaviside (ver figura 1.18b). El objetivo principal aquí es modelar las células nerviosas que permiten que el cerebro humano aprenda matemáticamente y permitir que un sistema informático demuestre un enfoque similar, siendo los modelos ANN la simplificación extrema de los sistemas neuronales humanos [31].




(a) Pulsos de activación neuronal

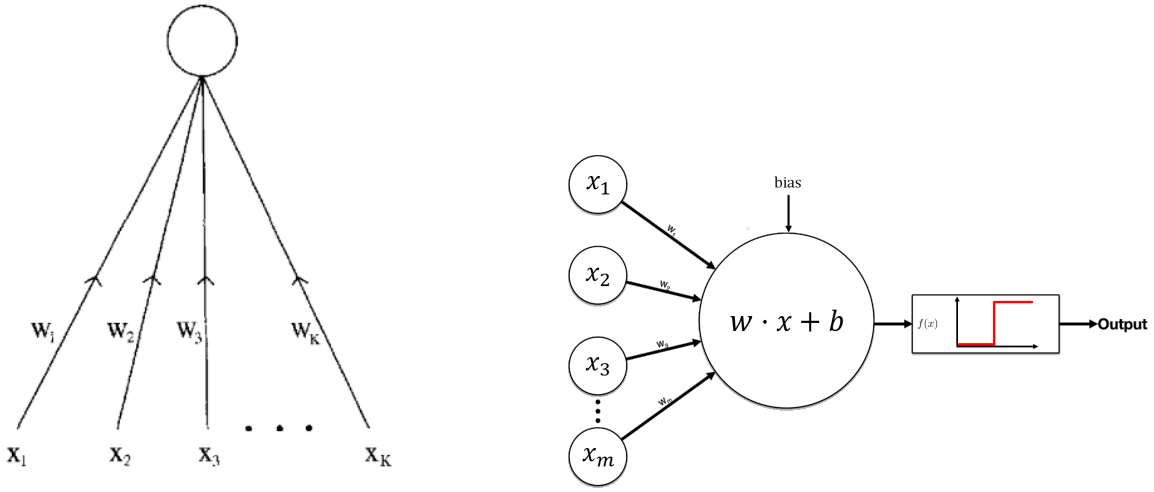
(b) Función de paso

Figura 1.1: Sistema Neuronal Humano

Como se observa en la figura 1.18a, se debe tener en cuenta que dentro de la neurona existe una diferencia de potencial como característica de la composición química que esta

contiene (potasio y sodio). En su parte interna [K+] es alto y [Na+] es bajo. mientras que fuera de la neurona, [Na+] es alto, y [K+] es bajo. El potencial de reposo negativo es generado tanto por proteínas cargadas negativamente dentro de la célula como por la permeabilidad relativamente mayor de la membrana a K+ en comparación con Na+ [32].

En resumen, la apertura de los canales de Na+ dependientes de voltaje durante el potencial de acción conduce a la despolarización de la célula, mientras que la apertura de los canales de K+ dependientes de voltaje conduce a la repolarización. La apertura de estos canales está regulada por la diferencia de voltaje a través de la membrana celular.




(a) Perceptrón como un grafo

(b) Modelo matemático perceptrón

Figura 1.2: Descripción gráfica del Modelo Perceptrón

En términos generales, podemos decir que una red neuronal artificial consiste en un grafo dirigido 1.2a con unidades (o neuronas) situadas en los vértices. Algunos de estos son unidades de entrada, que reciben señales del mundo exterior. El resto se denominan unidades de cálculo. Una o más de estas unidades de cálculo se especifican como unidades de salida. Estas son las unidades con grado de salida cero en el gráfico dirigido.

De forma aislada, una unidad de computación tiene cierto número, k, de entradas, y es capaz de asumir varios estados, cada uno descrito por un vector $w = (w_0, w_1, ...w_p) \in \mathbb{R}^p$ de $p$ números reales, conocidos como **pesos** o **parámetros**. Aquí, p, el número de parámetros de la unidad, dependerá de k. Si la unidad es una unidad de umbral lineal o una unidad sigmoidea, entonces p = k + 1 y, en estos casos, es útil pensar que los pesos $w_1, w_2, ..., w_k$ se asignan a cada una de las k entradas, como ilustra la figura 1.2b

Por lo tanto, un perceptrón como unidad básica de las redes neuronales, se concibe co-

mo un tipo de clasificador binario lineal que se puede usar para clasificar datos que son linealmente separables. Funciona tomando un conjunto de características de entrada, multiplicándolas por un conjunto de pesos y pasando el resultado a través de una función de activación que produce un valor de salida. El algoritmo de perceptrón ajusta los pesos de las características de entrada durante el entrenamiento para minimizar el error entre la etiqueta de clase predicha y la etiqueta de clase real. Este ajuste se realiza utilizando un enfoque de descenso de gradiente, que actualiza los pesos en la dirección del descenso más pronunciado de la función de error.

En un conjunto de datos de entrenamiento separables linealmente de $n$ dimensiones, se necesita un hiperplano para la separación. Esto representa un subespacio lineal de dimensión $n-1$. Como todo hiperplano de dimensión $(n-1)$ en $\mathbb{R}^n$ puede describirse mediante la siguiente ecuación:

$$\sum_{i=1}^{n} a_i x_i = \theta \tag{1.1}$$

- $Definicin$: Dos conjuntos $M1 \Subset \mathbb{R}^n$ y $M2 \Subset \mathbb{R}^n$ son linealmente separables si los números reales $a_1, ..., a_n, \theta$ existen con:

$$\sum_{i=1}^{n} a_i x_i > \theta \qquad \forall x \in M_1 \tag{1.2}$$

y

$$\sum_{i=1}^{n} a_i x_i < \theta \qquad \forall x \in M_2 \tag{1.3}$$

Donde el valor $\theta$ es denotado como el valor de umbral.

- $Teorema$: Sea $w = w_0, w_1, ...w_p) \in \mathbb{R}^p$ un vector de peso y $x \in \mathbb{R}^p$ un vector de entrada. Un perceptrón representa una función $P : \mathbb{R}^p \rightarrow 0, 1$ que corresponde a la siguiente regla:

$$P(x) = \begin{cases} 1 & \text{si } wx > \sum_{i=1}^{n} a_i x_i > 0 \\ 0 & \text{en caso contrario} \end{cases}$$

- $Teorema$: Una función $f : \mathbb{R}^p \rightarrow 0, 1$ puede representarse mediante un perceptrón si y

solo si los dos conjuntos de vectores de entrada positivos y negativos son linealmente separables [33].

El perceptrón es un algoritmo de clasificación muy simple. Es equivalente a una red neuronal de dos capas con activación por una función de umbral. Durante el desarrollo de las redes neuronales artificiales, ocurrió un hecho que retrasó las investigaciones en este campo. En 1969, Minsky y Papert demostraron que el perceptrón, considerado como un clasificador lineal, no podía resolver problemas no lineales. Este descubrimiento supuso un obstáculo significativo hasta que, aproximadamente diez años después, surgió una idea para solucionar esta limitación: utilizar capas de perceptrones interconectados con funciones de activación no lineales, lo que posteriormente se denominaría redes neuronales. Este hallazgo, junto con el gran avance en la capacidad de procesamiento informático, permitió que las redes neuronales salieran de su estado de hibernación [34].

#### 1.3.1.2. Red neuronal profunda (DNN)

El perceptrón multicapa, también conocido como red neuronal profunda (Deep Neural Network), es una generalización del perceptrón simple. Esta arquitectura consta de más de un nivel de neuronas y puede incluir una o varias capas ocultas entre la capa de entrada y la capa de salida. En estas capas ocultas, las funciones de activación entre las neuronas no son necesariamente lineales.

Las redes neuronales profundas pueden tener múltiples capas ocultas, lo que les permite aprender representaciones jerárquicas y realizar tareas de aprendizaje más complejas [35].

Las MLP se consideran las redes neuronales artificiales por defecto. Este tipo de redes también se representan mediante un dígrafo simple, el cual permite el paso de las entradas de capa en capa hasta llegar a la capa final.

### 1.3.2. Teoría detrás de las CNN y modelos de segmentación

#### 1.3.2.1. Red neuronal convolucional (CNN)

Las redes neuronales convolucionales (CNN) son un tipo de red neuronal comúnmente utilizada en tareas de reconocimiento de imágenes y videos. Están diseñados para apren-

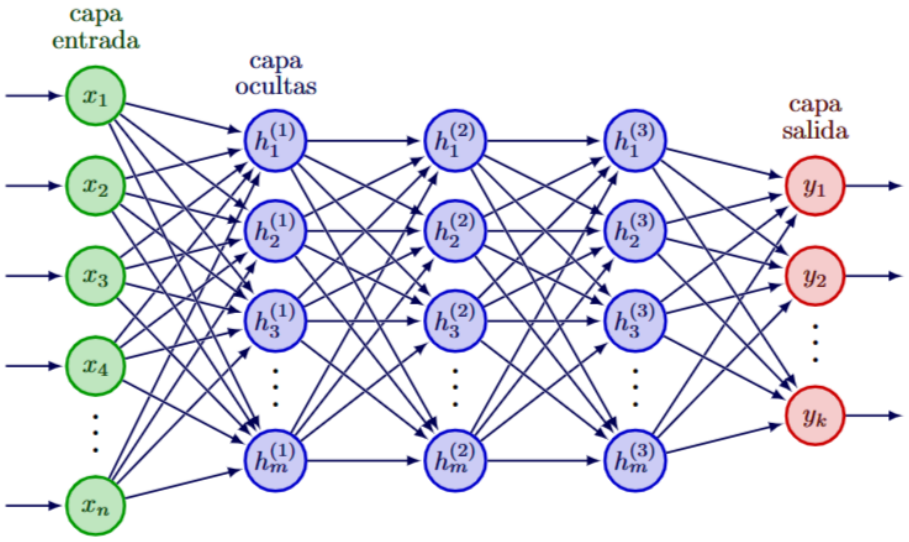



Figura 1.3: Red neuronal profunda con tres capas ocultas y una capa de salida

der de forma automática y adaptativa jerarquías espaciales de características a partir de imágenes o videos de entrada [36].

En una CNN, las unidades de cómputo se denominan neuronas o nodos y están dispuestas en capas. La primera capa suele ser una capa convolucional, que aplica un conjunto de filtros a la imagen de entrada para detectar características como bordes y esquinas. La salida de la capa convolucional pasa a través de una función de activación no lineal, como la Unidad lineal rectificada (ReLU), para introducir la no linealidad en la red.

Las capas posteriores pueden ser capas de agrupación, que reducen la muestra de la salida de la capa convolucional para reducir las dimensiones espaciales de los mapas de características. La capa final suele ser una capa completamente conectada, que calcula las puntuaciones de clase de salida en función de las características aprendidas por las capas anteriores [37].

Las CNN no son solo la adición o vinculación de unidades de computación, sino una arquitectura compleja que utiliza convolución, agrupación y capas totalmente conectadas para aprender y reconocer patrones complejos en imágenes y vídeos. Las arquitecturas de estas redes neuronales convolucionales pueden incluir un elevado número de capas ocultas entre la entrada y la salida, que hacen que a esta tecnología se la conozca con el nombre de aprendizaje profundo o deep learning (DL).

En el área de la detección de lesiones, las CNN poseen la enorme ventaja de aprender a identificar aquellas estructuras que se parecen más a una lesión, obviando estructuras

anatómicas normales que pueden tener intensidades similares[38] [39].

Por lo tanto, las redes neuronales convolucionales permiten que las redes profundas aprendan funciones en datos espaciales estructurados como imágenes, vídeo y texto. Matemáticamente, las redes convolucionales proporcionan herramientas para explotar la estructura local de los datos de manera efectiva, ya que las imágenes satisfacen ciertas propiedades estadísticas naturales.

En particular, a diferencia de una red neuronal normal, las redes convolucionales tienen neuronas dispuestas en 3 dimensiones: ancho, alto, profundidad (se adoptan las iniciales en inglés$(w, h, d)$). En las redes neuronales densas se tiene una matriz de pesos $(m, n)$para conectar una capa anterior de $n$ neuronas con una de $m$. Aquí se conectarán las $n$ neuronas con $d$ capas convolucionales usando un bloque de pesos $(w, h, d)$. La reducción de complejidad viene dado porque la dimensión de $(w, h)$ es muy inferior a $(m, n)$ [40].

Una operación de convolución básica que se aplica a una imagen bidimensional I como entrada, usando un kernel o filtro K bidimensional, da como resultado una nueva imagen S sería por ejemplo:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$$

#### 1.3.2.2. Elementos de implementación de una CNN

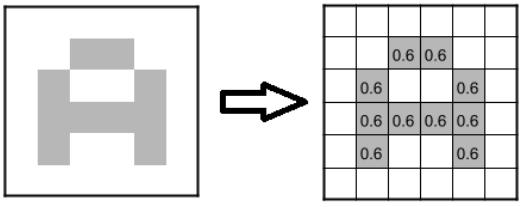



Figura 1.4: Representación de una imagen en valores normalizados
[41]

En las redes neuronales convolucionales (CNN), tiene parámetros e hiperparámetros. Es esencial entender la distinción entre ellos:

Parámetros: Los parámetros son los valores que la red aprende durante el proceso de entrenamiento. Estos son los pesos y sesgos de las capas de redes neuronales. En las CNN,

los parámetros suelen referirse a:

- Pesos de filtro: Estos son los núcleos aprendibles utilizados para las operaciones de convolución.
- Términos de sesgo: algunas capas pueden tener términos de sesgo asociados a cada filtro. Parámetros de capa totalmente conectados (si están presentes):
- Pesos: Estos son los pesos que conectan las neuronas entre las capas. Términos de sesgo: Al igual que las capas convolucionales, las capas totalmente conectadas pueden tener términos de sesgo.

**Hiperparámetros de una capa convolucional:**

Los hiperparámetros son ajustes que se establecen antes del entrenamiento y no se aprenden de los datos. Estas configuraciones influyen en cómo aprende la red e incluyen:

- Tasa de aprendizaje (hiperparámetro):

  La tasa de aprendizaje determina qué tan rápido aprende el modelo. Es un hiperparámetro crucial que controla el tamaño del paso durante las actualizaciones de peso en el proceso de entrenamiento.

- Tamaño del lote (hiperparámetro):

  El tamaño del lote determina el número de muestras utilizadas en cada paso hacia adelante y hacia atrás durante una iteración de entrenamiento. Afecta el uso de memoria y la velocidad de entrenamiento.

- Épocas (hiperparámetro):

  El número de épocas especifica cuántas veces se pasa todo el conjunto de datos de entrenamiento hacia adelante y hacia atrás a través de la red. Es un hiperparámetro porque debe decidir cuántas iteraciones son suficientes para el entrenamiento.

- Hiperparámetros de arquitectura:

  Estos incluyen parámetros como el número de capas, el tamaño de los núcleos en capas convolucionales, el número de filtros en cada capa y la profundidad de la arquitectura.

- Función de pérdida (hiperparámetro):

  La elección de la función de pérdida depende de la tarea específica, como la clasificación, la regresión o la segmentación. Las funciones de pérdida comunes incluyen error cuadrático, medio, entropía cruzada, categórica, etc.

- Algoritmo de optimización y sus hiperparámetros:

  Los hiperparámetros para los algoritmos de optimización incluyen el momento, las tasas de decaimiento y los valores épsilon. Los algoritmos de optimización usuales incluyen SGD, Adam, RMSprop, etc.

- Normalización por lotes (Batchnormalization):

  La normalización por lotes funciona normalizando las entradas a una capa en un mini lote de datos. Para cada característica (canal) en los datos de entrada, BatchNorm calcula la media y la desviación estándar sobre el mini lote. Luego resta la media y la divide por la desviación estándar.

- Validación Cruzada (Cross-Validation): La validación cruzada, como la validación cruzada k-fold, es una técnica frecuente para evaluar la capacidad de generalización de un modelo. Divide los datos en k particiones (folds), entrena el modelo en k-1 particiones y lo evalúa en la partición restante. Esto se repite k veces, y los resultados se promedian. Es especialmente útil cuando tienes un conjunto de datos pequeño.

- Enfoque detención temprana(earlystoping):

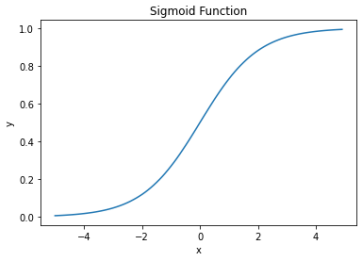
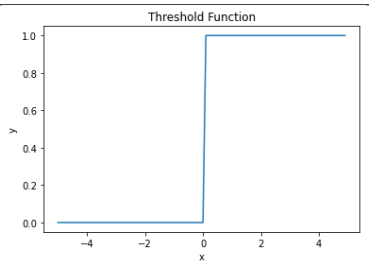
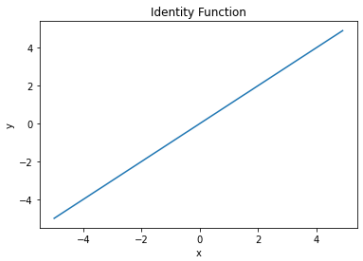
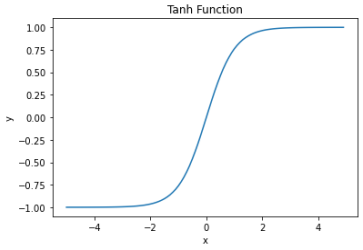
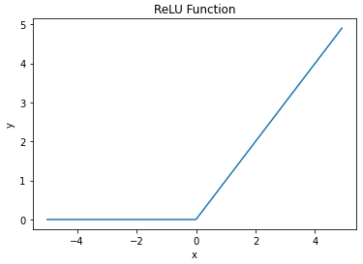
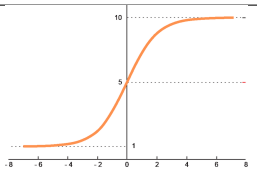
  Su objetivo es evitar el sobre ajuste (overfitting) y mejorar el rendimiento del modelo durante el proceso de entrenamiento. Primero, se divide el conjunto de datos en tres partes: entrenamiento, validación y prueba. La parte de entrenamiento se utiliza para ajustar los pesos y parámetros de la CNN, la parte de validación se utiliza para evaluar el rendimiento en datos no vistos durante el entrenamiento, y la parte de prueba se reserva para evaluar el rendimiento final después del entrenamiento; y segundo durante el entrenamiento, mide el rendimiento de la CNN en el conjunto de validación en intervalos regulares (por ejemplo, después de cada época de entrenamiento). El rendimiento puede medirse mediante métricas como la precisión.

- Callbacks: Las Callbacks se utilizan para realizar ciertas acciones en puntos específicos durante el entrenamiento, como después de cada época o después de que se procesa un lote de datos. Los más comunes son: Reduce LROnPlateau, TensorBoard

Integration, Data Augmentation, EarlyStopping, Mod Monitoring, Metrics. Son una poderosa herramienta para monitorear y controlar el proceso de entrenamiento de un modelo de aprendizaje automático

- Funciones de Activación: La función de activación específica utilizada puede afectar significativamente el rendimiento y el comportamiento de la red neuronal, pero no se aprende de los datos como pesos y sesgos; en su lugar, se especifica como parte de la arquitectura del modelo. La elección de la función de activación, como ReLU (Unidad lineal rectificada), sigmoide, tanh u otras, son hiperparámetros porque es una elección de configuración realizada por el diseñador del modelo antes del entrenamiento.

Tabla 1.1: Representación de Funciones de Activación

| Nombre | Representación | $f(x)$ |
|---|---|---|
| Función Sigmoide |   | $[\ \sigma(x) = \frac{1}{1+e^{-x}}]$ |
| Función Threshold (Heaviside) |   | $\text{thresh(x)} = \begin{cases} 1, & \text{if } x \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases}$ |
| Función Identidad |   | $f(x) = x$ |
| Función Tangente |   | $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ |
| Unidad lineal rectificada (ReLu) |   | $f(x) = \text{máx}(0, x)$ |
| Función Softmax |  | $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$ |

#### 1.3.2.3. Conjunto de datos

Al momento en que se decide emplear las redes neuronales, es la entrada de la primera capa de la red que proporciona una serie de datos que sirven de base para comenzar a entrenar un algoritmo, con el fin de que una máquina pueda aprender, los cuales son denominados conjunto de datos o data set. Para alcanzar el anterior objetivo, se divide dicho data set en tres conjuntos diferentes:

- Conjunto de entrenamiento (training): en la gran mayoría de casos se corresponde con el conjunto de mayor tamaño (cercano a 3 quintos) y se emplea para entrenar, es decir, ajustar los parámetros de la red en la etapa de aprendizaje para que las respuestas producidas en la capa de salida se ajusten lo más posible a los datos ya conocidos. De esta manera, se elaboran una serie de modelos para solventar el problema propuesto.
- Conjunto de validación (validation): encargado de seleccionar y ajustar los hiperparámetros de la red neuronal con el fin de validar todos los modelos anteriores y seleccionar el más óptimo atendiendo a la cercanía de la predicción. En relación con su tamaño, normalmente se corresponde en torno a una quinta parte del conjunto general. Sin embargo, no siempre es necesario dicho conjunto, puesto que si no hubiera hiperparámetros lo podríamos omitir.
- Conjunto de prueba (test): Su tamaño suele ser de dimensión similar al conjunto de validación, es decir, una quinta parte del conjunto de datos completo.

#### 1.3.2.4. Funciones de perdida

En el aprendizaje automático, el objetivo es entrenar un modelo que pueda predecir con precisión algunos resultados dados algunos datos de entrada. Durante el entrenamiento, el modelo ajusta sus parámetros (pesos y sesgos) para minimizar el error entre su salida prevista y la salida real. La función de pérdida es una función matemática que mide la diferencia entre la salida prevista y la salida verdadera. La función de pérdida de entropía cruzada es una función de pérdida comúnmente utilizada en problemas de clasificación

donde existen dos o más clases. La pérdida de entropía cruzada binaria entre las probabilidades predichas y las probabilidades verdaderas se define como: $(\hat{y})$ y $(y)$

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i})$$

Donde $y$ es un vector de etiquetas de clase verdaderas, $\hat{y}$ es un vector de probabilidades de clase predichas (salidas de Softmax)y $N$ es el número de clases.

El objetivo de entrenar una CNN es minimizar el valor de la función de pérdida ajustando los pesos y sesgos de la red utilizando un algoritmo de optimización como el descenso de gradiente o Adam.

Entropía cruzada binaria (BCE): La entropía cruzada binaria es una función de pérdida utilizada principalmente en problemas de clasificación binaria. Mide la disimilitud entre las etiquetas binarias verdaderas (0 o 1) y las probabilidades predichas de clase 1.

$$BCE(y, p)) = -[y \cdot log(p) + (1 - y) \cdot log(1 - p)] \tag{1.4}$$

**Función de pérdida en el aprendizaje profundo**

- Regresión

  MSE (error cuadrático medio)

  MAE (Error absoluto medio)

  Pérdida de Hubber

- Clasificación

  Entropía cruzada binaria

  Entropía cruzada categórica

En el contexto de las CNN, los términos función de pérdida y función de error se pueden usar indistintamente para referirse al mismo concepto: una función matemática que mide la diferencia entre la salida prevista del modelo y la salida real, y que se utiliza para optimizar el modelo durante el entrenamiento.
En el aprendizaje automático, hay varios tipos de gradientes utilizados en algoritmos de optimización para entrenar modelos. Estos son algunos ejemplos:

- Descenso de gradiente: El algoritmo de optimización más básico, que actualiza los parámetros del modelo en la dirección opuesta al gradiente de la función de pérdida.
- Descenso de gradiente estocástico (SGD): una variante del descenso de gradiente que actualiza los parámetros del modelo utilizando solo un subconjunto aleatorio de los datos de entrenamiento en cada iteración, que puede ser computacionalmente más eficiente.
- Mini-Batch Gradient Descent: Una variante del descenso de gradiente estocástico que actualiza los parámetros del modelo utilizando un pequeño lote aleatorio de datos de entrenamiento en cada iteración.
- Adam: Un popular algoritmo de optimización que adapta la tasa de aprendizaje en función del primer y segundo momento de los gradientes, lo que puede ayudar a acelerar la convergencia.
- RMSProp: Un algoritmo de optimización que utiliza una media móvil de los gradientes al cuadrado para ajustar la tasa de aprendizaje, lo que puede ayudar a lidiar con gradientes ruidosos.
- Adagrad: Un algoritmo de optimización que adapta la tasa de aprendizaje para cada parámetro en función de los gradientes históricos, lo que puede ayudar a lidiar con gradientes dispersos.

#### 1.3.2.5. Optimizador Adam

Para comprender mejor el concepto de algoritmos de optimización, es preciso inmiscuirnos en el algoritmo denominado descenso de gradiente, que implica calcular el gradiente de la función de pérdida con respecto a los parámetros (en este caso, los pesos del filtro) y actualizar los parámetros en la dirección opuesta al gradiente.

Suponiendo que se tiene una CNN con una función de pérdida $L$ que depende de los pesos de un filtro $w$. El objetivo es encontrar los pesos que minimizan la función de pérdida, es decir, los pesos que producen la mejor predicción para la tarea objetivo.

El gradiente de la función de pérdida con respecto a los pesos viene dado por las derivadas parciales de la función de pérdida con respecto a cada peso:

$$\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n} \tag{1.5}$$

Para actualizar los pesos utilizando el descenso de degradado, primero inicializamos los pesos a algunos valores aleatorios. Luego calculamos el gradiente de la función de pérdida con respecto a los pesos usando backpropagation:

$$\frac{dL}{dw_1}, \frac{dL}{dw_2}, \ldots, \frac{dL}{dw_n} \tag{1.6}$$

Luego actualizamos los pesos usando la siguiente fórmula:

$$w' = w - \alpha \cdot \frac{dL}{dw} \tag{1.7}$$

Donde $w'$ es el nuevo valor de los pesos, $\alpha$ es la tasa de aprendizaje o learning rate (un hiperparámetro que controla el tamaño de la actualización), y dL/dw es el gradiente de la función de pérdida con respecto a los pesos.

La regla de actualización funciona moviendo los pesos en la dirección de descenso más pronunciado de la función de pérdida. Al aplicar repetidamente esta regla de actualización para un cierto número de iteraciones o hasta la convergencia, podemos encontrar los pesos que minimizan la función de pérdida y producen la mejor predicción para la tarea objetivo.

El gradiente es un vector que apunta en la dirección del aumento máximo de la función, por lo que para obtener la dirección de descenso más pronunciado, simplemente negamos el vector de degradado.

Por ejemplo, si el gradiente de la función de pérdida es [2, 3, -1], entonces la dirección de descenso más pronunciado es [-2, -3, 1]. Esto nos dice que si actualizamos los pesos en la dirección de [-2, -3, 1], nos moveremos hacia el mínimo de la función de pérdida, ya que esta es la dirección en la que la función de pérdida disminuye más rápidamente.

El tamaño de la actualización está controlado por la tasa de aprendizaje, que determina qué tan lejos nos movemos en la dirección del descenso más pronunciado.

El optimizador Adam es un algoritmo popular de optimización utilizado en el aprendizaje profundo para actualizar los pesos y sesgos de las redes neuronales durante el entrena-

miento. Combina los beneficios de otros dos algoritmos de optimización: Adaptive Moment Estimation (Adam) y Root Mean Square Propagation (RMSProp).

Con Adam optimizer, los pesos y sesgos se pueden actualizan a partir del gradiente de las funciones de perdida (Loss Function) tanto con DICE como también a entropía cruzada; así, para hacer optimización de gradiente con DICE, las siguientes fórmulas y pasos son convenientes para actualización de pesos de la red:

- Calcula el gradiente de la función de pérdida con respecto a los pesos y sesgos:

$$g_t = \nabla_{\theta_t} J(\theta_t) \tag{1.8}$$

  Donde $g_t$ es el gradiente (degradado) en el paso de tiempo $t$, $\theta_t$ es el valor actual de los pesos y sesgos, y $J(\theta_t)$ es el valor de la función de pérdida en $\theta_t$.

  El símbolo $\nabla$ representa el operador de gradiente en matemáticas, un vector de derivadas parciales con respecto a cada dimensión de una función multivariable. La notación $\nabla_\theta$ representa el gradiente de una función con respecto a los parámetros del modelo $\theta$.

- Calcula el primer y segundo momento del degradado:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{1.9}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{1.10}$$

  Donde $m_t$ y $v_t$ son estimaciones del primer y segundo momento del gradiente en el paso de tiempo $t$, $\beta_1$ y $\beta_2$ son las tasas de decaimiento exponencial para el primer y segundo momento, $g_t$ representa gradiente de la función de pérdida con respecto a los pesos y $g_t^2$ representa el cuadrado elemento lógico del gradiente 1.5.

- Calcule las estimaciones de primer y segundo momento corregidas por sesgo:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{1.11}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{1.12}$$

Donde $\hat{m}_t$ y $\hat{v}_t$ son las estimaciones corregidas por sesgo del primer y segundo momento.

- Actualice los pesos y sesgos (Backpropagation) utilizando las estimaciones corregidas por sesgo:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{1.13}$$

Donde $\alpha$ es la tasa de aprendizaje, $\epsilon$ es una pequeña constante para evitar la división por cero, y $\sqrt{\hat{v}_t}$ y $\hat{m}_t$ son la raíz cuadrada y las estimaciones del primer momento, respectivamente. El optimizador de Adam actualiza los pesos y sesgos de la red neuronal calculando el gradiente de la función de pérdida, estimando el primer y segundo momento del gradiente, calculando las estimaciones con corrección de sesgo y luego actualizando los pesos y sesgos usando estas estimaciones y la tasa de aprendizaje. El algoritmo adapta la tasa de aprendizaje a cada parámetro individual, en función de la magnitud del primer y segundo momento del gradiente, lo que da como resultado una convergencia más rápida y un mejor rendimiento de generalización.

- IoU: la Intersección sobre unión (IoU) también se puede utilizar como una función de pérdida durante el entrenamiento para tareas de segmentación de imágenes.

$$IoU = |\frac{A \cap B}{A \cup B}| \tag{1.14}$$

Donde A y B son los conjuntos de píxeles en las máscaras de segmentación predichas y reales, respectivamente. Los gradientes de la función de pérdida de IoU con respecto a los pesos de la red se pueden calcular mediante retropropagación donde su coeficiente puede representarse similar al coeficiente de Sorensen-Dice:

$$\nabla_\omega L = \frac{\partial IoU}{\partial \omega} \tag{1.15}$$

Donde $L$ es la función de pérdida de IoU y $\omega$ representa los pesos de la red. Matemáticamente, la regla de actualización para el optimizador de Adam en este caso se puede definir con las mismas estimaciones de la regla de optimización para con el coeficiente DICE explicada anteriormente; de tal manera el optimizador Adam es el responsable de ajustar los pesos de la red durante el entrenamiento para minimizar la pérdida.

#### 1.3.2.6. Segmentación de imágenes

Desde el punto de vista de la cognición humana, quizás una de las habilidades más significativas sea la capacidad de segmentar imágenes, lo cual consiste en identificar y agrupar patrones visuales en elementos o conceptos abstractos significativos. Este proceso de agrupación perceptiva es la base de la capacidad para comprender y catalogar la información presente en las imágenes. Con el desarrollo actual de la informática, se abre la posibilidad de extender los conceptos que rigen la percepción visual humana a sistemas computacionales, lo que en el procesamiento digital de imágenes se conoce generalmente como anotación o segmentación de imágenes [42].

#### 1.3.2.7. Segmentación instantánea

Existen principalmente dos enfoques para abordar la segmentación de instancias: métodos basados en segmentación y métodos basados en detección.

Los métodos basados en segmentación se centran en predecir categorías de nivel de píxel y a continuación, agregar píxeles de la misma clase para obtener los resultados finales de segmentación de instancias. Por otro lado, los métodos basados en la detección realizan directamente la detección de objetos y, a continuación, refinan los cuadros delimitadores detectados, prediciendo las máscaras de nivel de píxel para cada instancia de objeto. Estos métodos suelen utilizar marcos de detección de objetos, donde cada objeto es singularizado en una imagen por medio de una ventana delimitadora (bounding box) y una máscara binaria, que determina qué píxeles de dicha ventana pertenecen al objeto localizado. Se trata, por tanto, de un sistema híbrido entre la detección de objetos y la segmentación semántica, y en cierto modo puede ser considerado como una evolución del primero.

Uno de los algoritmos más utilizados en Segmentación de Instancia, Mask R-CNN, que evolucionó a partir de un procedimiento concebido para localizar objetos: R-CNN. A diferencia de lo que sucede con YOLO, que funciona de un solo golpe o disparo.

#### 1.3.2.8. Segmentación semántica

Continuando, con la segmentación semántica como un problema de la visión por computadora que se desarrolla en la comprensión del contexto de una imagen, la segmentación

semántica es una tarea de la visión artificial que consiste en asignar una etiqueta a cada píxel de una imagen, de modo que los píxeles que pertenecen al mismo objeto o región reciben la misma etiqueta. A diferencia de la detección de objetos, que solo identifica la presencia de objetos en una imagen, la segmentación semántica brinda una comprensión más detallada de la imagen al etiquetar cada píxel según su categoría semántica, como un automóvil, una carretera o un peatón.

Existen diferentes estrategias para realizar tareas de segmentación, como el uso de técnicas morfológicas junto a métodos como el de Otsu [43], también técnicas de aprendizaje no supervisado y técnicas de aprendizaje profundo o Deep Learning para clasificación automática [44], el cual es un subcampo del aprendizaje automático que emplea Redes Neuronales para realizar predicciones a base de datos de entrada.

Hoy en día las redes neuronales convolucionales (CNN) han demostrado ser muy efectivas en la segmentación semántica. Estas redes aprovechan las operaciones matemáticas, las técnicas de umbral y la morfología matemática de diferentes maneras:

Operaciones matemáticas: Las CNN utilizan operaciones matemáticas, como la convolución y las funciones de activación, para procesar las imágenes de entrada. La convolución es una operación clave en las CNN, donde se aplican filtros a las imágenes para extraer características locales y aprender representaciones más abstractas. Las funciones de activación, como la función ReLU, introducen no linealidad en la red, lo que le permite capturar relaciones más complejas entre los píxeles de la imagen.

Técnicas de umbral: Las técnicas de umbral, como se mencionó anteriormente, se utilizan para convertir imágenes en imágenes binarias, donde los píxeles se clasifican como objeto o fondo. Esto se aplica como una etapa de preprocesamiento en algunos enfoques de segmentación semántica, permitiendo separar los objetos de interés del fondo.

Morfología matemática: La morfología matemática, con operaciones como erosión, dilatación, apertura y cierre, se utiliza para manipular la forma y la estructura de los objetos en una imagen. Estas operaciones son particularmente útiles para refinar y mejorar las máscaras de segmentación obtenidas de otras técnicas. Por ejemplo, la erosión puede eliminar ruido y detalles innecesarios, mientras que la dilatación puede cerrar brechas y mejorar la conectividad de los objetos.

En cuanto a los métodos más avanzados, además de las CNN, existen enfoques como las redes neuronales completamente convolucionales (FCN), que se diseñan específicamente

para la segmentación semántica al conservar la información espacial a través de convoluciones y saltos de conexión. También se utilizan arquitecturas como U-Net, que aprovechan la estructura de codificador-decodificador para capturar características a diferentes escalas y lograr segmentaciones precisas.

En resumen, la segmentación semántica utiliza redes neuronales convolucionales y métodos avanzados, como FCN y U-Net, para asignar etiquetas a cada píxel de una imagen. Estas redes aprovechan operaciones matemáticas, técnicas de umbral y morfología matemática para procesar imágenes, extraer características relevantes y lograr segmentaciones precisas de objetos y regiones en las imágenes. Los modelos de DL han demostrado ser herramientas poderosas para el procesamiento de imágenes médicas; y en cuanto a su rendimiento de estos modelos, como en otros modelos de aprendizaje automático, depende del contexto de los datos.

Hay varios niveles de granularidad en los que las computadoras pueden obtener una comprensión de las imágenes. Para cada uno de estos niveles hay un problema definido en el dominio de Computer Vision describiendo estos problemas como sigue a continuación:

#### 1.3.2.9. Clasificación de imágenes

El bloque de construcción más fundamental en la visión artificial es el problema de clasificación de imágenes, donde dada una imagen, se espera que la computadora genere una etiqueta discreta, que es el objeto principal de la imagen. En la clasificación de imágenes se asume que solo hay un objeto (y no varios) en la imagen.

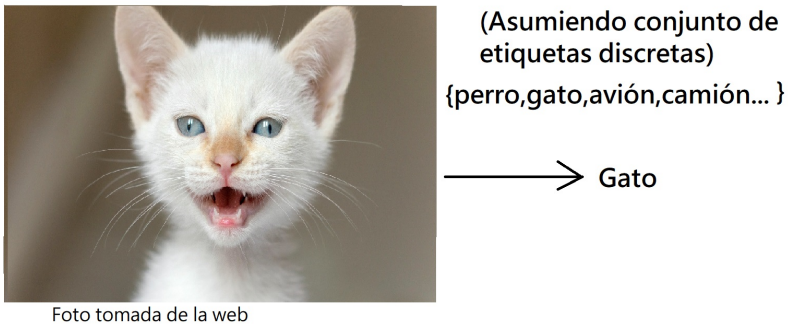



Figura 1.5: Clasificación en imágenes. Adaptado de [45]

#### 1.3.2.10. Clasificación con localización

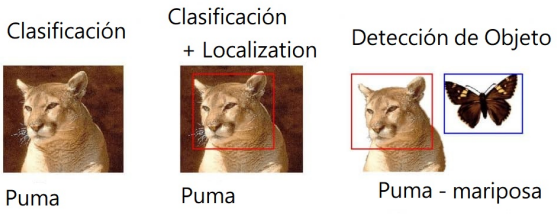



Figura 1.6: Clasificación con localización. Adaptado de [46]

En la localización, junto con la etiqueta discreta, también esperamos que el cómputo localice dónde está presente exactamente el objeto en la imagen. Esta localización se implementa normalmente mediante un cuadro delimitador que se puede identificar mediante algunos parámetros numéricos con respecto al límite de la imagen. Incluso en este caso, la suposición es tener solo un objeto por imagen.

#### 1.3.2.11. Detección de objetos

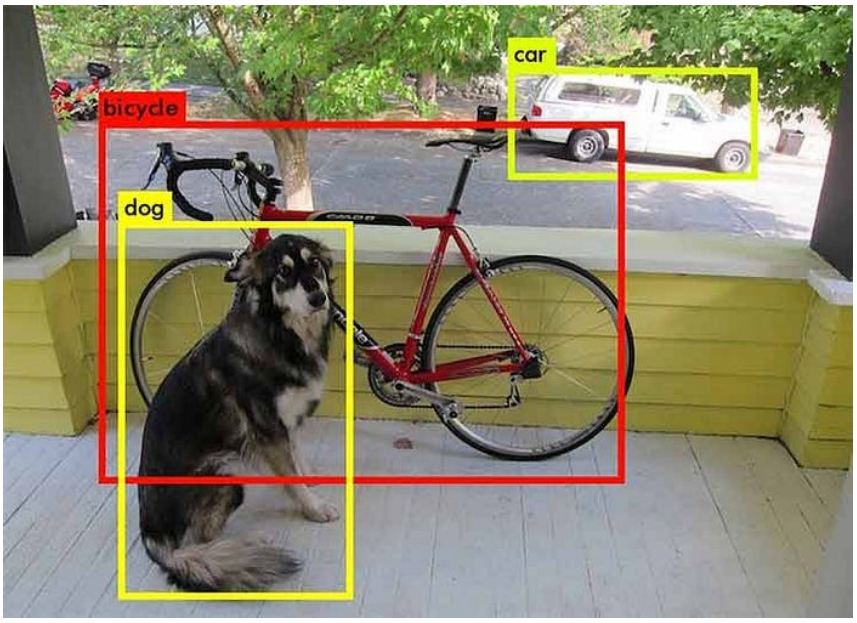



Figura 1.7: Detección de objeto. Tomado de [45]

La detección de objetos extiende la localización al siguiente nivel, donde ahora la imagen no está limitada a tener un solo objeto, sino que puede contener varios objetos. La tarea

consiste en clasificar y localizar todos los objetos de la imagen. Una vez más, la localización se realiza utilizando el concepto de cuadro delimitador.

#### 1.3.2.12. Segmentación semántica

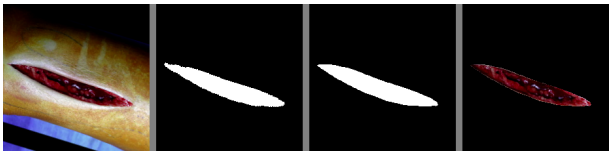


Figura 1.8: Segmentación semántica. Elaboración propia

Un sencillo ejemplo de segmentación semántica es la separación de las imágenes en dos clases distintas. Por ejemplo, en la figura 1.8, la imagen de una herida se empareja con otra versión en la que se muestran los píxeles de la imagen segmentados en dos clases distintas: herida y fondo.

El objetivo de la segmentación semántica de imágenes es etiquetar cada **píxel** de una imagen con una **clase** correspondiente de lo que se está representando. La segmentación semántica es un algoritmo de deep learning que asocia una etiqueta o categoría a cada píxel presente en una imagen. Se utiliza para reconocer un conjunto de píxeles que conforman distintas categorías [47].

El resultado esperado en la segmentación semántica no son solo etiquetas y parámetros de cuadro delimitador. La salida en sí es una imagen de alta resolución (típicamente del mismo tamaño que la imagen de entrada) en la que cada píxel se clasifica en una clase particular. Por lo tanto, es una clasificación de imagen a nivel de píxel[45].

#### 1.3.2.13. Segmentación de instancias

La segmentación de instancias está un paso por delante de la segmentación semántica en la que, junto con la clasificación a nivel de píxel, esperamos que el equipo clasifique cada instancia de una clase por separado.

La segmentación de instancia combina el método de segmentación semántica que interpreta los objetos de una imagen y la detección de objetos, que los localiza dentro de la imagen, por lo tanto, es aquel sistema de Visión Artificial en el que cada objeto es singularizado en una imagen por medio de una ventana delimitadora (bounding box) y una máscara

binaria, que determina qué píxeles de dicha ventana pertenecen al objeto localizado. Se trata, por tanto, de un sistema híbrido entre la detección de objetos y la segmentación semántica, y en cierto modo puede ser considerado como una evolución del primero [48].

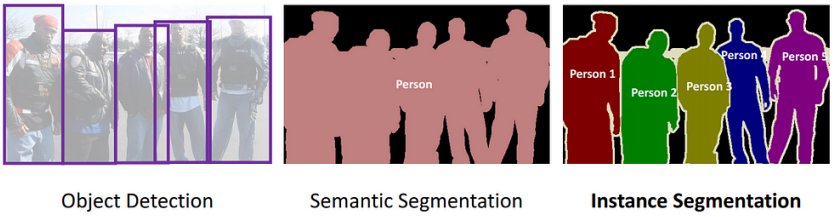



Figura 1.9: Detección de objetos vs. segmentación semántica vs. segmentación de instancias. Tomado de [49].

#### 1.3.2.14. La convolución, la agrupación máxima y la convolución transpuesta

Antes de avanzar en un modelo específico de CNN, es muy importante comprender las diferentes operaciones que se utilizan típicamente en una red convolucional.

#### 1.3.2.15. La necesidad de un muestreo ascendente

En el uso de redes neuronales para generar imágenes, generalmente implica un muestreo ascendente de baja resolución a alta resolución. Existen varios métodos para llevar a cabo la operación de muestreo ascendente:

- Interpolación del vecino más cercano
- Interpolación bilineal
- Interpolación bi cúbica

Todos estos métodos implican algún método de interpolación que se debe elegir al decidir una arquitectura de red. Es como una ingeniería manual de características y no hay nada que la red pueda aprender mientras que en las redes neuronales convolucionales (CNN) en lugar de interpolación bi cúbica u otro método, utilizan capas de convolución transpuesta para lograr el upsampling. Estas capas son entrenables y se utilizan para restaurar detalles espaciales en el mapa de características.

#### 1.3.2.16. Convolución transpuesta

Para explicación de este importante tema dentro de las redes neuronales, se tomó como base la publicación de [50] mencionado anteriormente, donde las figuras 1.11, 1.12, 1.13, 1.14, 1.16,1.17, 1.18; forman parte de la publicación original, además de la mención de otras fuentes que ayudan a entender estos conceptos.

Para que la red aprenda a muestrear de manera óptima, es factible usar la convolución transpuesta. No utiliza un método de interpolación predefinido. Tiene parámetros aprendibles. Es útil comprender el concepto de circunvolución transpuesta tal como se utiliza en documentos y proyectos importantes como:

- El generador en DCGAN toma valores muestreados aleatoriamente para producir una imagen de tamaño completo.
- La segmentación semántica utiliza capas convolucionales para extraer entidades en el codificador y luego restaura el tamaño de la imagen original en el decodificador para que pueda clasificar cada píxel de la imagen original.

Para su información: la circonvolución transpuesta también se conoce como:

- Convolución fraccionada
- Deconvolución
- Convolución transpuesta

#### 1.3.2.17. Operación de convolución

Usando como ejemplo simple para explicar cómo funciona la operación de convolución. Se retoma el ejercicio dado por el autor en el que se tiene una matriz 4x4 y se aplica una operación de convolución sobre ella con un núcleo de 3x3, sin relleno, y con un paso de 1. Como se muestra más abajo, la salida es una matriz 2x2.

Operación de convolución

$$out_{imag} = \left[\frac{i + 2p - k}{s}\right] + 1$$

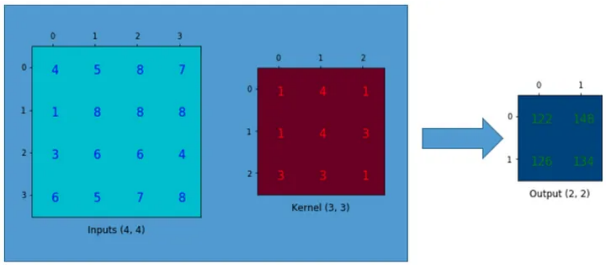



Figura 1.10: Matriz entrada, kernel de convolución e imagen salida. Tomado de [50]

Donde $Out_{imag}$ se refiere a la dimensión de la imagen de salida cuando se solapa un kernel de dimensión $k$ sobre una imagen de dimensión $i$, padding $p$. y zancadas $s$ [49].
La operación de convolución calcula la suma de la multiplicación de elementos entre la matriz de entrada y la matriz del núcleo. Como no hay relleno y el paso de 1,es posible hacer esto solo 4 veces. Por lo tanto, la matriz de salida es dimensiones 2x2.

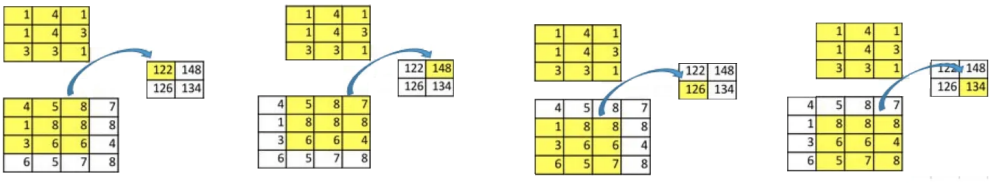


Figura 1.11: Operación de convolución. Tomado de [50]

Un punto importante de dicha operación de convolución es que existe la conectividad posicional entre los valores de entrada y los valores de salida. Por ejemplo, los valores superiores izquierdos de la matriz de entrada afectan al valor superior izquierdo de la matriz de salida. Más concretamente, el núcleo 3x3 se utiliza para conectar los 9 valores en la matriz de entrada a 1 valor en la matriz de salida. **Una operación de convolución forma una relación de varios a uno.**

#### 1.3.2.18. Retrocediendo la convolución

Ahora, suponiendo que se quiere ir en la otra dirección; es decir, asociar 1 valor en una matriz a 9 valores en otra matriz. **Es una relación de uno a muchos.** Esto es como retroceder en la operación de convolución, y es la idea central de la **convolución transpuesta**.

Por ejemplo, aumentando el muestreo de una matriz 2x2 a una matriz de 4x4. La operación mantiene la relación de 1 a 9.

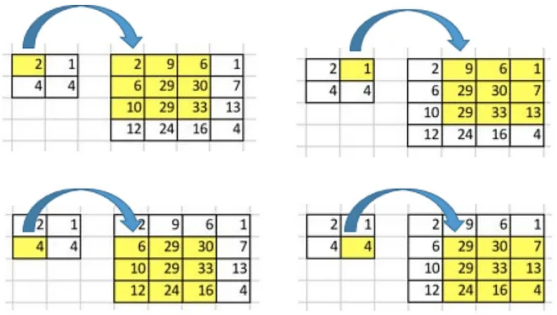



Figura 1.12: Retrocediendo de la convolución

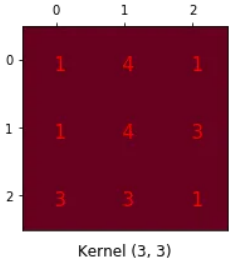



Figura 1.13: Kernel Matriz de convolución

Para hablar de cómo se realiza tal operación, es necesario definir la matriz de convolución y la **matriz de convolución transpuesta**.

#### 1.3.2.19. Matriz deconvolución

Esencialmente, en lugar de expresar el kernel anterior como una matriz 3x3, al realizar la transformación convolucional, se expresa como una matriz 4x16. Y en lugar de expresar la entrada anterior como una matriz 4x4, se expresa como un vector 16x1: Reorganizando el kernel 3x3 en una matriz 4x16, como se muestra a continuación:

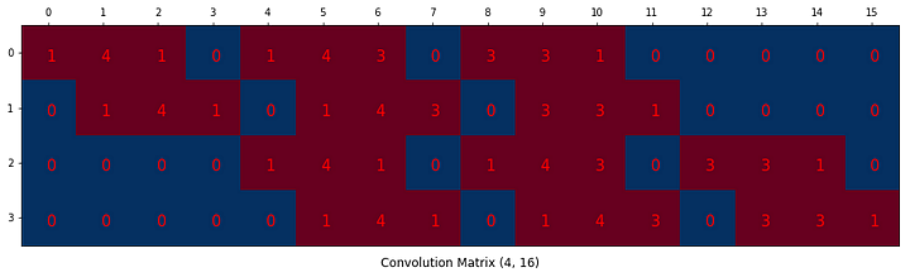



Figura 1.14: Un Kernel de 3x3 en una Matriz de convolución de 4x16 relleno con ceros (Zero-Padding) Convolución Matrix (Imagen por [50])

La razón por la matriz tiene dimensiones 4x16 se debe a que si la entrada y la salida fueran a desenrollarse en vectores de izquierda a derecha, de arriba a abajo, la convolución podría representarse como una matriz dispersa C donde los elementos distintos de cero son los elementos $W_{i,j}$ del kernel (siendo $i$ y $j$ la fila y la columna del kernel respectivamente):

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

Figura 1.15: Matriz dispersa C. La convolución como operación matricial

Usando esta representación, el pase hacia atrás se obtiene fácilmente transponiendo C; en otras palabras, el error se propaga hacia atrás multiplicando la pérdida por $C^T$. Esta operación toma un vector de 4 dimensiones como entrada y produce un vector de 16 dimensiones como salida, y su patrón de conectividad es compatible con $C$ por construcción. En particular, el Kernel w define las matrices C y $C^T$ utilizadas para el paso hacia adelante y atrás [51].

4 filas: en total, es posible realizar cuatro convoluciones dividiendo una matriz de entrada 4x4 en cuatro matrices de 3x3.
16 columnas: la matriz de entrada se transformará en un vector de 16x1. Para realizar la multiplicación de matrices, tiene que ser de 16 columnas[52]. En esta matriz de convolución cada fila define una operación de convolución. Cada fila es solo una matriz de kernel reorganizada con cero relleno en diferentes lugares.

Para usarlo, la matriz de entrada (4x4) se aplana en un vector columna (16x1) multiplicando matricialmente la matriz de convolución 4x16 con la matriz de entrada 16x1 vectorizada (vector columna de 16 dimensiones) como se muestra en la figura 1.17a.

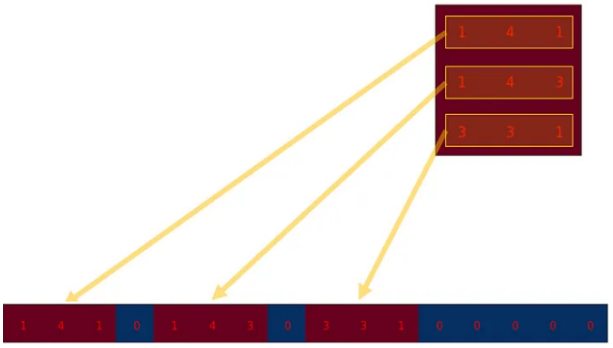
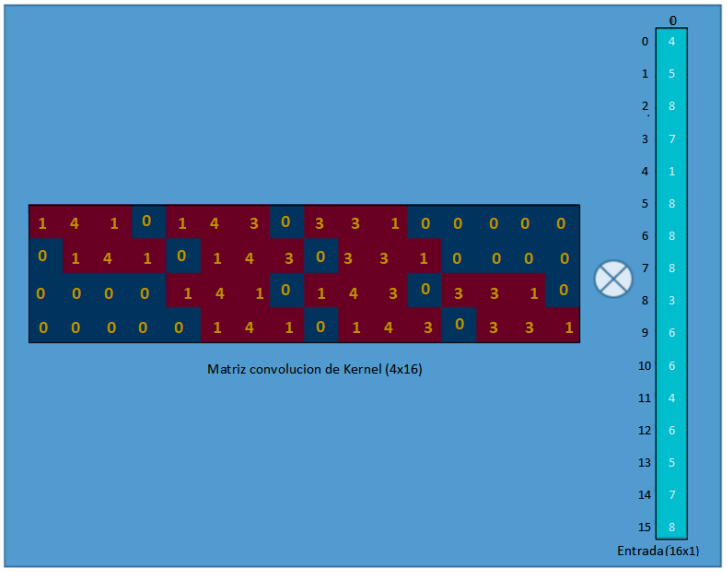
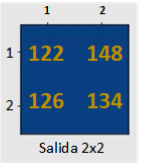


Figura 1.16: 3x3 Kernel as 4x16 Zero-Padded Convolution Matrix




(a) Matriz kernel transpuesto




(b) Salida remodelada en dimensiones 4x4

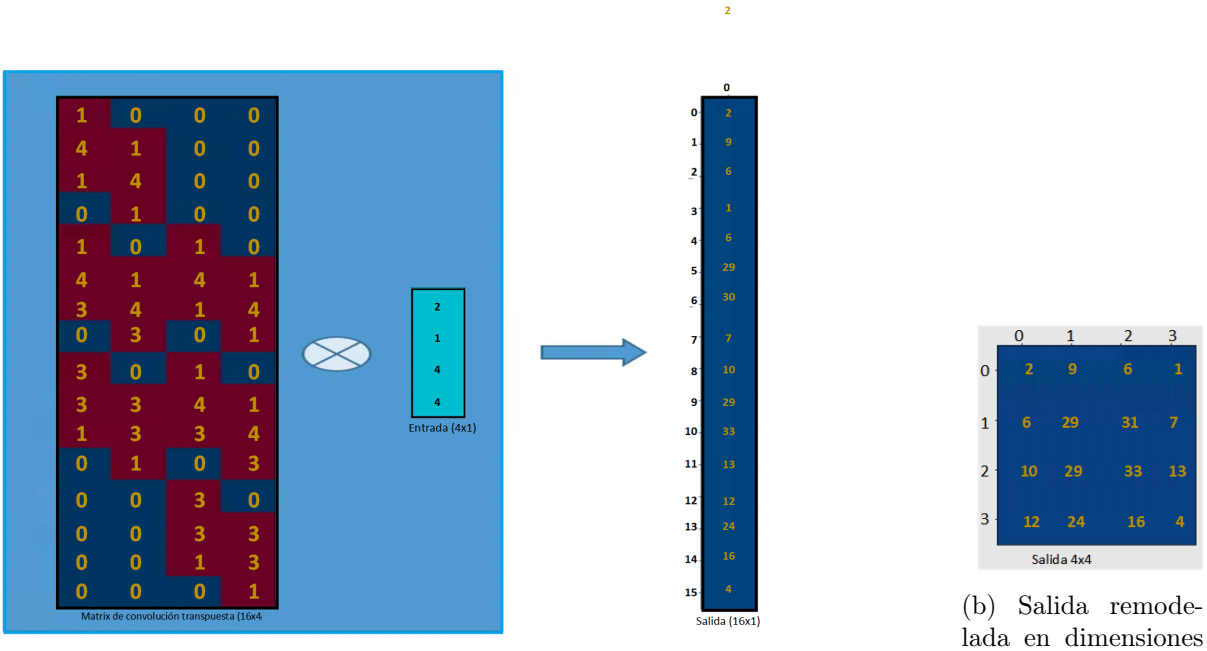
Figura 1.17: Matriz de convolución transpuesta

La matriz 4x1 de salida se puede remodelar en una matriz 2x2 que da el mismo resultado que antes. En resumen, una matriz deconvolución no es más que un reordenamiento de los pesos del núcleo, y una operación deconvolución se puede expresar utilizando la matriz de convolución. El punto es que con la matriz de convolución, puedes ir de 16 (4x4) a 4 (2x2) porque la matriz deconvolución es 4x16. Entonces, una matriz de 16x4, puede pasar de 4 (2x2) a 16 (4x4).

### 1.3.2.20. Matriz de convolución transpuesta

Para pasar de 4 (2x2) a 16 (4x4) dimensiones; entonces se usa una matriz de 16x4. Pero manteniendo la relación del 1 a 9.

Suponiendo que se transpone la matriz de convolución C (4x16) a C.T. (16x4). Podemos multiplicar matricialmente C.T. (16x4) con un vector columna (4x1) para generar una matriz de salida (16x1). La matriz transpuesta conecta 1 valor a 9 valores en la salida.




(a) Matriz kernel transpuesto

(b) Salida remodelada en dimensiones 4x4

Figura 1.18: Matriz de convolución transpuesta

Donde la salida se puede remodelar en dimensiones 4x4.

Resultando en el aumento del muestreo de una matriz más pequeña (2x2) en una más grande (4x4). La circunvolución transpuesta mantiene la relación de 1 a 9 debido a la forma en que establece los pesos. Los valores de peso reales en la matriz no tienen que provenir de la matriz de convolución original. Lo importante es que el diseño del peso se transpone del de la matriz de convolución.

La operación de convolución transpuesta forma la misma conectividad que la convolución normal pero en dirección inversa.

Podemos usarlo para llevar a cabo un muestreo ascendente. Además, los pesos en la circunvolución transpuesta son aprendibles. Así que no se necesita un método de interpolación predefinido.

Aunque se llama la convolución transpuesta, no significa que toma alguna matriz de convolución existente y use la versión *transpuesta*. El punto principal es que la asociación entre la entrada y la salida se maneja de manera inversa en comparación con una matriz de convolución estándar (asociación de uno a muchos en lugar de muchos a uno).

Como tal, la convolución transpuesta no es una convolución, pero emula la convolución transpuesta usando una convolución. Aumenta el muestreo de la entrada agregando ceros entre los valores de la matriz de entrada, de manera que la convolución directa produzca el mismo efecto que la convolución transpuesta.

### 1.3.3. Segmentación de imágenes médicas

Dentro del campo de la medicina, se define a la sutura, como la técnica que se usa para aproximar los bordes de una herida [53]. La clasificación algorítmica de imágenes de heridas es una tarea difícil debido a la variabilidad en la apariencia de los sitios de heridas.

Sin embargo, las CNN, un subgrupo de redes neuronales artificiales (ANN) se han mostrado muy prometedoras en el análisis de imágenes visuales donde actualmente se pueden aprovechar para clasificar las heridas quirúrgicas. Gracias a los avances en software y hardware, en forma de potentes algoritmos y unidades informáticas, han permitido que los algoritmos de aprendizaje profundo resuelvan una amplia variedad de tareas que antes se consideraban difíciles de abordar para las computadoras[54].

El tratamiento de imágenes con ANN, incluye diferentes etapas como el preprocesamiento, que ocasiona una mejora de las imágenes al normalizarlas, o la reducción de ruido con las mismas dimensiones que la imagen original, además de la reducción de datos o extracción de características de las imágenes, lo que implica extraer una cantidad de características menor que la cantidad de píxeles de la imagen en la capa de entrada de la red [55].

Actualmente, existen investigaciones sobre tipos específicos de heridas (úlcera de pie diabético, úlcera por presión, melanoma u otros tipos de cáncer en la piel, etc.) que se centran principalmente en tratamientos clínicos. Aunque las investigaciones analizan el trabajo previo sobre estadísticas de heridas, características, medición y planificación del tratamiento, no se incluyen enfoques o sistemas automatizados [56].

Así, la elección de la arquitectura a menudo depende de factores como la complejidad de la tarea, los recursos computacionales disponibles y el equilibrio deseado entre precisión y velocidad. En el caso de estudio presente se opta por usar segmentación de imagen binaria.

#### 1.3.3.1. Descripción general de las métricas de evaluación apropiadas que evalúa adecuadamente el rendimiento en segmentación de imágenes medicas.

Todas las métricas presentadas se basan en el cálculo de una **matriz de confusión** para una máscara de segmentación binaria, que contiene el número de predicciones de verdadero positivo (TP), falso positivo (FP), verdadero negativo (TN) y falso negativo (FN). Los rangos de valores de todas las métricas presentadas van desde cero (peor) hasta uno (mejor).

Se puede describir a la matriz de confusión como una herramienta utilizada para evaluar el rendimiento de un modelo de clasificación, incluidas las redes neuronales convolucionales (CNN) como U-Net, en el contexto de segmentación de imágenes. Siendo así, el modelo de segmentación de imágenes desarrolla una matriz de confusión de la siguiente manera:

Entrenamiento de modelos: Durante la fase de entrenamiento, el modelo se entrena en un conjunto de datos etiquetado que incluye imágenes de entrada y las correspondientes máscaras de segmentación de verdad en el terreno (Ground Truth en inglés) o imagen. El objetivo es enseñar al modelo a segmentar con precisión objetos o regiones de interés en las imágenes.

Predicción: Después del entrenamiento, el modelo se utiliza para hacer predicciones en un conjunto de datos separado, que puede incluir imágenes que nunca ha visto antes. Para cada imagen de entrada, el modelo genera una máscara de segmentación prevista.

Comparación con Ground Truth: Las máscaras de segmentación predichas se comparan con las máscaras de segmentación de Ground Truth para cada imagen del conjunto de datos (Data set). Esta comparación implica una evaluación de píxeles, donde cada píxel de la máscara predicha se compara con el píxel correspondiente en la máscara Ground Truth.

Matriz de confusión: La matriz de confusión se construye con base en las comparaciones de píxeles. La matriz de confusión es una matriz 2x2 (en segmentación binaria) o una matriz NxN (en segmentación multiclase), donde N es el número de clases.

Cálculo de métricas: Varias métricas de rendimiento se calculan en función de los valores de la matriz de confusión. Las métricas comunes incluyen:

- Exactitud: La proporción de píxeles correctamente clasificados (TP + TN) con respecto al número total de píxeles.
- Precisión: La proporción de verdaderos positivos (TP) a todas las predicciones positivas (TP + FP).
- Recuerdo (sensibilidad o tasa positiva verdadera): La proporción de verdaderos positivos (TP) a todos los positivos reales (TP + FN).
- Puntuación F1: El medio armónico de precisión y recuerdo, que equilibra la precisión y el recuerdo.
- IoU (Intersección sobre Unión): La intersección de las regiones de la verdad predicha y la verdad del terreno divididas por su unión.

#### 1.3.3.2. Matriz de confusión

| | | Ground truth | |
|---|---|---|---|
| | | HCP(1) | Background (0) |
| Prediction | HCP(1) | TP | FP |
| | Background(0 | FN | TN |

Para interpretación de estos términos en contexto, el objetivo en este desafío es segmentar una máscara de herida para cada imagen de herida corto punzante. Cada máscara está representada por 0 para el fondo o 1 para la herida corto-punzante (HCP).

- **TP (Verdadero Positivo):** representa el número de píxeles HCP que se han clasificado correctamente como HCP.
- **FP (falso positivo):** representa el número de píxeles de fondo que se clasifican erróneamente como HCP (debido a una desalineación).

- **FN (falso negativo):** representa el número de píxeles HCP que se clasifican erróneamente como fondo.
- **TN (Verdadero** representa el número de píxeles de fondo que se han clasificado correctamente como fondo.

A partir de la gráfica se puede profundizar en cada métrica de evaluación y para mejor comprensión unas definiciones de estas.

#### 1.3.3.3. Recall o recuperación:

Para la clasificación de píxeles: **La** puntuación de Recall, también conocida como *sensibilidad* o *tasa de positivos verdaderos*, es el número de resultados positivos verdaderos dividido por el número de todas las muestras que deberían haberse identificado como positivas.

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

#### 1.3.3.4. Exactitud/Accuracy:

La puntuación de precisión o en inglés Accuracy es el número de predicciones correctas, que consiste en predicciones positivas y negativas correctas divididas por el número total de predicciones.

$$Exactitud = \frac{TP+TN}{TP+TN+FN+FP}$$

#### 1.3.3.5. Coeficiente de DICE:

También conocido como índice de Sorensen-Dice o F1-Score binario, es una métrica que mide la similitud entre dos conjuntos, en este caso, el conjunto de píxeles etiquetados como positivos en la segmentación predicha y el conjunto de píxeles etiquetados como positivos en la segmentación de referencia (ground truth).

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

#### 1.3.3.6. Índice Jaccard (IoU):

El **índice de Jaccard**, también conocido como **Intersección sobre Unión (IoU),** es el área de la intersección sobre la unión de la segmentación predicha y la verdad del terreno.

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

#### 1.3.3.7. Puntuación F1):

La **Puntuación F1**, es una métrica que combina la precisión (precision) y la recuperación (recall) en una sola métrica. La fórmula de la puntuación F1 es:

$$F1 = \frac{2x\text{Precision}x\text{Recall}}{\text{Precision}+\text{Recall}}$$

El coeficiente de Dice y IoU son las métricas más utilizadas para la segmentación semántica porque ambas métricas penalizan los falsos positivos, que es un factor común en conjuntos de datos desequilibrados de clase alta como MIS. Sin embargo, la elección del coeficiente de Dice sobre IoU o viceversa se basa en casos de uso específicos de la tarea.

El coeficiente de Dice y la puntuación F1 son métricas relacionadas que se utilizan para evaluar la calidad de las segmentaciones, pero el coeficiente de Dice se enfoca en la similitud y la superposición entre conjuntos, mientras que la puntuación F1 combina precisión y recuperación en una única métrica. Ambas métricas son útiles para evaluar la precisión de los modelos de segmentación.

Cabe resaltar que el contexto de la evaluación del rendimiento de los modelos de aprendizaje automático y sobre todo en cuestión de valores paramétricos, el término en inglés «Accuracy» puede tener dos significados diferentes dependiendo del contexto:

1. Precisión (Accuracy): Es una métrica general que mide la proporción de predicciones correctas entre el número total de predicciones. Se calcula como el número total de predicciones correctas dividido por el número total de predicciones.

$$\text{Precisión (Accuracy)} = \frac{\text{Número de Predicciones Correctas}}{\text{Número Total de Predicciones}}$$

2. Precisión (Precision): Es una métrica que se utiliza en el contexto de los problemas de clasificación binaria y mide la proporción de verdaderos positivos (predicciones positivas

correctas) entre todas las predicciones positivas. Se calcula como el número de verdaderos positivos dividido por la suma de los verdaderos positivos y los falsos positivos, justo como en 40.

$$\text{Precisión (Precision)} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Positivos}}$$

# Capítulo 2

# Desarrollo del Sistema de Reconocimiento y Segmentación de Heridas

Este capítulo presenta el proceso de desarrollo de un algoritmo de visión artificial basado en aprendizaje profundo (Deep Learning), diseñado para segmentar y reconocer correctamente las heridas de tipo corto punzante para futuras aplicaciones en la línea de investigación de sutura autónoma. Se describe en detalle el procedimiento de codificación del algoritmo, así como la exploración de los modelos de redes neuronales convolucionales entrenados para la extracción de características de las imágenes de heridas cortantes y su segmentación semántica. Este proceso se especifica en detalle, abarcando desde la obtención y preparación de los datos hasta la definición y entrenamiento de los modelos. Por último, se analizan los resultados del entrenamiento de los modelos: U-Net2D, SegNet y DeepLabV3+ con ResNet50. Y se realizan ajustes en sus parámetros con el objetivo de optimizar su precisión. Estas modificaciones permiten obtener conclusiones preliminares acerca de la capacidad de los modelos para predecir con precisión nuevas heridas.

## 2.1. Entornos de implementación de redes neuronales en Python: Utilizando TensorFlow, Keras y Google Colab

Antes de elegir el modelo de CNN adecuado para la segmentación y reconocimiento de heridas, se realizó una evaluación de las necesidades y recursos disponibles. Esto incluyó desde la experiencia previa en la creación y entrenamiento de redes neuronales; hasta la calidad y cantidad de los datos de entrenamiento, así como el hardware y software disponibles para el entrenamiento y evaluación de los modelos.

Teniendo en cuenta las necesidades y recursos disponibles para implementar los modelos de redes neuronales y aportar al objetivo de la investigación, se opta por utilizar entornos de desarrollo basados en la nube como Google Colab, que proporcionan una interfaz de bloc de notas de Jupyter y viene preinstalado con bibliotecas populares de Python como TensorFlow y Keras. En el caso específico de Google Colab, este entorno ofrece numerosas ventajas, como el acceso gratuito o de paga a potente infraestructura informática que incluye GPU y TPU, acelerando significativamente los cálculos para modelos de aprendizaje automático y otras tareas computacionalmente intensivas[57].

Además, los cuadernos de Google Colab se pueden compartir y colaborar fácilmente, convirtiéndolos en una herramienta popular para los equipos de ciencia de datos y la educación en línea. Por otro lado, existen más lenguajes de programación que admiten la implementación de redes neuronales, como MATLAB[1], R[2] y Julia[3]. Sin embargo, Python es actualmente el lenguaje más popular y ampliamente utilizado para el aprendizaje profundo debido a su simplicidad, flexibilidad y amplio soporte de la comunidad de código abierto [58].

En resumen, Google Colab se presenta como una poderosa herramienta de programación de CNN y Python como el lenguaje más popular y mejor documentado para programadores, disponiendo de una amplia gama de recursos, herramientas y funcionalidades para construir, entrenar y evaluar redes neuronales y en general experimentar con tensores en

[1]MATLAB: entorno de programación y desarrollo de software ampliamente utilizado en ingeniería y ciencias, conocido por su capacidad para realizar cálculos numéricos avanzados, análisis de datos y visualización de resultados de manera eficiente

[2]R: lenguaje de programación y un entorno de software diseñado específicamente para estadísticas y análisis de datos. Es una herramienta poderosa para la manipulación, visualización y modelado de datos

[3]Julia: lenguaje de programación de alto rendimiento diseñado para aplicaciones científicas y técnicas. Se destaca por su velocidad de ejecución y cálculos numéricos intensivos

el campo del aprendizaje automático.

## 2.2. Selección de la red neuronal para segmentación de imágenes

Para el planteamiento del problema de segmentación de heridas saturables, se llevó a cabo una exhaustiva investigación de la literatura sobre los sistemas de Machine Learning (ML) y sus avances en el reconocimiento de características en imágenes biomédicas. Esta búsqueda bibliográfica permitió concluir que las Redes Neuronales Convolucionales (CNN) representan una excelente alternativa para llevar a cabo la clasificación pixel a pixel, también conocida como segmentación semántica. Esto se debe a la capacidad inherente de las CNN para procesar imágenes digitales y aprender características específicas de ellas, tales como formas, texturas y patrones [59].

Es relevante destacar que cada modelo de CNN posee sus propias fortalezas y debilidades, lo que los hace más adecuados para ciertas aplicaciones o tipos de imágenes. La selección cuidadosa de una CNN apropiada para la segmentación de imágenes de heridas saturables representa un paso crítico en el proceso de clasificación. Con una arquitectura bien seleccionada y un conjunto de datos de entrenamiento representativo, es posible lograr una segmentación semántica precisa de las imágenes, lo que facilitará un diagnóstico rápido y preciso de las heridas saturables. En el contexto de este proyecto, se optó por explorar modelos teniendo en cuenta su costo computacional, precisión, historial en aplicaciones médicas previas y su capacidad para funcionar correctamente con un conjunto de datos pequeño.

La selección de los modelos de redes neuronales se basó en una investigación sistemática a partir de múltiples publicaciones a cerca de estudios comparativos de las principales arquitecturas de CNN más recientes, como Kumar et al. [60] aplican. La tabla 2.1 compara algunos modelos de CNN que permiten la segmentación semántica en imágenes y se resaltan sus principales ventajas y desventajas:

| **Descripción** | **Ventajas** | **Desventajas** |
|---|---|---|
| U-Net: CNN que utiliza una arquitectura de codificador-decodificador para segmentar imágenes. | Eficiente en términos de uso de recursos computacionales.<br>Puede ser entrenado en un conjunto de datos relativamente pequeño.<br>Preserva la información de baja y alta resolución en la generación de la máscara de segmentación.<br>Valiosa para la segmentación y clasificación de imágenes médicas. | Tendencia al sobre-ajuste.<br>Limitaciones en la escalabilidad.<br>Problemas en la identificación de objetos pequeños.<br>Requiere de datos etiquetados. |
| ResNet: Es conocido por su estructura profunda, que permite que la red aprenda funciones residuales en lugar de mapear la entrada directamente a la salida. | Efectivo para el aprendizaje de características de imagen en grandes conjuntos de datos.<br>Efectivo en aplicaciones de transferencia de aprendizaje. | No está diseñada específicamente para la segmentación semántica.<br>Dificultad en la identificación de bordes.<br>Efectivo para el aprendizaje de características de imagen en grandes conjuntos de datos. |
| RefiNet: Utiliza una arquitectura de múltiples rutas para fusionar características de diferentes niveles de la red y mejorar la precisión de la segmentación. | Permite múltiples rutas de retroalimentación que permiten realizar una segmentación refinada.<br>Arquitectura modular que permite su uso en diferentes tareas de visión por computadora.<br>Puede ser entrenada con conjuntos de datos pequeños y grandes. | Requiere más datos de entrenamiento.<br>Mayor complejidad computacional, requiriendo más memoria y potencia de cálculo para entrenar y utilizar. |

Continúa en la siguiente página

**Tabla 2.1 – continuación de la página anterior**

| Descripción | Ventajas | Desventajas |
|---|---|---|
| FCN: Una red neuronal completamente convolucional que puede procesar imágenes de cualquier tamaño. | Capacidad para segmentar imágenes de cualquier tamaño.<br>Eficiencia computacional en términos de tiempo de entrenamiento y memoria necesaria.<br>Capacidad de generalización, puede ser entrenada en un conjunto de datos y ser utilizada para segmentar imágenes de otro conjunto de datos. | Problemas para mantener la resolución de las imágenes.<br>Se pierde información espacial y detalles importantes de la imagen.<br>Requiere grandes cantidades de datos de entrenamiento para evitar el sobre-ajuste.<br>Baja precisión en objetos pequeños. |
| Mask R-CNN: Extiende de la arquitectura de detección de objetos R-CNN con una rama de segmentación para cada objeto detectado. | Precisión en la detección y segmentación.<br>Capacidad para segmentar múltiples objetos de diferentes tamaños y formas.<br>Adecuada para detección objetos en tiempo real y la segmentación de imágenes médicas. | Requiere una gran cantidad de recursos computacionales para el entrenamiento y la inferencia.<br>Requiere grandes cantidades de datos de entrenamiento para obtener resultados precisos.<br>Es una arquitectura compleja. |
| SegNet: Usa una técnica de Max-pooling índices en el encoder, para almacenar información sobre la ubicación de los píxeles en la imagen original. | Altamente preciso en la segmentación de imágenes<br>Eficiente en términos de tiempo de entrenamiento y memoria necesaria.<br>Aplicada en la segmentación de imágenes médicas y satelitales. | Problemas para mantener la resolución.<br>Pierde información espacial.<br>Requiere grandes cantidades de datos. |
| Pyramid Scene Parsing Network: PSPNet utiliza una pirámide de características para obtener información contextual en diferentes escalas. | Capacidad de segmentar imágenes grandes sin necesidad de dividirlas.<br>Flexibilidad de uso, como la segmentación de objetos en tiempo real, la clasificación de escenas y la detección de objetos. | Arquitectura compleja que requiere una gran cantidad de recursos computacionales.<br>El entrenamiento puede llevar varias horas o incluso días.<br>Requiere grandes cantidades de datos de entrenamiento para obtener resultados precisos. |

Continúa en la siguiente página

**Tabla 2.1 – continuación de la página anterior**

| **Descripción** | **Ventajas** | **Desventajas** |
|---|---|---|
| DeepLab: Basada en la arquitectura de codificador-decodificador y una red de atracción de contexto para la segmentación de imágenes. | Altamente precisa para la segmentación de imágenes y la detección de objetos<br>Amplia gama de aplicaciones.<br>Eficiencia computacional ya que utiliza técnicas de reducción de dimensionalidad y una arquitectura de red más delgada que otras arquitecturas. | Requiere grandes cantidades de datos de entrenamiento para obtener resultados precisos.<br>Sensible a la calidad de la imagen de entrada.<br>Dificultad de implementación y uso por su complejidad. |

Tabla 2.1: Algunas arquitecturas para tareas de segmentación en imágenes

La tabla 2.2 relaciona los criterios de selección empleados, teniendo en cuenta principalmente la precisión de los modelos en tareas de segmentación semántica de imágenes, el costo computacional, su historial en aplicaciones médicas previas y la cantidad de datos necesarios para el entrenamiento con resultados óptimos.

| Modelo | Costo Computacional | Precisión | Aplicaciones Médicas | Data set Pequeño |
|---|---|---|---|---|
| U-Net | Moderado | Alta | Sí | Sí |
| ResNet | Alto | Alta | Sí | No |
| RefiNet | Bajo-Moderado | Alta | No | Sí |
| FCN | Moderado | Alta | Sí | No |
| Mask R-CNN | Alto | Muy Alta | Sí | No |
| SegNet | Bajo-Moderado | Media | No | Sí |
| PSPNet | Moderado | Alta | Sí | No |
| DeepLab | Moderado | Muy Alta | Sí | No |

Tabla 2.2: Relación entre modelos de CNN y criterios de Selección empleados.

Para elegir los modelos con mejores resultados en términos de precisión y menor costo de entrenamiento, se consideraron modelos utilizados previamente en tareas de segmentación semántica en imágenes. De entre los modelos más prominentes en la literatura, se optó por U-Net2D, SegNet y DeepLabV3+ con ResNet50 para abordar la problemática de la segmentación semántica de heridas planteada. Esta elección de modelos se fundamenta en su eficacia previamente demostrada en aplicaciones similares y se adecúa a los objetivos de la presente investigación.

U-Net2D y SegNet son, en primera instancia, arquitecturas de redes neuronales convolucionales (CNN) diseñadas específicamente para la segmentación semántica de imágenes médicas. Ambos modelos se caracterizan por emplear una estructura codificador-decodificador que es eficaz para este propósito.

Por otra parte, DeepLabV3+ con ResNet50 como codificador es un modelo de segmentación semántica que se destaca por su profundidad en el uso de redes neuronales convolucionales, lo cual le permite realizar segmentación semántica de manera precisa. A diferencia de U-Net2D y SegNet, DeepLabV3+ utiliza una técnica denominada "convolución atrous" para expandir el campo receptivo y capturar características de alto nivel, sin necesidad de incrementar la cantidad de parámetros.

### 2.2.1. U-Net2D

La red *U-Net* es una arquitectura de CNN diseñada para abordar tareas de segmentación semántica en imágenes médicas y biomédicas, basada en un encoder-decoder con conexiones de concatenación. Fue desarrollada por Olaf Ronneberger et al.[61] y su nombre proviene de su estructura en forma de 'U'o simétrica. Es ampliamente utilizada debido a su capacidad para capturar tanto información de contexto como detalles finos en las imágenes, además es efectiva para procesar imágenes de alta resolución con pocos datos de entrenamiento. Esta red es popular en aplicaciones médicas debido a su eficacia y versatilidad, así como su capacidad para segmentar con precisión bordes y texturas de las lesiones, debido a las conexiones de concatenación que permiten fusionar las características de alto y bajo nivel del encoder con el decoder, lo que mejora la precisión de la segmentación.

Por otro lado, su amplia adopción en la comunidad de aprendizaje profundo y su eficiencia en el uso de recursos computacionales la hace ideal para trabajar con conjuntos de datos relativamente pequeños y garantiza que haya herramientas y documentación disponibles para su implementación.

#### 2.2.1.1. Arquitectura

La arquitectura de la red U-Net 2D se compone de dos rutas distintas. La primera de ellas, conocida como la ruta de contracción o codificador, consiste en una serie convencional de capas de agrupación convolucional y máxima. Esta ruta tiene como objetivo capturar

el contexto en la imagen, reduciendo gradualmente su tamaño mientras profundiza en las características. Esto permite a la red aprender información sobre qué elementos se encuentran en la imagen.

La segunda ruta, denominada la ruta de expansión simétrica o decodificador, se utiliza para lograr una localización precisa de las instancias presentes en la imagen. De manera intuitiva, el decodificador recupera información de ubicación o dónde se encuentran estas instancias. Esto se logra aplicando un proceso gradual de muestreo ascendente, lo que aumenta el tamaño de la imagen mientras disminuye su profundidad hasta alcanzar la forma original de la imagen de entrada.

En resumen, la U-Net 2D es una red totalmente convolucional (FCN) de extremo a extremo, lo que significa que está compuesta únicamente por capas convolucionales y no contiene ninguna capa densa. Esto le permite aceptar imágenes de cualquier tamaño, lo que la hace altamente versátil en aplicaciones de segmentación semántica. La figura 2.1 muestra la arquitectura del modelo empleado en este proyecto.

Como se puede observar en la imagen 2.1, la arquitectura de U-Net2D empleada se caracteriza por su estructura codificador-decodificador, que consta de las siguientes etapas clave:

- **Entrada (Input)**: La red toma una imagen de entrada con dimensiones (batch, `input_dim_x`, `input_dim_y`, `num_channels`) que representan el tamaño del lote, ancho, alto y canales de color respectivamente. Para la investigación se están empleando imágenes de la forma $(1, 224, 224, 3)$
- **Codificador (Encoder)**: Esta etapa se compone de cuatro bloques convolucionales con diferentes niveles de profundidad y resolución. Cada bloque convolucional se compone de dos o tres capas convolucionales con función de activación ReLU para aprender características de alto nivel de la imagen de entrada y aumenta progresivamente el número de filtros para extraer información contextual. El número de filtros comienza con `n_filters= 32` y se duplica en cada bloque del codificador.
- **Capas de Agrupación (Max-Pooling)**: Después de cada bloque de convolución, se aplica una capa de agrupación máxima (Max-pooling) para reducir la resolución espacial de las características a la mitad y mantener solo la información más relevante.

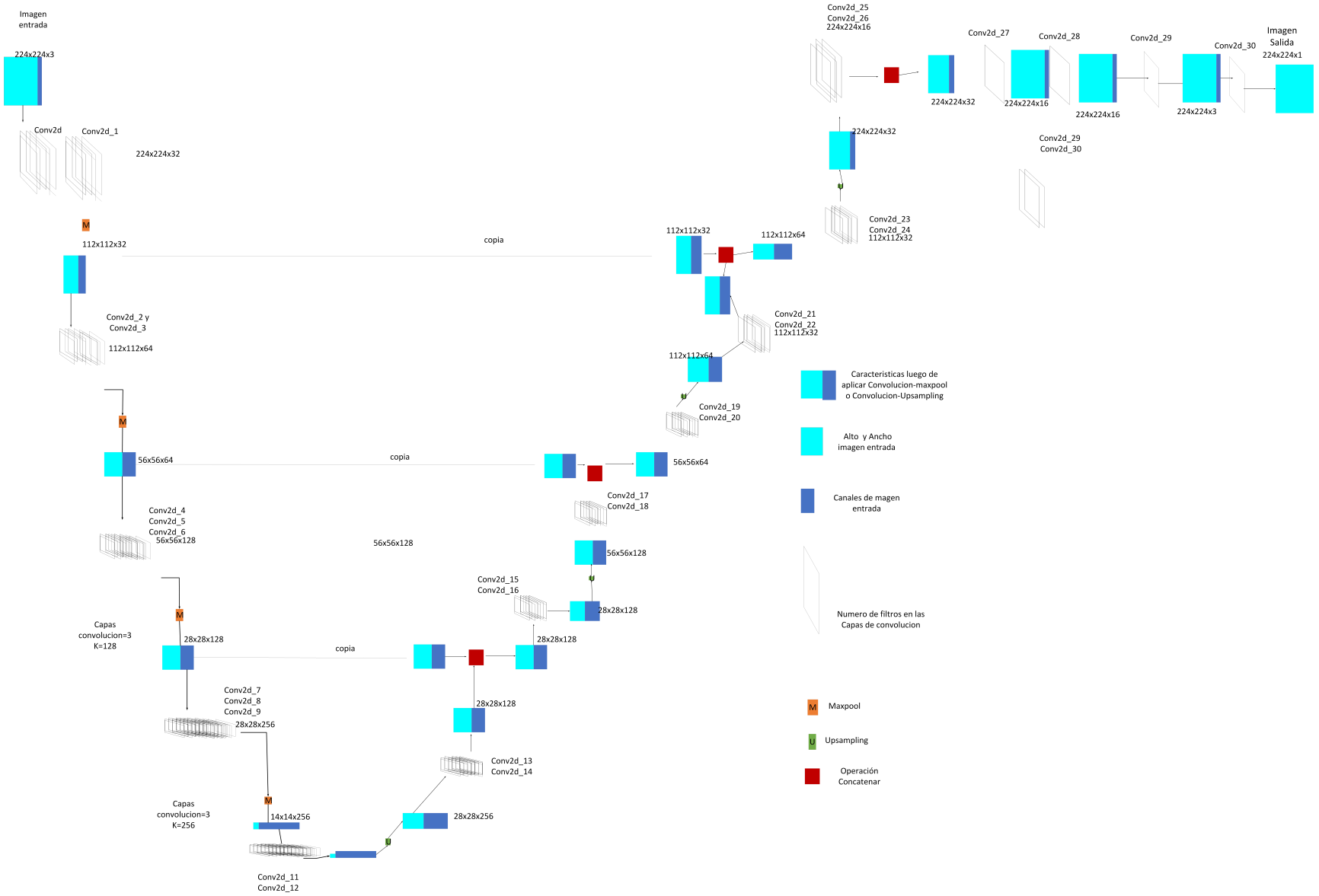



Figura 2.1: Descripcion de la Arquitectura Red U-Net2D obtenida

- **Decodificador (Decoder)**: Luego, la red pasa a una etapa de decodificación donde el decoder es una red convolucional simétrica al encoder, que se compone de cuatro bloques deconvolucionales. Cada bloque deconvolucional contiene una capa de upsampling que aumenta la resolución al doble, seguida de una capa de concatenación que fusiona las características del bloque correspondiente del encoder, y dos o tres capas convolucionales. En esta etapa, las características de alto nivel aprendidas en el codificador se combinan con las características en la etapa de upsampling para obtener una representación más detallada de la imagen.
- **Salida (Output)**: Finalmente, la red produce una salida que tiene la misma forma que la entrada original (1,224,224,3). La última capa utiliza la función de activación sigmoide para generar una máscara de segmentación binaria que identifica las áreas de interés en la imagen (1,224,224,1).

#### 2.2.1.2. Parámetros del modelo

Es importante identificar parámetros de los modelos como el número de capas, los filtros por capa o las funciones de activación utilizadas como se muestra a continuación, si desea ver una lista más completa de los parámetros y capas del modelo puede encontrarlos en el anexo A.2

- **Número de Capas**: U-Net2D consta de múltiples capas convolucionales en el codificador y el decodificador para capturar características complejas, distribuidas de la siguiente forma:
  - Capas de convolución (Conv2D): 32 capas.
  - Capas de reducción (MaxPooling2D): 4 capas.
  - Capas de expansión (UpSampling2D): 4 capas.
- **Filtros por capa**: El número de filtros por capa aumenta progresivamente en el codificador, donde se multiplica el número de filtros por 2 en cada bloque de convolución. Esto permite que la red aprenda características de nivel creciente de abstracción. En nuestro caso se ha comenzado con n_filters= 32 para el primer bloque, seguido de 64 filtros para el segundo bloque, 128 filtros para el tercer bloque y 256 filtros para el cuarto bloque.

- **Tamaño del Kernel** Todas las capas de convolución (Conv2D) utilizan un kernel de tamaño 3x3, excepto las capas de upsampling, donde se utiliza un kernel de tamaño 2x2 para duplicar la resolución de la imagen mientras se asciende.
- **Función de Activación**: En todo el modelo, se utiliza la función de activación ReLU (Rectified Linear Unit) para introducir no linealidad en las capas convolucionales (Conv2D). A excepción de la capa de salida, donde se utiliza una función sigmoide para generar una máscara de segmentación binaria de la forma (1,224,224,1).

### 2.2.2. SegNet

SegNet es una arquitectura de CNN de igual manera basada en un encoder-decoder, pero con conexiones de salto, que fue desarrollada por Vijay Badrinarayanan, Alex Kendall y Roberto Cipolla. Fue diseñada específicamente para tareas de segmentación semántica en imágenes [62]. Esta arquitectura se ha ganado su lugar destacado en la comunidad de investigadores y profesionales gracias a su capacidad para aprender y reconstruir características de nivel detallado en las imágenes con un costo computacional muy bajo. Tal destreza ha impulsado su efectividad en diversas aplicaciones, incluyendo las aplicaciones médicas, donde ha demostrado su efectividad en la segmentación de heridas gracias a las conexiones de salto que permiten transferir la información de los índices de los Max-pooling del encoder al decoder, lo que ayuda a preservar los detalles de los bordes [63].

#### 2.2.2.1. Arquitectura

La arquitectura del modelo SegNet al igual que la del modelo U-Net2D, se distingue por su enfoque único, compuesta la estructura encoder-decoder. En la ruta de codificación, se capturan las características esenciales en la imagen, mediante una secuencia de capas convolucionales y de agrupación, permitiendo que la red aprenda gradualmente el contexto y las características de la imagen al reducir su tamaño. De esta manera, la red puede discernir qué elementos se encuentran en la imagen con mayor claridad.

De forma análoga al modelo anterior, en la etapa de decodificación se recupera la información sobre la ubicación precisa de las instancias, logrando que la salida final sea coherente con la forma original de la imagen de entrada.

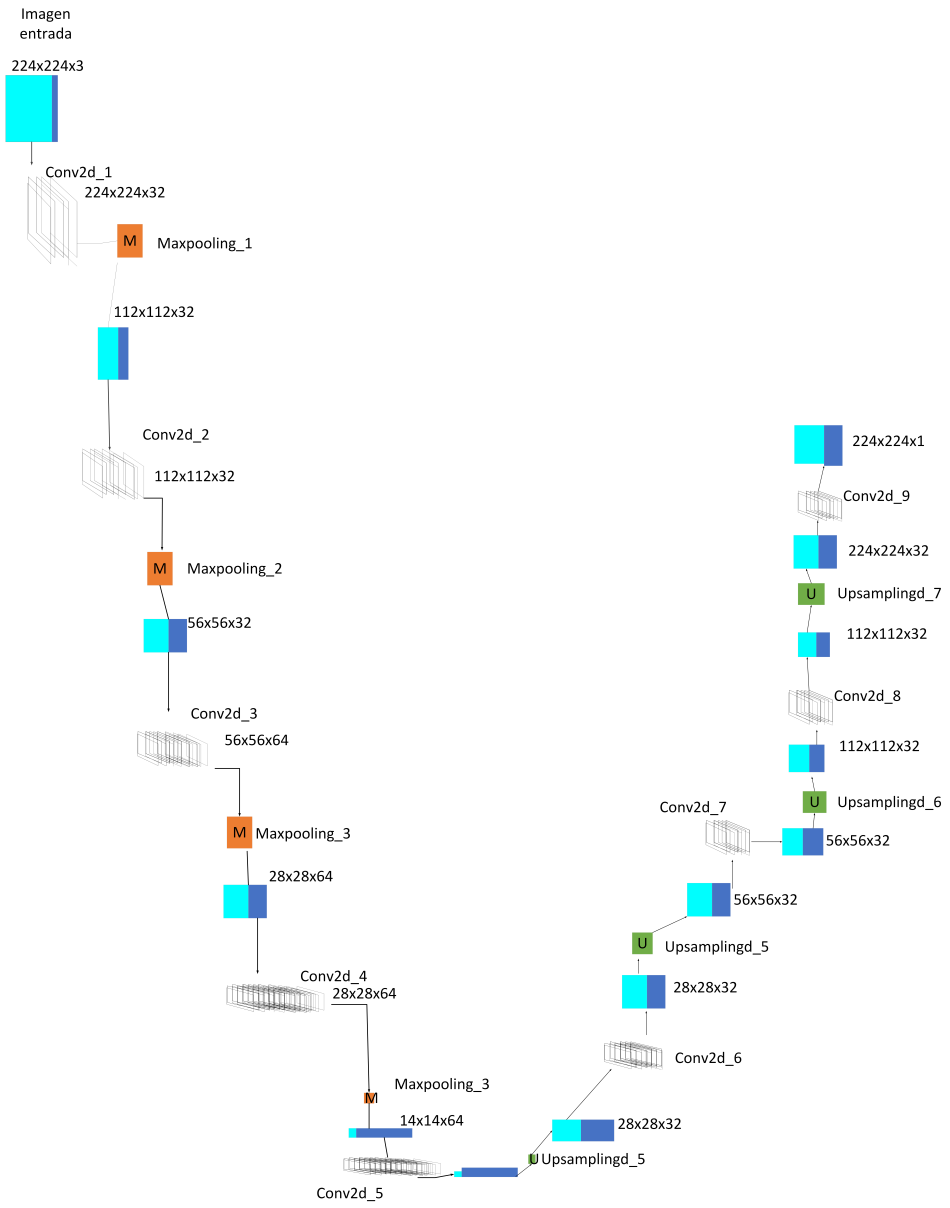



Figura 2.2: Arquitectura de la Red Segnet

La figura 2.2 muestra la representación gráfica de la arquitectura del modelo SegNet empleado y la conexión entre sus capas.

#### 2.2.2.2. Parámetros del modelo

Teniendo en cuenta los mismos parámetros clave de para todos los modelos como el número de capas, los filtros por capa o las funciones de activación empleados, en el caso de SegNet, se muestran a continuación:

- **Número de capas**: SegNet consta de múltiples capas convolucionales en el codificador y el decodificador. En la implementación diseñada, el codificador tiene 4 capas de convolución, y el decodificador tiene 4 capas de convolución transpuesta o UpSampling. Distribuidas como se lista a continuación:
  - Capas de convolución (Conv2D): 9 capas.
  - Capas de reducción (MaxPooling2D): 4 capas.
  - Capas de expansión (UpSampling2D): 4 capas.
- **Filtros por capa**: En la implementación diseñada, se utilizan los siguientes en cada sección:
  - En el encoder, se mantuvo el valor de `n_filters= 32` durante las dos primeras capas Conv2D, mientras que en las dos capas posteriores se duplicó el valor `n_filters` con el fin de alcanzar un nivel creciente de abstracción.
  - En todas las capas del decoder se mantuvo el valor `n_filters= 32`
- **Tamaño del Kernel**: El tamaño del kernel varía para diferentes capas en el modelo SegNet. A continuación se muestran los tamaños de kernel utilizados en las capas principales de los cuatro bloques del modelo:
  - Conv2D en el codificador: kernel_size=(9,5,5,5)
  - UpSampling2D en el decodificador: kernel_size=(7,5,5,1)
- **Función de Activación**: En toda la red, se utiliza la función de activación ReLU (Rectified Linear Unit)

Adicionalmente, la tabla 2.3 muestra en detalle las capas que componen el modelo SegNet así como los parámetros asociados a cada una.

| Layer | Type | Output Shape | N.º Parámetros |
|---|---|---|---|
| input_2 | InputLayer | (None, 224, 224, 3) | 0 |
| conv2d_31 | Conv2D | (None, 224, 224, 32) | 7,808 |
| max_pooling2d_4 | MaxPooling2D | (None, 112, 112, 32) | 0 |
| conv2d_32 | Conv2D | (None, 112, 112, 32) | 25,632 |
| max_pooling2d_5 | MaxPooling2D | (None, 56, 56, 32) | 0 |
| conv2d_33 | Conv2D | (None, 56, 56, 64) | 51,264 |
| max_pooling2d_6 | MaxPooling2D | (None, 28, 28, 64) | 0 |
| conv2d_34 | Conv2D | (None, 28, 28, 64) | 102,464 |
| max_pooling2d_7 | MaxPooling2D | (None, 14, 14, 64) | 0 |
| conv2d_35 | Conv2D | (None, 14, 14, 32) | 51,232 |
| up_sampling2d_4 | UpSampling2D | (None, 28, 28, 32) | 0 |
| conv2d_33 | Conv2D | (None, 28, 28, 32) | 50,208 |
| up_sampling2d_5 | UpSampling2D | (None, 56, 56, 32) | 0 |
| conv2d_37 | Conv2D | (None, 56, 56, 32) | 25,632 |
| up_sampling2d_6 | UpSampling2D | (None, 112, 112, 32) | 0 |
| conv2d_38 | Conv2D | (None, 112, 112, 32) | 25,632 |
| up_sampling2d_7 | UpSampling2D | (None, 224, 224, 32) | 0 |
| conv2d_39 | Conv2D | (None, 224, 224, 1) | 33 |

Tabla 2.3: Capas de la red SegNet

### 2.2.3. DeepLabV3+ con ResNet50

DeepLabV3+ con ResNet50 como backbone es una arquitectura avanzada de segmentación semántica que combina la potencia de la red convolucional profunda ResNet50 utilizada para extraer características de alto nivel de las imágenes, con el enfoque de Pyramid Pooling y dilated/atrous convolutions que refina los detalles de los bordes de los objetos segmentados. Este modelo está basado en el modelo DeepLabV3 el cual se inspira en el trabajo de Chen et al.[64], que propusieron una red convolucional con módulos de pirámide espacial atrous (ASPP) para capturar información a múltiples escalas.

Este modelo es ampliamente utilizado en tareas de segmentación de imágenes médicas debido a su capacidad para capturar información contextual y detalles finos en las imágenes para generar segmentaciones precisas y consistentes en diferentes condiciones de iluminación, contraste y resolución [65].

#### 2.2.3.1. Arquitectura

La arquitectura de DeepLabV3+ con ResNet50 como backbone sigue la estructura encoder-decoder como común denominador de los modelos seleccionados, esta estructura le permite obtener información de contexto de imagen en su encoder y fusionarla con la información de ubicación de las instancias en el decoder.

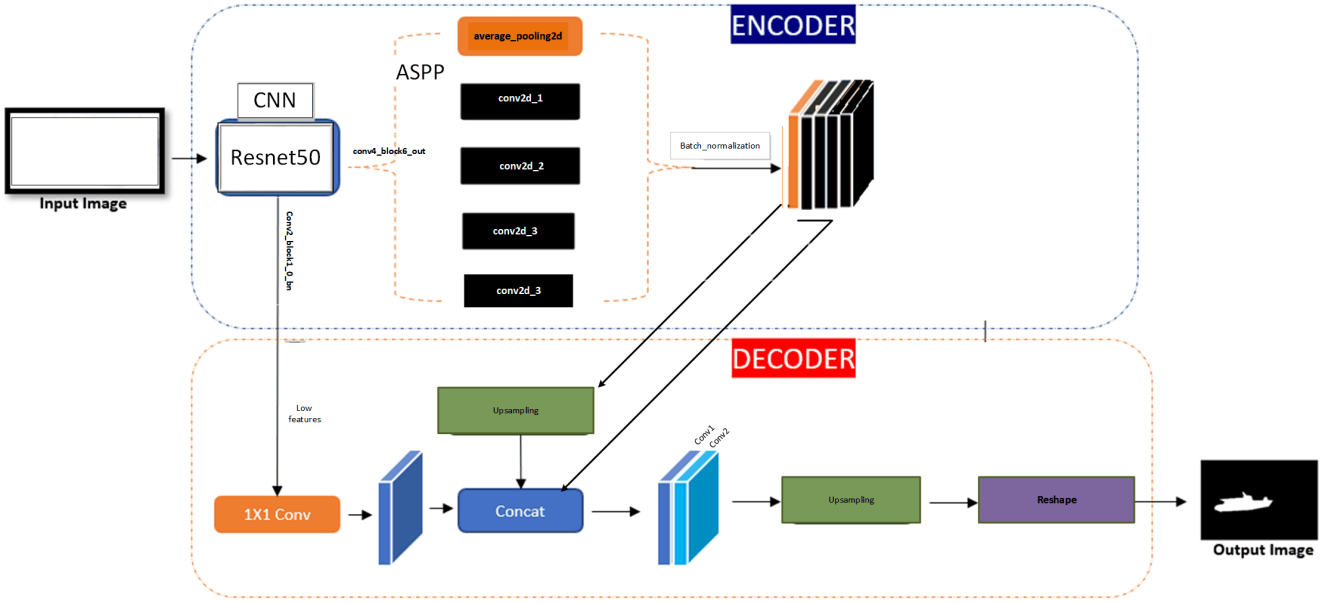



Figura 2.3: Representación de la Arquitectura generada de la Red DeeplabV3+

La figura 2.3 muestra la representación gráfica de la red y esta se compone de las siguientes etapas clave:

- **Encoder (ResNet50)**: Utiliza la red ResNet50 que se compone de cinco bloques convolucionales con diferentes niveles de profundidad y resolución, preentrenada con los pesos de ImageNet como un codificador para extraer características de alto nivel de la imagen de entrada. Las características se obtienen de la capa `conv4_block6_out` de ResNet50 lo que produce un mapa de características de baja resolución pero rico en contexto, que se pasa al módulo ASPP.
- **ASPP (Atrous Spatial Pyramid Pooling)**: En esta etapa, se aplica el módulo ASPP para capturar información contextual de diferentes escalas espaciales. Esto se logra mediante cuatro convoluciones atrous (o dilatadas) paralelas con diferentes tasas de dilatación, seguidas de una convolución de 1x1, a la que se añade una capa de agrupación promedio. Los cinco mapas de características resultantes se combinan

mediante una capa de concatenación y se escalan mediante la capa de Squeeze-and-Excite.

- **Decodificador**: El decoder es el módulo encargado de recuperar los detalles finos de los bordes de los objetos. Para ello, se realiza una fusión de características entre las salidas del ASPP y las características de nivel inferior (obtenidas de la salida del bloque `conv2_block2_out` del encoder ResNet50). Luego, se aplican capas convolucionales para refinar las características y se utiliza la convolución transpuesta (upsampling) para aumentar la resolución espacial usando una interpolación bilineal.
- **Salida**: La capa de salida utiliza una convolución 1x1 para generar una máscara de segmentación binaria que identifica las áreas de interés en la imagen. La función de activación sigmoide se aplica a la salida para obtener probabilidades de pertenencia a la clase de segmentación.

#### 2.2.3.2. Parámetros del modelo

Teniendo en cuenta la implementación de la red DeepLabV3+ usando ResNet50 como encoder, se listan a continuación los parámetros claves del modelo:

- **Número de Capas**: El modelo DeepLabV3+ con ResNet50 se compone principalmente de las capas de la red ResNet50, que determina la profundidad y la capacidad de extracción de características del encoder. En este caso, se utilizan 50 capas, lo que ofrece un buen equilibrio entre rendimiento y complejidad. Además, se aplican varias capas convolucionales adicionales en el decodificador.
- **Tasas de dilatación**: Las tasas de dilatación de las convoluciones atrous, controlan el tamaño del campo receptivo y la resolución espacial de las características extraídas. En este caso, se utilizan las tasas 6, 12 y 18 para el último bloque del encoder en el módulo ASPP, lo que permite capturar información a múltiples escalas sin perder resolución.
- **Filtros por Capa**: El número de filtros por capa, determina el número de canales o características que se generan en cada capa. En este caso, se utilizan 256 filtros para el módulo ASPP y el decoder, y 48 filtros para la proyección de las características de bajo nivel del encoder. Estos valores se escogen para reducir la dimensión y el coste computacional del modelo sin perder información relevante.

- **Funciones de activación**: En toda la red, se utiliza la función de activación ReLU (Rectified Linear Unit) que introducen no linealidades en el modelo y permiten aprender patrones complejos. Exceptuando la última capa que utiliza la función sigmoide para obtener la máscara de segmentación binaria, que representa la probabilidad de pertenencia a la clase herida.

Adicionalmente, el anexo A.3 muestra el resumen de las capas que conforman el modelo y sus parámetros asociados.

En resumen, se ha determinado que los modelos U-Net2D, SegNet y DeepLabV3+ con ResNet50 son elecciones sobresalientes para abordar la tarea de segmentación semántica de heridas cortantes en el contexto de este proyecto. Esto se debe en gran medida a su arquitectura encoder-decoder, que posibilita la extracción de características relevantes de las imágenes, contribuyendo significativamente a una mayor precisión en la clasificación. Esta estructura también habilita la aplicación efectiva de técnicas de aumento de datos, lo que amplía la capacidad en los modelos para generalizar el conocimiento adquirido de las características extraídas, a la vez que previene el riesgo de sobre ajuste. En conjunto, estos modelos elegidos se destacan como opciones sólidas y prometedoras para la segmentación semántica de heridas cortantes en el marco de este proyecto.

## 2.3. Implementación de las CNN para segmentación de heridas

Los modelos U-Net2D, SegNet y DeepLabV3+ con ResNet50 seleccionados han demostrado resultados óptimos al combinar la capacidad de aprendizaje profundo de las redes neuronales convolucionales con la preservación de la información espacial. En esta sección, se presentan las etapas clave necesarias para el entrenamiento exitoso de dichos modelos utilizando TensorFlow y Python, desde la obtención y preparación de los datos hasta la validación de los modelos. Se exploran además cada una de estas etapas en detalle, destacando las consideraciones importantes y las decisiones clave que deben tomarse a lo largo del proceso. Con esta guía, los investigadores y profesionales de la salud también podrían utilizar el poder de las redes neuronales para mejorar la precisión y eficiencia en sus tareas de diagnóstico y análisis de imágenes.

La figura 2.4 muestra una estructura general del proceso de entrenamiento de los modelos seleccionados.

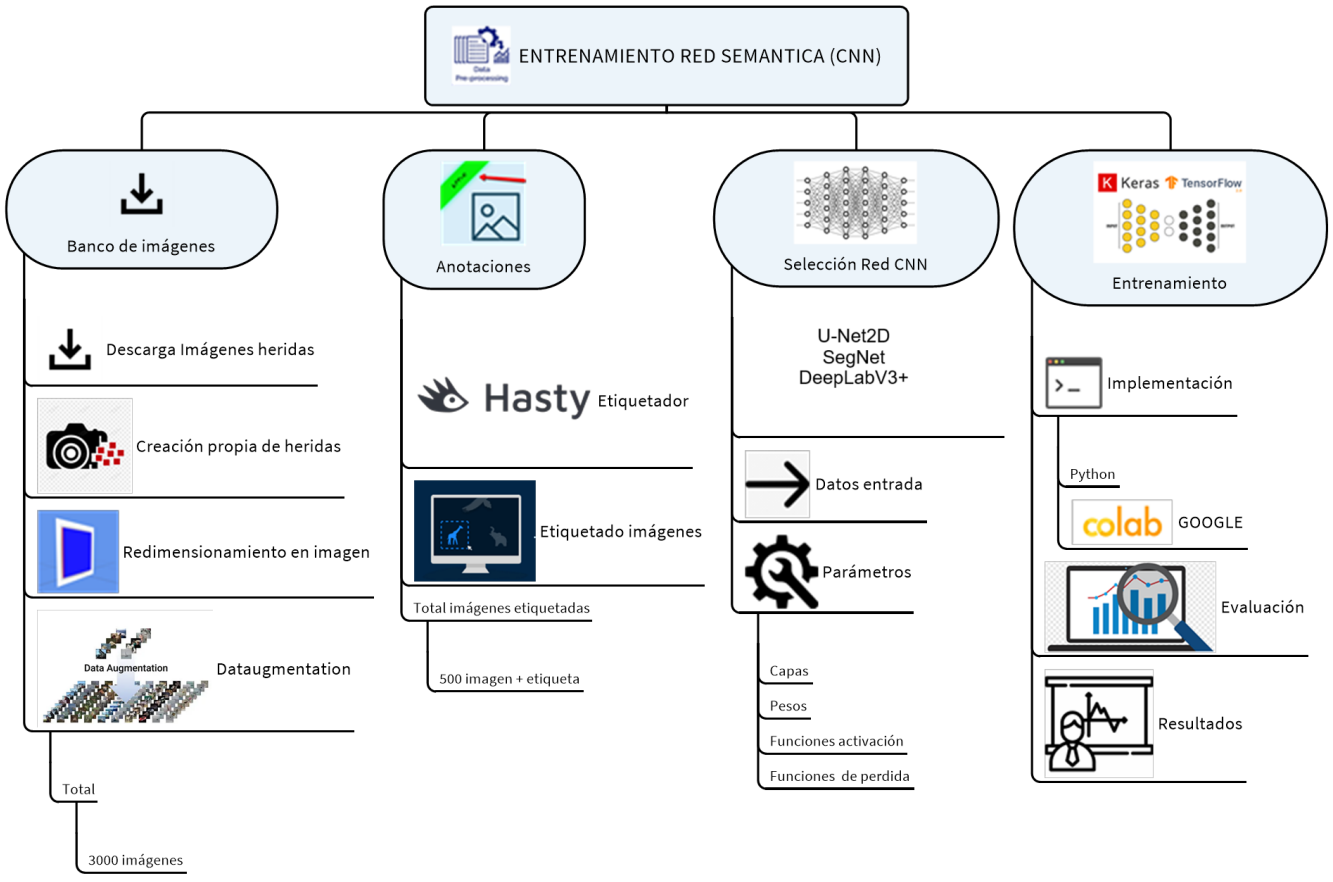



Figura 2.4: Esquema general del proceso de entrenamiento de los modelos

De igual manera, la figura 2.5 muestra las principales etapas del entrenamiento, partiendo desde la recolección y procesamiento de los datos, hasta el entrenamiento y validación del modelo. Además, relaciona las actividades para lograr la tarea de segmentación semántica en imágenes médicas.

### 2.3.1. Etapa 1: Obtención y preparación de los datos:

La etapa de obtención y preparación de los datos es fundamental para el entrenamiento exitoso de los modelos U-Net2D, SegNet y DeepLabV3+, en la tarea de segmentación semántica en imágenes médicas. En esta etapa, inicialmente se adquieren los datos necesarios y se aplican diversas técnicas de preprocesamiento a los mismos, para asegurar la calidad y coherencia de los datos de entrada a cada modelo. A continuación, se describen en detalle las actividades involucradas en esta etapa:

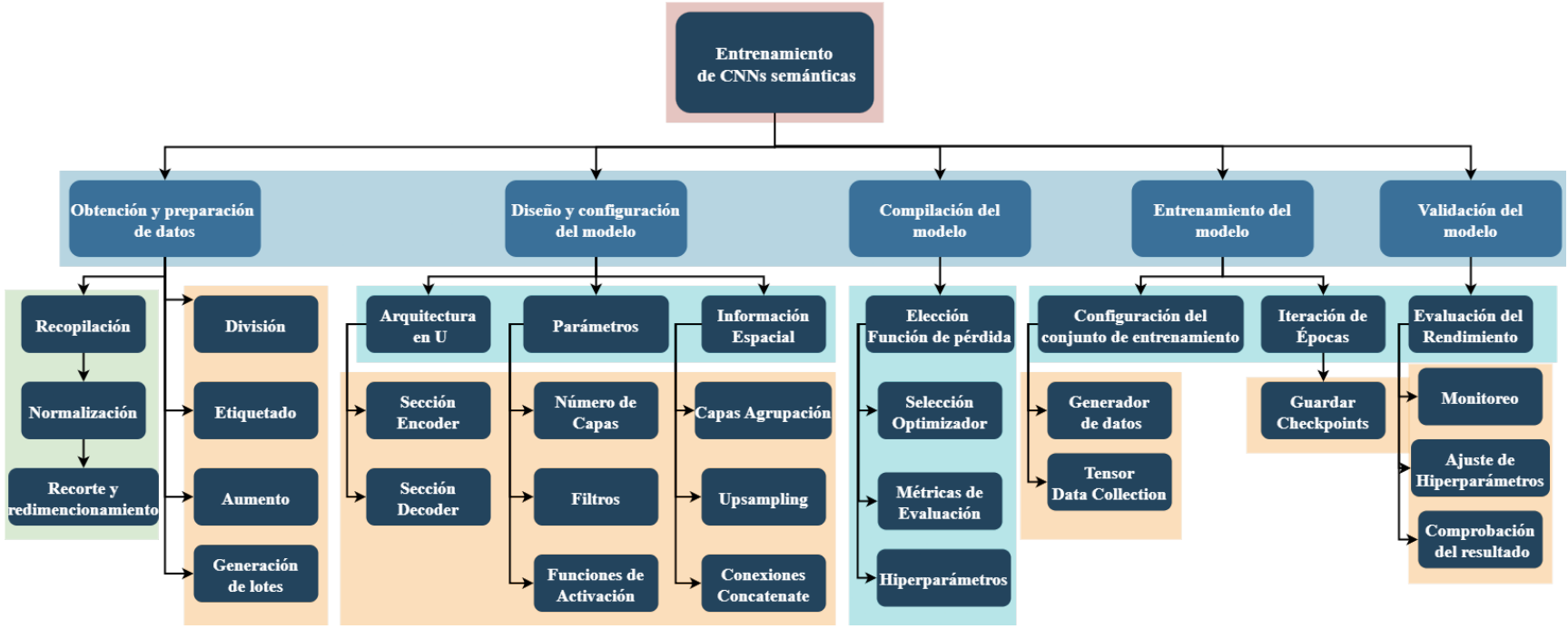



Figura 2.5: Etapas del entrenamiento de los modelos seleccionados

1. **Recopilación de datos:**

   La recolección de datos para la segmentación semántica de heridas corto-punzantes presenta diversas limitaciones que afectan la calidad y cantidad de información disponible. Una de las principales limitaciones radica en la escasez de datos de heridas cortantes, lo cual restringe la capacidad de entrenamiento de las redes neuronales utilizadas en este proceso. Esto se debe en gran medida a la falta de investigaciones enfocadas específicamente en este tipo de heridas, así como a las restricciones de acceso a imágenes que contienen sangre y heridas profundas. Con el objetivo de abordar esta limitación, se llevó a cabo una recopilación de imágenes médicas de heridas corto-punzantes. Estas heridas presentaban generalmente una forma ovalada y mostraban la separación de la piel, así como diferentes niveles de profundidad. Para obtener estas imágenes, se utilizaron diversas fuentes, como bases de datos médicas, repositorios públicos como Kaggle, simuladores de heridas, así como vídeos de procedimientos quirúrgicos de sutura. Puede encontrar más información sobre el origen de los datos en el anexo A.4

   Inicialmente, se recopiló una lista preliminar de 420 imágenes. Sin embargo, solo 349 de estas cumplían con los criterios de elegibilidad establecidos. Estos criterios incluían la presencia de una herida fácilmente identificable, una buena resolución de la imagen, la ausencia de exceso de sangre y la variabilidad en los escenarios representados. El objetivo era obtener imágenes claras y en condiciones adecuadas para aplicar un procedimiento como la sutura. La figura 2.6 proporciona un ejemplo de las heridas corto punzantes encontradas en las imágenes seleccionadas.

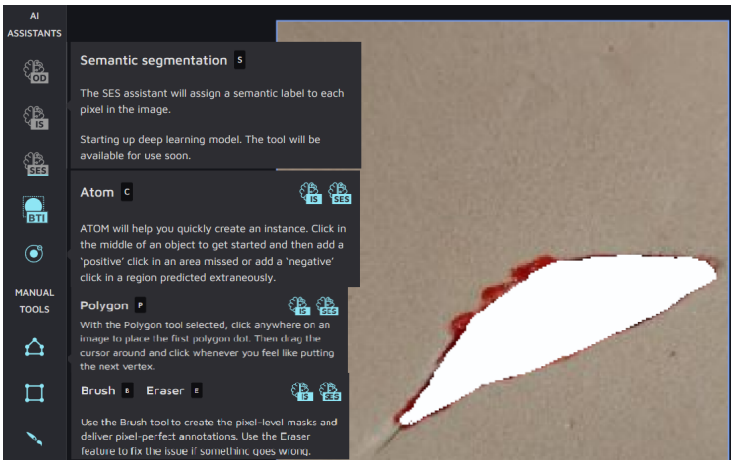

Figura 2.6: Ejemplos de heridas recuperadas de la web

**2. Anotación de datos:**

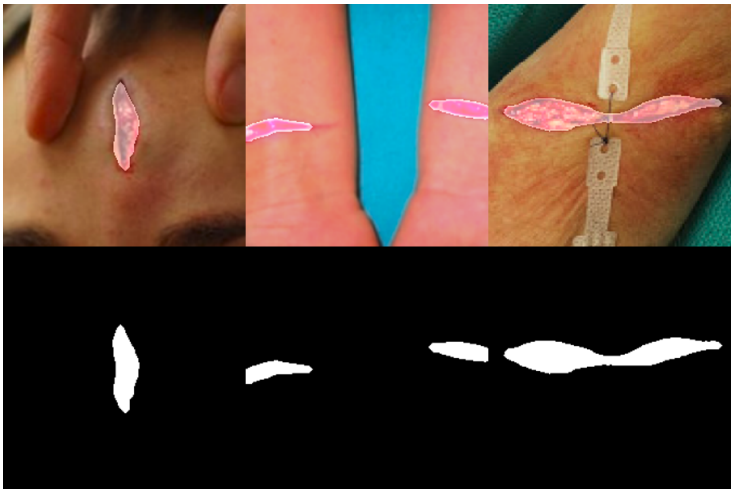
Debido a la variabilidad de los datos de entrenamiento disponibles, compuestos por imágenes de heridas corto-punzantes reales y artificiales, ninguno de los datos vienen previamente anotados, esto implicó la tarea de anotación o etiquetado manual de los datos. Se identificaron y etiquetaron las regiones de interés en cada imagen, marcando las áreas que se desean segmentar. Estas anotaciones fueron utilizadas como datos de referencia durante el entrenamiento de los modelos.

Para conseguir el etiquetado del conjunto de datos de entrenamiento, se utilizó **Hasty.ai** la cual es una plataforma de aprendizaje automático centrada en datos que ayuda a las empresas y personas con datos únicos a construir y desplegar aplicaciones de visión artificial más rápido y de manera más confiable. Aunque existen numerosas herramientas disponibles, se optó por Hasty por su interfaz intuitiva, asistentes de etiquetado basados en IA y su eficacia en la tarea de etiquetado y anotación de imágenes. Además, cuenta con su Asistente de Segmentación Semántica, que permite obtener una anotación más rápida y efectiva, puesto que cada anotación que se realice manualmente se envía a su entrenador de modelos para crear un modelo personalizado del proyecto, lo cual acelera el etiquetado con un factor de 10-100 veces.

La figura 2.7a muestra algunas de las herramientas de etiquetado manual que ofrece **Hasty.ai** y que al usarlas van entrenando el Asistente de Segmentación Semántica. Además, la figura 2.7b muestra un ejemplo de etiquetado y la respectiva máscara semántica generada por la aplicación.




(a) Herramientas manuales



(b) Etiquetado y máscara semántica

Figura 2.7: Etiquetado de imágenes usando Hasty.ai

**3. División de datos:**

Después de recolectar y anotar los datos de heridas corto-punzantes, se dividen en conjuntos de entrenamiento, validación y prueba. Como se muestra en la Figura 2.8, cada conjunto de datos tiene un propósito específico. El conjunto de entrenamiento se utiliza para ajustar los parámetros o pesos del modelo. El conjunto de validación se utiliza para ajustar los hiperparámetros y evaluar el rendimiento durante el entrenamiento. Finalmente, el conjunto de prueba se utiliza para evaluar el rendimiento final del modelo.

Debido al limitado número de imágenes que componen el conjunto de entrenamiento, se decidió dividir el lote de imágenes solo en subconjuntos de entrenamiento y validación, utilizando una proporción de 70 % y 30 % respectivamente. Esta división

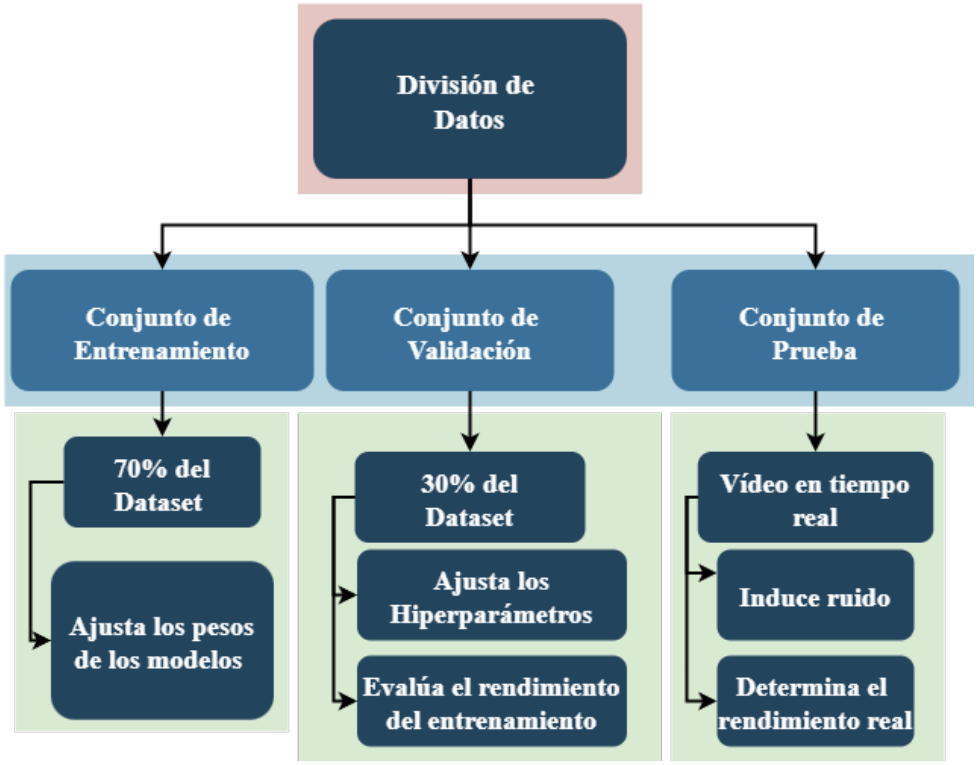



Figura 2.8: División de los datos recolectados

permitió evaluar el rendimiento de las redes y evitar el sobre ajuste.

Para definir el conjunto de prueba y determinar el rendimiento final de los modelos U-Net2D, SegNet y DeepLabV3+, se optó por grabar un vídeo en tiempo real. Esto permitió inducir condiciones de ruido como variaciones en la iluminación, cambios en el estado de la herida y desplazamiento del objetivo.

4. **Redimensionamiento y recorte de imágenes:**

El preprocesamiento de las imágenes desempeña un papel fundamental en el logro de la calidad y precisión del reconocimiento. Al analizar detenidamente el conjunto de entrenamiento recolectado, compuesto por imágenes reales y artificiales, se observa una variabilidad en los tamaños o resoluciones de las imágenes adquiridas, lo cual representa un desafío para la tarea de reconocimiento.

Con el fin de abordar esta situación, se implementa una etapa de redimensionamiento de las imágenes, buscando lograr dimensiones uniformes. En este caso particular, se optó por una resolución de 224x224 píxeles. Esta elección es significativa debido a que muchos algoritmos de reconocimiento requieren que las imágenes tengan el mismo tamaño para poder realizar comparaciones de manera efectiva. La uniformidad en

el tamaño facilita el procesamiento y entrenamiento y comparación de los modelos seleccionados.

Además del redimensionamiento, se lleva a cabo un recorte de las imágenes, especialmente en aquellas obtenidas de la web. Este proceso consiste en enfocarse en la región de interés o la herida corto-punzante en particular. Al realizar este recorte, se contribuye a la reducción de ruido al eliminar píxeles que no son relevantes para la tarea de segmentación. Esta estrategia permite mejorar la calidad de los datos utilizados para el entrenamiento del modelo y, a su vez, optimiza el desempeño del reconocimiento.

Para ilustrar visualmente los cambios realizados a las imágenes del conjunto de datos, la figura 2.9 muestra una imagen original y su versión pre procesada.

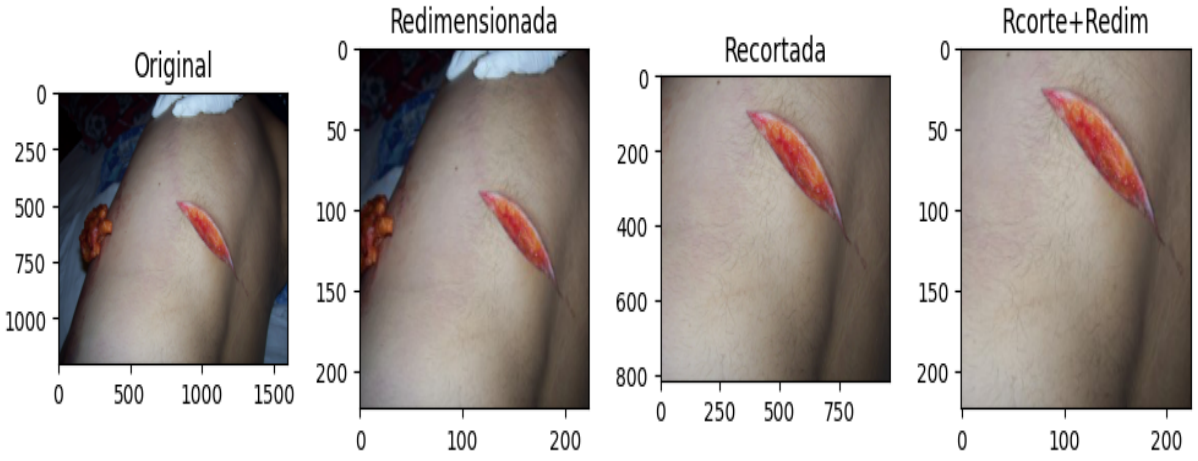



Figura 2.9: Comparación, imagen original y preprocesada

**5. Normalización de datos:**

Al analizar los datos recopilados de heridas corto-punzantes, se identificaron inconsistencias en los formatos de las imágenes y sus canales de colores, lo cual afecta su uniformidad. La Figura 2.10 muestra las modificaciones aplicadas al conjunto de datos para lograr la homogeneidad requerida. Estas modificaciones incluyeron cambios en los formatos, ajustes en los canales de color y normalización de la intensidad de los píxeles.

Algunas imágenes del conjunto de datos de entrenamiento preparado, estaban en formato JPG, mientras que otras se encontraban en formato PNG. La diferencia

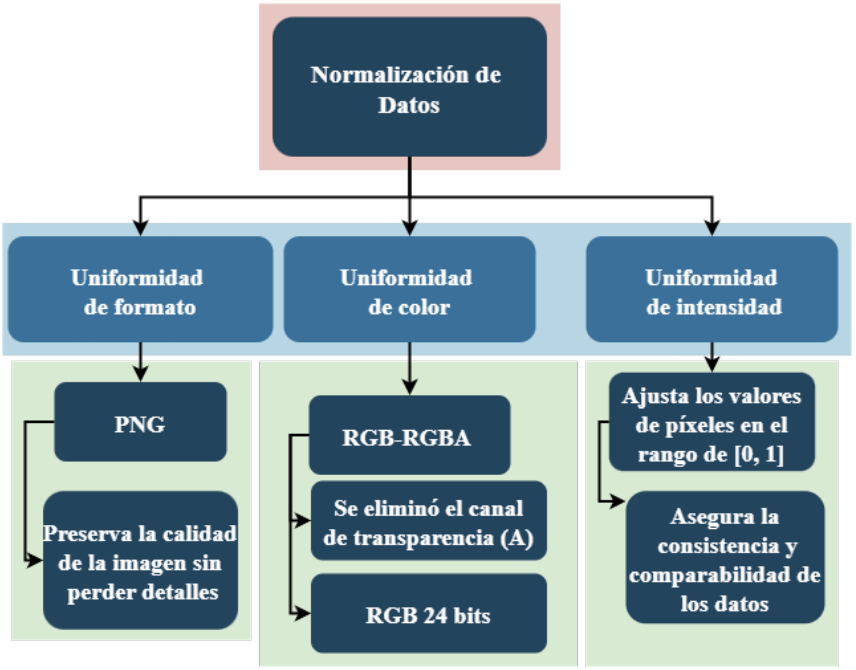



Figura 2.10: Normalización de los datos de entrenamiento

principal entre una imagen en formato PNG (Portable Network Graphics) y una en formato JPG (Joint Photographic Experts Group) radica en el algoritmo de compresión utilizado y en las características de cada formato. A continuación, se mencionan algunas diferencias clave:

- Compresión: El formato PNG utiliza compresión sin pérdida, lo que significa que la imagen se comprime sin sacrificar la calidad. Sin embargo, esto puede resultar en archivos más grandes. Por otro lado, el formato JPG utiliza compresión con pérdida, lo que implica una reducción en la calidad de la imagen para lograr una mayor compresión. Esto se traduce en archivos más pequeños, pero con una pérdida mínima de calidad perceptible.
- Tipo de imágenes: El formato PNG es ideal para imágenes con áreas de colores sólidos o imágenes con bordes nítidos. Por otro lado, el formato JPG es más eficiente en la compresión de imágenes fotográficas y otras imágenes complejas que contienen una amplia gama de colores y detalles sutiles.
- Uso en la web: El formato PNG se utiliza comúnmente para gráficos web, íconos y elementos de diseño que requieren transparencia. También es útil cuando se necesita una calidad visual óptima sin preocuparse por el tamaño del archivo. Por otro lado, el formato JPG se utiliza ampliamente para fotografías en la

web, ya que ofrece una buena relación entre calidad y tamaño de archivo, lo que permite tiempos de carga más rápidos.

Teniendo en cuenta estas diferencias analizadas, se optó por utilizar el formato PNG para todo el conjunto de datos, justificado por el objetivo del proyecto de preservar la calidad de la imagen sin sacrificar detalles para poder identificar los bordes de la herida de manera más precisa.

Además de los formatos, también se encontraron imágenes tanto en formato RGB como en formato RGBA. En una imagen RGB, se utilizan tres canales (rojo, verde y azul) para representar el color en cada píxel. Cada canal almacena la intensidad de ese color utilizando 8 bits, lo que da un total de 24 bits por píxel. Por otro lado, una imagen RGBA es similar a una imagen RGB, pero incluye un canal adicional llamado alfa, que representa la información de transparencia u opacidad de cada píxel. En una imagen RGBA, cada píxel se representa utilizando 32 bits, donde los canales rojo, verde, azul y alfa se almacenan en 8 bits cada uno.

Para unificar las imágenes en un solo formato, se decidió eliminar el canal de transparencia en las imágenes RGBA, obteniendo así un total de 500 imágenes RGB de 24 bits en formato PNG.

Finalmente, para asegurar la consistencia y comparabilidad de los datos, se realiza la normalización de las imágenes. en el caso particular de este proyecto, se aplicó la técnica de normalización de intensidad. Esta técnica ajusta los valores de los píxeles de la imagen para que estén en el rango de [0, 1] en lugar de [0, 255] y se logra restando el valor mínimo de píxel de la imagen y dividiendo por el rango de valores de píxeles (es decir, la diferencia entre el valor máximo y el mínimo). Esta normalización de intensidad es importante porque permite comparar y procesar las imágenes de manera más efectiva. Al normalizar las imágenes, se eliminan las disparidades en la escala de valores de píxeles, lo que ayuda a que el modelo de reconocimiento trabaje de manera consistente en todas las imágenes. Además, esta técnica resalta las características relevantes de las heridas corto-punzantes al ajustar los niveles de intensidad de los píxeles.

Es importante mencionar que la normalización de los datos se realizó después de la etapa de preprocesamiento, que incluyó la redimensión y recorte de las imágenes. Al aplicar la normalización de intensidad a las imágenes ya pre procesadas, se garantizó que todas las imágenes tuvieran una escala de valores coherente y estuvieran listas para el entrenamiento de los modelos de segmentación seleccionados.

6. **Aumento de datos:**

Con el fin de superar las limitaciones en la disponibilidad de imágenes reales y enriquecer el conjunto de entrenamiento para la segmentación semántica de heridas corto-punzantes, se implementaron diversas estrategias de aumento de datos.

Entre estas estrategias se incluyó la generación de heridas artificiales, para simular heridas en tejido humano y capturar la separación de la piel y la presencia de sangre. Estas imágenes artificiales se crearon siguiendo un enfoque que consistió en realizar cortes con un bisturí en láminas de fomi de distintos tonos de color piel, a fin de simular la apariencia de la piel humana. Además, se utilizaron placas de icopor para generar la sensación de profundidad y tejido maltratado. Estas placas se ubicaron debajo de las láminas de fomi y se perforaron a diferentes profundidades, siguiendo el contorno de los cortes en el fomi para recrear la apariencia de sangre humana, se cubrieron los huecos resultantes con una mezcla de colorante rojo para alimentos y miel de abeja, lo cual proporcionó una textura similar a la de la sangre. La imagen 2.11 muestra algunos ejemplos de las formas de herida corto-punzante más comunes, generados artificialmente.

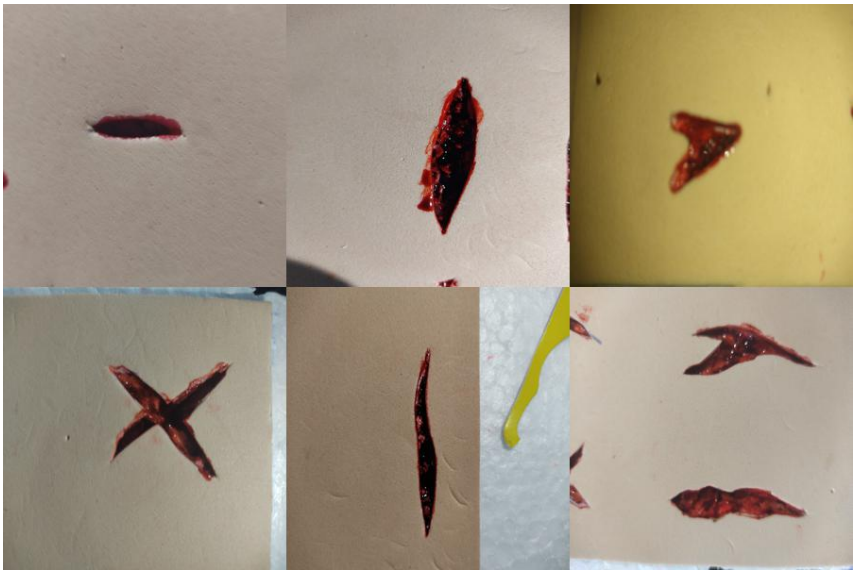


Figura 2.11: Ejemplos de heridas diseñadas artificialmente

Lo anterior permitió obtener 151 imágenes adicionales de heridas corto-punzantes artificiales que se sumaron al conjunto de datos de entrenamiento, elevando el total a 500 imágenes. Aunque este número sigue siendo relativamente bajo en términos de muestras, estas imágenes conforman una base sólida que puede ser ampliada posteriormente mediante técnicas de aumento de datos adicionales.

En resumen, mediante la combinación de imágenes reales y artificiales, se logró enriquecer el conjunto de datos, alcanzando un total de 500 imágenes para el entrenamiento de los modelos U-Net2D, SegNet y DeepLabV3+ que permita la segmentación semántica de heridas corto-punzantes. Aunque este número puede considerarse aún bajo, representa una base inicial sólida que puede ser ampliada mediante otras técnicas de aumento de datos para mejorar el rendimiento de los modelos. Estas técnicas consisten en atribuir transformaciones o modificaciones a las imágenes existentes, generando nuevas muestras realistas y representativas de la variabilidad presente en los datos reales.

Para aumentar la variabilidad del conjunto de entrenamiento y evitar el sobre ajuste, así como mejorar la capacidad de los modelos para generalizar a nuevos casos, se aplican distintas transformaciones a las imágenes del conjunto base recopilado, entre las cuales se encuentran cambios en brillo, tonalidad y contraste, así como volteo horizontal y vertical, rotación y transposición.

La imagen 2.12 sintetiza el desarrollo del aumento de datos, además, se describen a continuación cada una de sus etapas:

- Importación de bibliotecas: Se importan las bibliotecas necesarias para el procesamiento de imágenes, manipulación de archivos y operaciones matemáticas. En este caso se emplearon las bibliotecas TensorFlow, OS, math y PIL.
- Definición de variables: Se definen las variables necesarias, como el número de imágenes a generar mediante aumento de datos, las rutas de las carpetas de entrada y salida de imágenes y etiquetas. Con el fin de incrementar significativamente la cantidad de datos de entrenamiento para los modelos, se estableció que se generarían 6 imágenes adicionales por cada imagen del conjunto de entrenamiento base, el cual cuenta con 500 imágenes. De esta manera, el nuevo conjunto final está conformado por 3000 imágenes.
- Definición de transformaciones: Se crea una función que aplica una serie de transformaciones aleatorias a cada imagen y etiqueta, asegurando que las mismas transformaciones se apliquen tanto a la imagen en color como a su máscara semántica, manteniendo la consistencia en los datos. Estas transformaciones incluyen cambios en brillo, tonalidad, contraste, volteo horizontal y vertical, transposición y rotación.
- Procesamiento de archivos de imágenes y etiquetas: Se itera sobre los archivos de imágenes y etiquetas en las carpetas de entrada. Cada imagen y etiqueta se

carga utilizando la biblioteca PIL.

- Generación de imágenes aumentadas: Dentro del bucle principal, se generan $n$ imágenes aumentadas para cada imagen y etiqueta. Se copian las imágenes y etiquetas originales, se convierten en tensores y se pasan a la función encargada de aplicar las transformaciones aleatorias definidas. A continuación, se vuelven a convertir en imágenes, se binariza la etiqueta y se guardan las imágenes y etiquetas aumentadas en las carpetas de salida.


Figura 2.12: Etapas del proceso de aumento de datos

El diagrama de flujo mostrado en la figura 2.13, describe el proceso para aumentar el conjunto de datos de imágenes y etiquetas. El proceso comienza cargando cada imagen y etiqueta de las carpetas de entrada. Luego, para cada imagen y etiqueta, se realizan $n$ copias y se aplican transformaciones aleatorias a cada copia para crear imágenes y etiquetas aumentadas. Estas imágenes y etiquetas aumentadas se convierten en objetos Image de Pillow, se binarizan las etiquetas aumentadas y se guardan en las carpetas de salida.

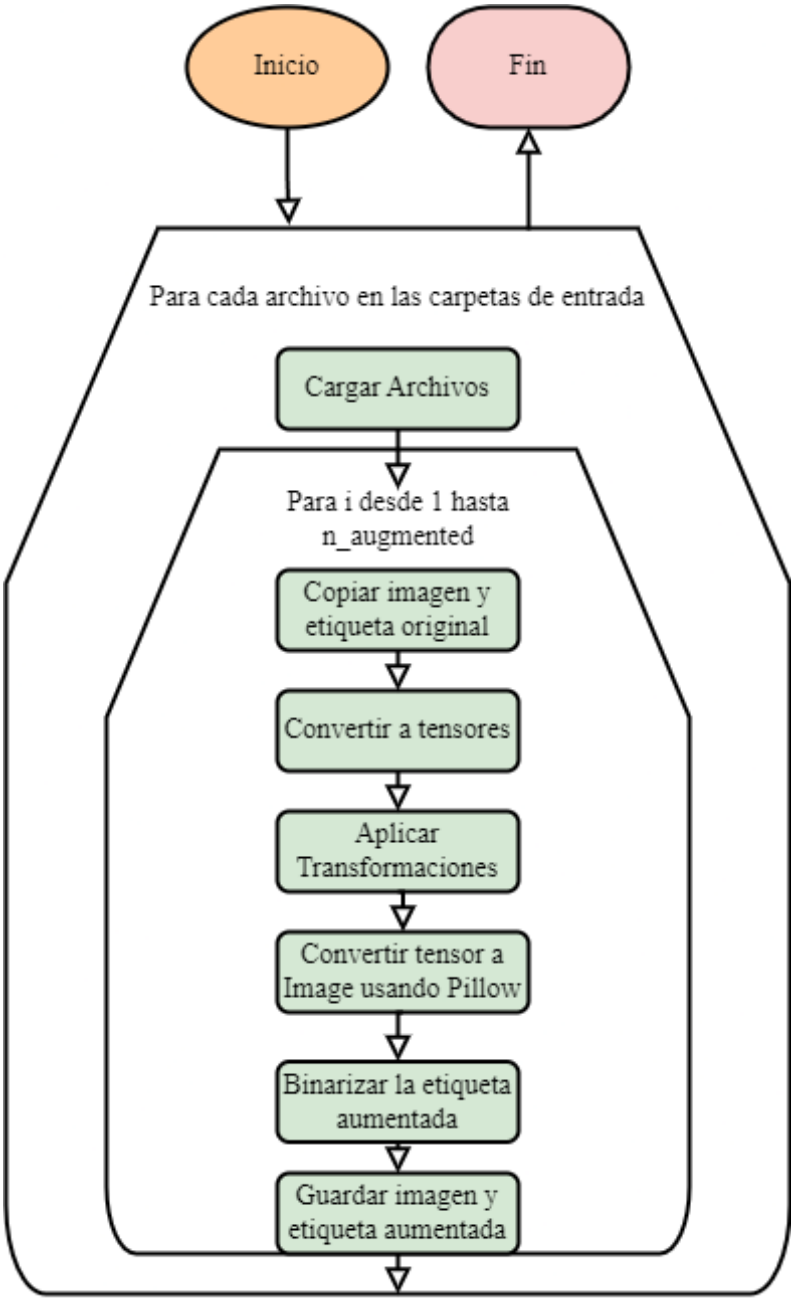



Figura 2.13: Diagrama de flujo del proceso de aumento de datos

**7. Generación de lotes de datos:**

Para entrenar eficientemente los modelos, se utiliza el concepto de lotes, donde los datos se agrupan en conjuntos de tamaño fijo que contienen tanto las imágenes de entrada como sus respectivas anotaciones. Esta estrategia permite procesar múltiples ejemplos de forma simultánea, acelerando así el proceso de entrenamiento. El tamaño

del lote determina la cantidad de ejemplos utilizados antes de ajustar los pesos de la red. Es importante encontrar un equilibrio entre el tamaño del lote, la precisión del gradiente y la disponibilidad de memoria.

Un tamaño de lote más grande puede proporcionar una estimación más precisa del gradiente, lo que a su vez acelera el proceso de entrenamiento. Sin embargo, es necesario considerar las limitaciones de memoria, especialmente cuando se utiliza el entorno de Google Colab. Si el tamaño del lote excede la capacidad de memoria asignada por Colab, es probable que se produzcan errores de memoria insuficiente durante el entrenamiento.

Por otro lado, un tamaño de lote más pequeño puede resultar en una estimación de gradiente más ruidosa, pero puede ser beneficioso en situaciones donde se manejan conjuntos de datos grandes o como es el caso, cuando los recursos computacionales son limitados.

Teniendo en cuenta estas consideraciones y las limitaciones de memoria en el entorno de Google Colab, se opta probar tamaños de lote moderados entre 16 y 32. Esto permitirá un equilibrio entre la precisión del gradiente y la capacidad de memoria disponible. Es importante realizar pruebas y ajustes adicionales para encontrar el tamaño de lote óptimo en función de los recursos específicos y las necesidades de segmentación.

En conclusión, estas actividades de obtención y preparación de datos analizadas son esenciales para garantizar la calidad y consistencia de los datos utilizados en el entrenamiento de los modelos y cumplir el objetivo del proyecto, enmarcado en la segmentación semántica de imágenes biomédicas.

### 2.3.2. Etapa 2: Compilación del modelo:

La etapa de compilación de los modelos U-Net2D, SegNet y DeepLabV3+ es crucial para la implementación efectiva de la segmentación semántica de heridas en imágenes médicas. En esta etapa, se configuran y se preparan los modelos para el entrenamiento. Esto implica parametrizar la arquitectura definida de cada modelo, que para el caso de estudio, los modelos reciben imágenes de la forma: $((n\_filters = 16, input\_dim\_x = 224, input\_dim\_y = 224, num\_channels = 3))$. Además, es necesario seleccionar una función de pérdida adecuada, un optimizador y configurar otras opciones de entrenamiento

como las métricas de evaluación y los hiperparámetros. Teniendo en cuenta la teoría descrita en la Sección 1.3 (página 16), se dispone de material suficiente para realizar elección de parámetros convenientes y asociados a la etapa de Compilación de los modelos.

A continuación, se describen en detalle las actividades involucradas en esta etapa:

1. **Selección de la función de pérdida:**

   La elección de la función de pérdida apropiada depende en primer lugar del propósito del sistema entrenado. Para la tarea de segmentación semántica en imágenes vista como un problema de clasificación a nivel de píxeles, algunas funciones de pérdida comúnmente usadas son la entropía cruzada, el coeficiente Dice o la distancia de Hausdorff, como se conceptualiza en la Sección 1.3 (página 39). Estas funciones permiten comparar la segmentación predicha con las anotaciones de referencia. Sin embargo, son mayormente usadas en tareas de segmentación multiclase, donde se pretende identificar varios objetos en las imágenes.

   En el contexto de esta investigación, se pretende segmentar correctamente una única clase denominada herida suturable, convirtiendo la problemática en una tarea de segmentación semántica binaria, donde la máscara de segmentación predicha por los modelos representa los píxeles clasificados en la clase definida como herida. En este sentido, se seleccionó la función de pérdida entropía cruzada binaria (binary_crossentropy). Esta función de pérdida se utilizará durante el entrenamiento para calcular la discrepancia entre las máscaras de segmentación predichas por cada modelo y las máscaras de segmentación reales etiquetadas por los expertos.

2. **Selección del optimizador:**

   Seleccionar el optimizador es una etapa crucial en la compilación y posterior entrenamiento de los modelos de segmentación semántica binaria elegidos. pues puede tener un impacto significativo en la rapidez con la que cada modelo converge a la solución óptima, así como en la calidad de la predicción obtenida. Estos algoritmos son los métodos encargados de ajustar los parámetros de cada red con el objetivo de minimizar la función de pérdida elegida, basándose en las características de los datos de entrenamiento. Como se conceptualiza en la sección 1.3 (página 21), entre los optimizadores más comúnmente utilizados se encuentran el Descenso de Gradiente Estocástico (SGD) y Adam.

   En el contexto del proyecto, se ha seleccionado Adam como el optimizador debido a su eficiencia y rendimiento superior en comparación con otros optimizadores. Con esta elección, es necesario definir una tasa de aprendizaje (Learning Rate) que

hace parte de los hiperparámetros que recibe el optimizador y puede ser modificado iterativamente con base en la conformidad con el desempeño de los modelos de segmentación durante el entrenamiento.

3. **Selección de las métricas de evaluación:**

   Para evaluar de manera cuantitativa la calidad y el rendimiento de los modelos durante el entrenamiento, es necesario definir métricas que permitan determinar la precisión de cada modelo al identificar y clasificar correctamente los píxeles en las imágenes de entrada.

   Como se menciona en la página 38, algunas de las métricas más comúnmente empleadas en la segmentación semántica de imágenes incluyen la precisión, sensibilidad (recall), el coeficiente de Dice y el índice de Jaccard (también conocido como Intersección sobre Unión o IoU).

   En el marco investigativo, se seleccionaron las métricas mencionadas para evaluar el rendimiento de cada uno de los modelos durante el entrenamiento, ayudando así a prevenir el sobre ajuste. Estas métricas también se utilizarán para comparar el desempeño en la segmentación de imágenes de los tres modelos estudiados.

4. **Configuración de hiperparámetros:**

   Los hiperparámetros son variables definidas antes del proceso de entrenamiento y que influyen en la velocidad y calidad de este. Estos incluyen la tasa de aprendizaje, el tamaño del lote, el número de épocas, entre otros, y pueden afectar significativamente el proceso de optimización durante el entrenamiento de los modelos. Para el entrenamiento de los tres modelos en estudio se emplearon los hiperparámetros listados en la tabla 2.4

| **Learning Rate** | **Batch Size** | **Epochs** | **Loss** |
|---|---|---|---|
| $1e-4$ | 16 | 60 | $binary\_crossentropy$ |

Tabla 2.4: Hiperparámetros usados para el entrenamiento de los modelos

A continuación, se listan las implicaciones de los hiperparámetros seleccionados, en el proceso iterativo de optimización de los modelos durante el entrenamiento de los mismos:

- **Learning Rate:** Una tasa de aprendizaje alta puede hacer que el modelo converja rápidamente a una respuesta, pero también puede causar que se salte el mínimo global y se quede atascado en un mínimo local. Por otro lado, una tasa

de aprendizaje baja puede permitir al modelo alcanzar el mínimo global, pero puede requerir más tiempo para converger. Para el entrenamiento de los tres modelos, inicialmente se definió como $1e-3$ que fue modificada posteriormente a $1e-4$ para lograr un buen equilibrio entre precisión y velocidad durante el entrenamiento.

- **Bach Size:** El tamaño del lote puede afectar la estabilidad y la velocidad del entrenamiento. Un lote grande puede proporcionar una estimación más precisa del gradiente, pero también puede llevar a un entrenamiento más lento y a un uso intensivo de los recursos computacionales. En el marco de la investigación, se inició con un tamaño de lote de 16, que se aumentó hasta un máximo 32 imágenes por lote, sin desbordar los recursos computacionales asignados en la sesión del entorno de Google Colab. Esto incurre en un buen equilibrio entre precisión y velocidad del entrenamiento.
- **Épocas:** El número de épocas determina cuántas veces el algoritmo de aprendizaje trabajará en todo el conjunto de datos de entrenamiento. Un número insuficiente de épocas puede resultar en un modelo sub ajustado, mientras que demasiadas épocas pueden llevar a un sobre ajuste. En el contexto de estudio, inicialmente se probó un número de 100 épocas, pero se observó que los modelos convergían a una respuesta en aproximadamente la mitad de ellas, lo cual incurría en un sobre ajuste de los modelos, lo cual motivó a reducir el número de épocas de entrenamiento a solo 60.

Es preciso señalar que la selección de hiperparámetros es un proceso que requiere experimentación. Es necesario ajustar y probar diferentes combinaciones de hiperparámetros hasta estar conforme con el desempeño de cada modelo. A menudo, este proceso puede implicar un equilibrio entre la precisión de los modelos y el tiempo de entrenamiento.

Una vez completadas todas las actividades en la etapa de diseño y configuración de los modelos U-Net2D, SegNet y DeepLabV3+, así como su compilación, se tiene una arquitectura definida y todos los parámetros necesarios para comenzar la etapa de entrenamiento. Cabe destacar que el entrenamiento y la configuración de los modelos pueden requerir experimentación y ajustes iterativos para optimizar el rendimiento y la capacidad de generalización de cada modelo.

### 2.3.3. Etapa 3: Entrenamiento del modelo:

La etapa de entrenamiento de los modelos U-Net2D, SegNet y DeepLabV3+ es el corazón del proceso de implementación de la segmentación semántica de heridas corto-punzantes en imágenes médicas. El entrenamiento es un proceso iterativo que requiere experimentación y ajuste fino de sus hiperparámetros. A medida que los modelos se entrenan a lo largo de varias épocas, aprenden a identificar y clasificar mejor las características relevantes en las imágenes de entrada.

En esta etapa, los modelos ya compilados aprenden características a partir de los datos de entrenamiento. Esto implica alimentar la red con imágenes y sus correspondientes máscaras de segmentación, y permitir que el optimizador ajuste sus parámetros para minimizar la función de pérdida elegida. Además, se monitorizan las métricas de evaluación seleccionadas, para evaluar el rendimiento de cada modelo, tanto en el conjunto de datos de entrenamiento, como en el de validación y así prevenir el sobre ajuste de los mismos. A continuación se describe el proceso de creación de la función diseñada como generador de datos:

1. **Generador de datos:**

   Al trabajar con grandes conjuntos de datos de imágenes de alta resolución puede ser una tarea desafiante en términos de eficiencia y gestión de recursos. En este contexto, un generador de datos se convierte en una herramienta esencial. Este componente dinámico y eficiente permite cargar, normalizar y proporcionar imágenes y etiquetas por lotes durante la etapa de entrenamiento de los modelos, optimizando el uso de la memoria y acelerando el mismo. El generador de datos es, por tanto, una piedra angular en la construcción de modelos de segmentación de imágenes precisos y efectivos, aliviando la carga de computacional y facilitando la manipulación de datos en el contexto de visión artificial.

   La figura 2.14 muestra la representación gráfica del diagrama de flujo, asociado a la función diseñada como generador de datos.

   En contexto, como se puede observar en la figura 2.14, el flujo arranca con la inicialización del generador de datos, el cual recibe los siguientes parámetros: path, split_ratio, x, y, color_space. Los cuales representan el directorio donde se encuentra el conjunto de datos, la tasa para dividir los datos (70 % entrenamiento, 30 %validación), las dimensiones (x, y) de las imágenes y el espacio de color de las mismas. Cabe destacar que el generador espera que el directorio proporcionado esté

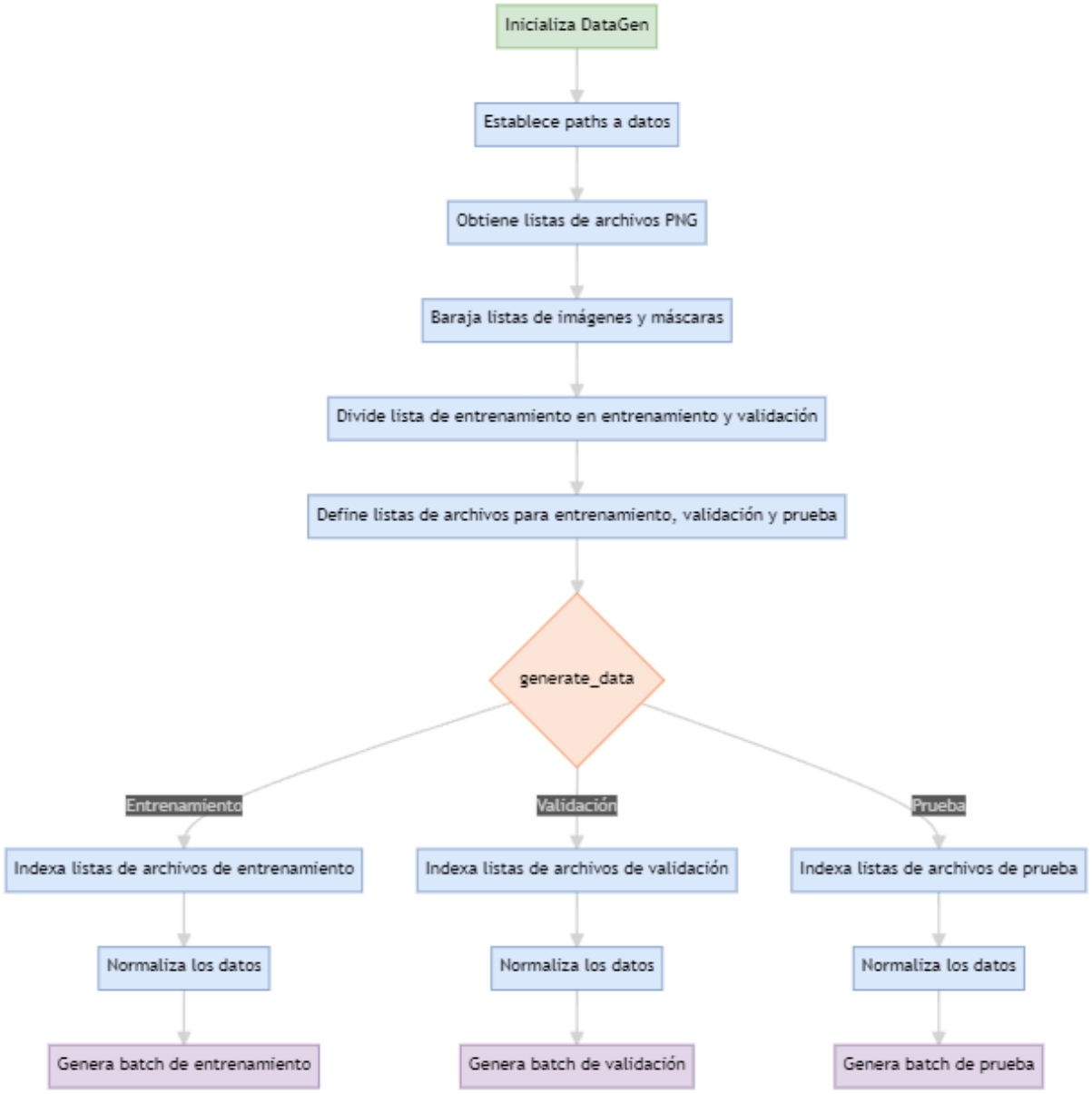



Figura 2.14: Diagrama de flujo del generador de datos usado

correctamente separado en carpetas, `train, test` que contienen los datos, `images, labels` respectivamente. Para el caso de investigación, los parámetros utilizados se observan en la tabla 2.5.

| **split_ratio** | **x, y** | **color_space** |
|---|---|---|
| 0,3 | 224, 224 | RGB |

Tabla 2.5: Parámetros usados para el generador de datos

Finalmente, en la decisión `generate_data`, se generan lotes de datos para el entrenamiento, validación o prueba en función de cuál de las etapas esté activa. Para cada lote, se lee una imagen y su etiqueta correspondiente del disco, se verifica que tengan las dimensiones correctas, se normaliza su intensidad de píxeles en el rango (0-1) y finalmente se añaden al lote. Este proceso se repite hasta que se han generado suficientes lotes.

**2. Otros parámetros del entrenamiento:**

Además de lo anterior, para la etapa de entrenamiento de los modelos seleccionados, se debe tener en cuenta los parámetros que recibe la función model.fit() de Keras, los cuales son: el conjunto de datos de entrenamiento y el conjunto de datos de validación (proporcionados por el generador de datos), así como el número de épocas definido anteriormente dentro de los hiperparámetros. Adicionalmente, la función en cuestión recibe otro parámetro llamado callbacks, que contiene funciones adicionales que pueden realizar acciones en diferentes etapas del entrenamiento y son útiles para obtener visibilidad del estado interno y las estadísticas de cada modelo durante el entrenamiento.

Como se discute en la sección 1.3 (página 14) respecto a los callbacks, durante la etapa de entrenamiento de los modelos seleccionados, se usaron los siguientes:

- **ModelCheckpoint:** Este callback guarda el mejor modelo después de cada época. Donde sus parámetros `model_path` es la ubicación donde se guardará el modelo y `save_best_only=True` indica que solo se guardará el modelo con la mejor precisión en el conjunto de validación.
- **ReduceLROnPlateau:** Este callback reduce la tasa de aprendizaje (learning rate) cuando una métrica ha dejado de mejorar. En nuestro caso, la métrica es (Función de pérdida en el conjunto de validación). El factor por el cual se reduce la tasa de aprendizaje es 0.1. Si no hay mejora en la métrica después de 3 épocas (`patience=3`), la tasa de aprendizaje se reduce en el factor escogido.

- **EarlyStopping:** Este callback detiene el entrenamiento cuando una métrica ha dejado de mejorar. En este caso, la métrica seleccionada es la función de pérdida en el conjunto de validación (`val_loss`). Si no hay mejora en la métrica después de 5 épocas(`patience=5`), el entrenamiento se detiene.
- **CSVLogger:** Este callback es usado para guardar el registro de las métricas de evaluación durante el entrenamiento y validación del modelo en un archivo `.csv`. Dicho archivo permite graficar las métricas y visualizar el desempeño de los modelos.

## 2.4. Validación de los modelos

La validación es una parte esencial del flujo de trabajo en el marco de aprendizaje automático, pues permite evaluar el funcionamiento de cada modelo, ajustar los hiperparámetros para un mejor rendimiento y evitar el sobre ajuste. En esta sección, se pone un énfasis particular en la validación del entrenamiento de los modelos U-Net2D, SegNet y DeepLabV3+ con ResNet50 como encoder. Inicialmente, se propuso un enfoque de validación cruzada (sección 1.3, pág. 16) debido a la limitada cantidad de datos de entrenamiento disponibles (3000 imágenes y etiquetas). La idea inicial consistió en dividir el conjunto de datos en 5 pliegues, lo que implicaría entrenar los modelos 5 veces. Sin embargo, este enfoque de validación cruzada no resultó ser viable debido a las limitaciones de los recursos de GPU y CPU en la sesión de Google Colab como se observa en la figura 2.15.

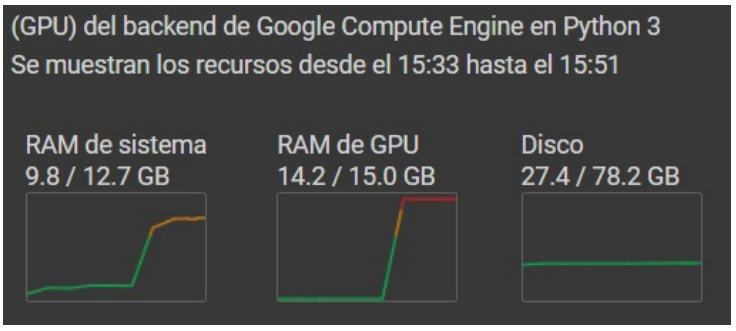



Figura 2.15: Recursos computacionales desbordados en la sesión de Colab

Por lo tanto, se optó por un enfoque de validación con detención temprana, como se analiza en la sección 1.3 pág. 16. Se realizó una única división de los datos (70 % para entrenamiento y 30 % para validación). Además, se guardaron puntos de control de los modelos con el mejor rendimiento de validación y se utilizó una detención temprana para interrumpir el entrenamiento cuando la función de pérdida de validación deje de mejorar.

A través de este proceso, se pudo validar eficazmente el rendimiento de los modelos. Esta sección muestra visualizaciones detalladas de las características aprendidas por los modelos, gráficas de las métricas de evaluación que facilitan una mejor comprensión de los resultados obtenidos y visualizaciones de las máscaras de segmentación generadas por los modelos. Además, se discutirán conclusiones preliminares sobre el rendimiento de los modelos, destacando que DeepLabV3+ ha demostrado ser superior en la tarea de segmentación semántica de heridas corto punzantes.

### 2.4.1. Visualización de las características aprendidas

**Métricas de validación:**

Durante el proceso de validación, se utilizaron métricas específicas para evaluar el rendimiento de cada modelo en el conjunto de datos de validación predispuesto. La elección de las métricas de validación depende del tipo de problema que se esté resolviendo. Para el contexto del proyecto, entendido como un problema de clasificación binaria, las métricas calculadas incluyen: exactitud, precisión, IoU (Intersección sobre Unión o jaccard) y el Coeficiente Dice.

En primer lugar, se entrenaron los modelos U-Net2D y SegNet sin un conocimiento profundo sobre la implicación de los hiperparámetros en el desempeño de los modelos. Esto permitió obtener una primera versión entrenada de los modelos, que aunque su desempeño fue bajo, proporcionó un primer acercamiento a las tendencias de los modelos y una idea general del número de épocas en las cuales convergían a una respuesta. La tabla 2.6 muestra las métricas promedio de entrenamiento para esta versión preliminar de cada modelo. De igual forma, la tabla 2.7 muestra las métricas promedio de validación para esta versión preliminar.

| **Modelo** | **Pérdida** | **Dice** | **Jaccard** | **Precisión** | **Recall** |
|---|---|---|---|---|---|
| U-Net2D | 7.36 % | 64.79 % | 82.72 % | 92.64 % | 92.71 % |
| SegNet | 9.7 % | 68.4 % | 58.51 % | 77.36 % | 72.11 % |

Tabla 2.6: Métricas de entrenamiento (versión preliminar)

| **Modelo** | **Pérdida** | **Dice** | **Jaccard** | **Precisión** | **Recall** |
|---|---|---|---|---|---|
| U-Net2D | 9.03 % | 62.03 % | 76.33 % | 89.46 % | 87.84 % |
| SegNet | 9.85 % | 68.45 % | 59.09 % | 79.33 % | 70.02 % |

Tabla 2.7: Métricas de validación (versión preliminar)

Así mismo, las gráficas 2.17 y 2.19 muestran las métricas de entrenamiento y validación de los modelos U-Net2D y SegNet respectivamente. Se puede observar que todas las métricas muestran un comportamiento consistente a lo largo de las épocas, siendo SegNet mucho más estable que su contraparte. Esto podría indicar que los modelos están aprendiendo. Sin embargo, como se observa en las figuras 2.16 y 2.18, la pérdida de validación, representada por la línea naranja, deja de disminuir y empieza a aumentar alrededor de la época 50. Mientras que la pérdida de entrenamiento, representada por la línea azul, sigue disminuyendo. Esto podría ser una señal de sobre ajuste en los modelos y sugiere la necesidad de modificar sus hiperparámetros.

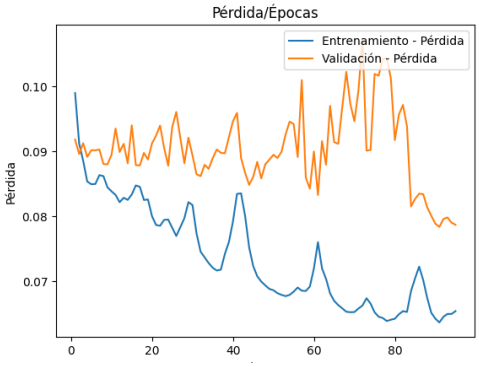



Figura 2.16: Pérdidas U-Net2D v1

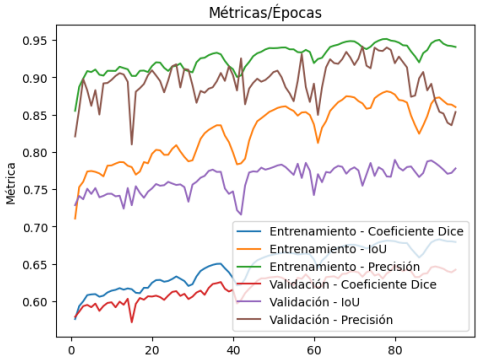



Figura 2.17: Métricas U-Net2D v1

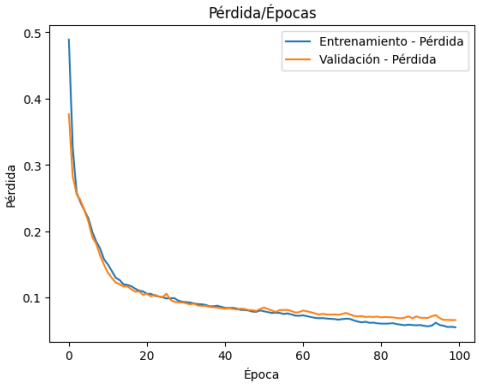



Figura 2.18: Pérdidas SegNet v1

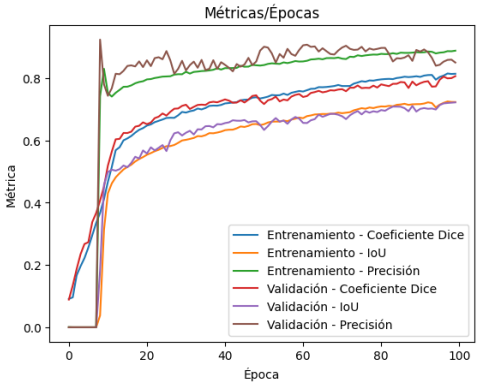



Figura 2.19: Métricas SegNet v1

El ajuste iterativo de hiperparámetros como la tasa de aprendizaje, el tamaño del lote y el número de épocas pueden tener un impacto significativo en el rendimiento de los modelos. Teniendo esto en cuenta, se generaron tres versiones de los modelos

U-Net2D y SegNet, con diferente combinación de hiperparámetros. La tabla 2.8 y la tabla 2.9 muestran las métricas promedio de entrenamiento y validación de cada versión generada.

| **Versión** | **Modelo** | **Pérdida** | **Dice** | **IoU** | **Precisión** | **Recall** |
|---|---|---|---|---|---|---|
| 1 | U-Net2D | 8.75 % | 65.92 % | 69.62 % | 84.21 % | 80.06 % |
| 2 | U-Net2D | 9.93 % | 59.12 % | 67.53 % | 81.65 % | 77.57 % |
| 3 | U-Net2D | 18.76 % | 52.98 % | 46.02 % | 69.09 % | 55.99 % |
| 1 | SegNet | 8.94 % | 71.3 % | 62.72 % | 82.69 % | 76.36 % |
| 2 | SegNet | 9.94 % | 67.31 % | 58.23 % | 77.96 % | 71.55 % |
| 3 | SegNet | 17.63 % | 47.99 % | 35.95 % | 65.35 % | 47.73 % |

Tabla 2.8: Promedio de métricas de entrenamiento obtenidas

| **Versión** | **Modelo** | **Pérdida** | **Dice** | **IoU** | **Precisión** | **Recall** |
|---|---|---|---|---|---|---|
| 1 | U-Net2D | 9.68 % | 64.64 % | 67.89 % | 83.35 % | 76.73 % |
| 2 | U-Net2D | 9.54 % | 58.78 % | 65.99 % | 80.67 % | 76.62 % |
| 3 | U-Net2D | 18.86 % | 51.95 % | 45.21 % | 69.33 % | 54.64 % |
| 1 | SegNet | 9.41 % | 70.38 % | 62.6 % | 81.55 % | 75.72 % |
| 2 | SegNet | 10.22 % | 65.49 % | 57.32 % | 79.17 % | 68.85 % |
| 3 | SegNet | 16.97 % | 48.02 % | 37.24 % | 62.26 % | 50.01 % |

Tabla 2.9: Promedio de métricas de validación obtenidas

En la figura 2.20, las curvas de color azul, que representan la pérdida de entrenamiento, comienzan con un valor alto y disminuyen rápidamente. Esto es una buena señal, ya que indica que los modelos U-Net2D y SegNet están aprendiendo características relevantes y mejorando a medida que se entrenan. Las líneas naranjas, que representan la pérdida de validación, comienzan con un valor más bajo y disminuyen más gradualmente. Esto también es positivo, ya que podría indicar que el modelo está mejorando su rendimiento en los datos de validación.

Sin embargo, es importante tener en cuenta que si la pérdida de validación deja de disminuir o empieza a aumentar, mientras la pérdida de entrenamiento sigue disminuyendo, esto podría ser una señal de sobre ajuste lo que implica el reajuste de hiperparámetros para optimizar su rendimiento. Además, se puede observar en la gráfica de pérdidas por época 2.20 que la curva de validación (color naranja) inicialmente está por debajo de la curva de entrenamiento (color azul), esto podría

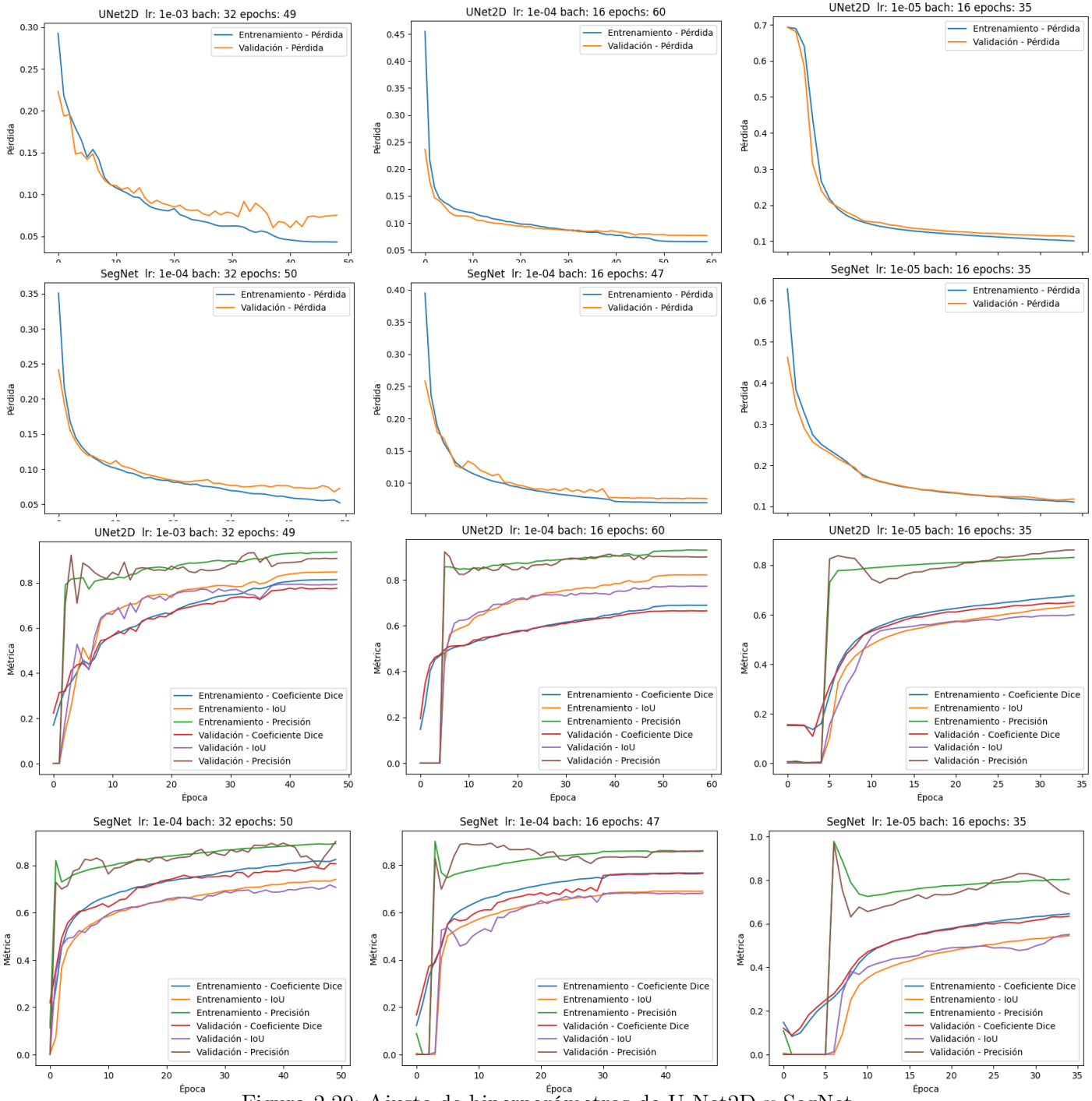



Figura 2.20: Ajuste de hiperparámetros de U-Net2D y SegNet

indicar que los modelos están desempeñándose mejor en el conjunto de validación que en el conjunto de entrenamiento lo cual es algo inusual, ya que normalmente esperaríamos que el modelo tenga un rendimiento mejor o similar en el conjunto de entrenamiento. No obstante, esto podría suceder en algunas circunstancias, por ejemplo, si el conjunto de validación es más fácil de predecir que el conjunto de entrenamiento. Aun así, este tipo de comportamiento podría ser un indicio de que algo inusual está sucediendo con el proceso de entrenamiento de los modelos U-Net2D y SegNet o con la división de los datos, podría ser útil investigar más a fondo para entender por qué está ocurriendo.

De igual manera, las métricas de entrenamiento y validación Dice, IoU, y precisión, como se observan en la figura 2.20 para los modelos U-Net2D y SegNet, muestran una tendencia general de aumento superior al 60 % a lo largo del tiempo, lo que indica que el modelo está aprendiendo y mejorando a medida que se itera en las épocas de entrenamiento. A su vez, también se observan algunas fluctuaciones en las métricas, lo cual es normal durante el entrenamiento de un modelo de aprendizaje profundo, ya que el modelo está tratando de optimizar una función de pérdida compleja en un espacio de alta dimensión (millones de parámetros de cada modelo).

Además de las métricas de validación utilizadas para el entrenamiento de cada versión, se optó por incluir variables como el tiempo total de entrenamiento y el promedio por cada época, como se observa en la tabla 2.10.

| **Versión** | **Modelo** | **Por época (s)** | **Entrenamiento (s)** |
|---|---|---|---|
| 1 | U-Net2D | 33.59 | 1645.84 |
| 2 | U-Net2D | 19.14 | 1148.25 |
| 3 | U-Net2D | 19.06 | 667.15 |
| 1 | SegNet | 13.59 | 679.36 |
| 2 | SegNet | 11.87 | 557.69 |
| 3 | SegNet | 12.44 | 435.41 |

Tabla 2.10: Tiempo de entrenamiento promedio

Como se puede apreciar en la figura 2.21, los modelos U-Net2D y SegNet pueden ser entrenados en un promedio de entre 10 y 30 minutos según la configuración de hiperparámetros elegida. Esta eficiencia se debe en gran medida a los recursos computacionales, como las GPU y las TPU, proporcionados por Google en la sesión de Colab. Sin embargo, otro factor clave que contribuye a esta rapidez es la detección temprana del entrenamiento. De no ser por esta detección, cada versión de los

modelos requeriría más de cinco horas para completar su entrenamiento y estarían más propensos al sobre ajuste.

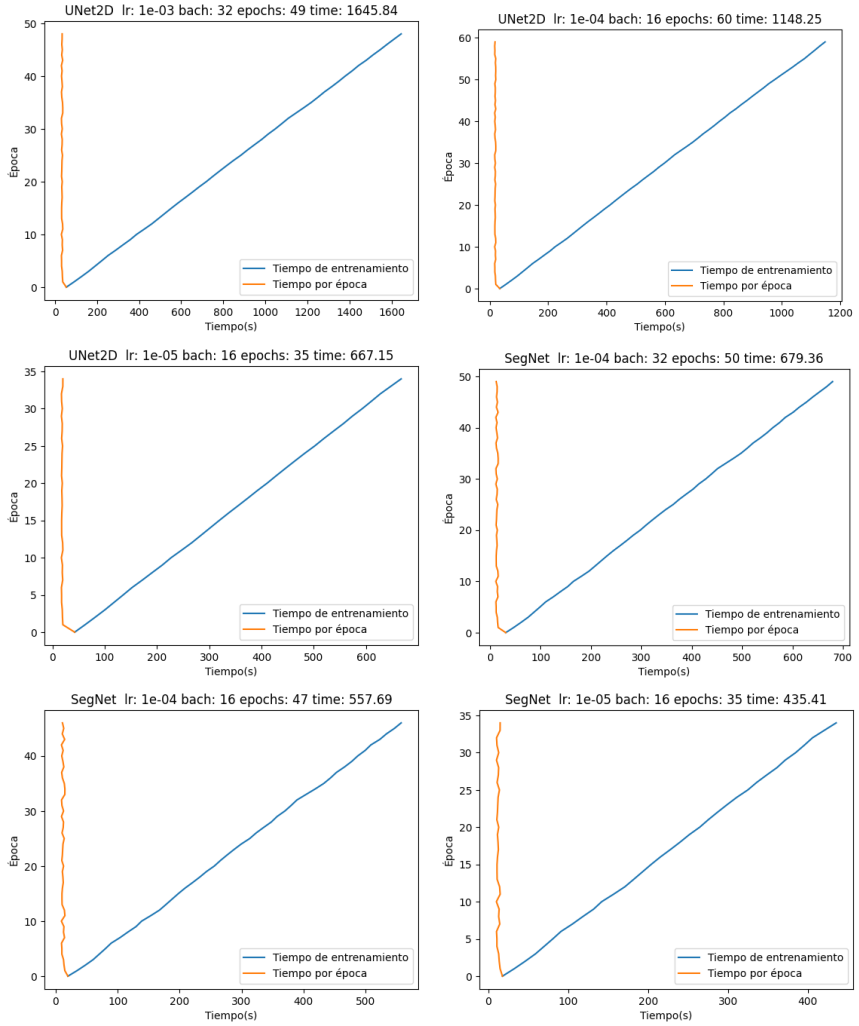



Figura 2.21: Tiempos de entrenamiento para U-Net2D y SegNet

Por otro lado, y en comparación con las versiones anteriores de los modelos U-Net2D y SegNet, el modelo DeepLabV3+ con ResNet50 como encoder, como se puede observar en la figura 2.22 parece tener un rendimiento más estable y consistente. Las métricas de entrenamiento y validación Dice, IoU, y precisión muestran una tendencia general de aumento superior al 80 % a lo largo del tiempo. De igual manera, las gráficas de pérdida por época no muestran signos de sobre ajuste, lo que sugiere que el modelo está aprendiendo características generalizables en lugar de memorizar los datos de entrenamiento. La tabla 2.11 y la tabla 2.12 resumen las métricas de entrenamiento y validación obtenidas respectivamente para las dos versiones generadas

del modelo DeepLabV3+.

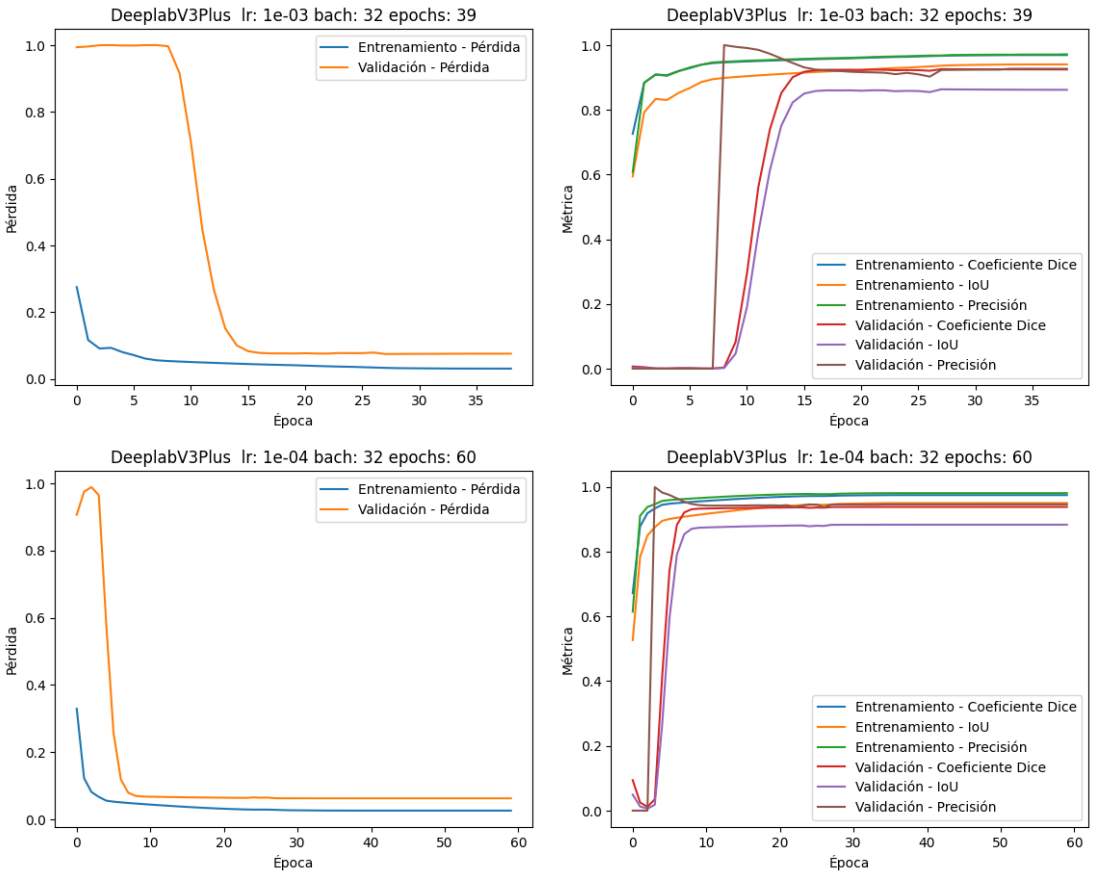



Figura 2.22: Métricas de evaluación del modelo DeepLabV3+

| Versión | Modelo | Pérdida | Dice | IoU | Precisión | Recall |
|---|---|---|---|---|---|---|
| 1 | DeepLabV3+ | 5.22 % | 94.78 % | 90.41 % | 94.62 % | 95.47 % |
| 2 | DeepLabV3+ | 3.9 % | 96.1 % | 92.78 % | 96.79 % | 97.39 % |

Tabla 2.11: Promedio de métricas de entrenamiento obtenidas

| Versión | Modelo | Pérdida | Dice | IoU | Precisión | Recall |
|---|---|---|---|---|---|---|
| 1 | DeepLabV3+ | 34.35 % | 65.69 % | 60.25 % | 74.23 % | 65.94 % |
| 2 | DeepLabV3+ | 13.62 % | 86.37 % | 80.66 % | 89.96 % | 85.91 % |

Tabla 2.12: Promedio de métricas de validación obtenidas

Siguiendo un procedimiento análogo a los modelos anteriores, como se muestra en la tabla 2.13, se determinó el tiempo total de entrenamiento y el promedio por cada época para el modelo DeepLabV3+ con ResNet50 como encoder. Como se puede apreciar, este modelo

requiere un tiempo de entrenamiento que varía entre 40 y 60 minutos, dependiendo de la configuración de hiperparámetros seleccionada. En comparación con los dos modelos anteriores, este modelo demanda aproximadamente el doble o el triple del tiempo de entrenamiento, lo cual es comprensible debido a su mayor número de parámetros y capas, lo que aumenta significativamente la complejidad del modelo. Sin embargo, gracias a los recursos computacionales proporcionados por Google Colab, se pudo obtener una buena precisión en la segmentación, en un tiempo de entrenamiento razonable.

| **Versión** | **Modelo** | **Por época (s)** | **Entrenamiento (s)** |
|---|---|---|---|
| 1 | DeepLabV3+ | 61.83 | 2411.43 |
| 2 | DeepLabV3+ | 60.59 | 3635.48 |

Tabla 2.13: Tiempo de entrenamiento promedio

Los resultados del entrenamiento de los modelos estudiados, proporcionan una visión valiosa sobre las capacidades de los modelos de aprendizaje automático para la segmentación y reconocimiento de heridas corto punzantes. La evidencia de que los modelos U-Net2D y SegNet están aprendiendo características relevantes y mejorando durante el entrenamiento es alentadora y sugiere que estos algoritmos tienen el potencial de desempeñar un papel fundamental en aplicaciones médicas de diagnóstico y seguimiento. Además, la destacada estabilidad y generalización observadas en el modelo DeepLabV3+ con ResNet50 como codificador son indicativas de un rendimiento prometedor. Estos hallazgos están estrechamente relacionados con la pregunta de investigación inicial sobre las características deseadas de un algoritmo de aprendizaje de máquina para la segmentación y reconocimiento de heridas corto punzantes, ya que subrayan la importancia de la capacidad de aprendizaje y generalización de los modelos en este contexto médico. Si bien estos resultados son prometedores, también indican la necesidad de una mayor investigación y validación para garantizar la confiabilidad y la efectividad de estos algoritmos en entornos clínicos del mundo real.

### 2.4.2. Visualización de las máscaras de segmentación

Adicionalmente, para validar el rendimiento de los modelos U-Net2D, SegNet y DeepLabV3+ en la tarea de segmentación semántica de heridas corto punzantes, se realizaron pruebas de inferencia con cada modelo utilizando el conjunto de datos original (500 imágenes y sus correspondientes etiquetas) antes del aumento de datos. Esta evaluación

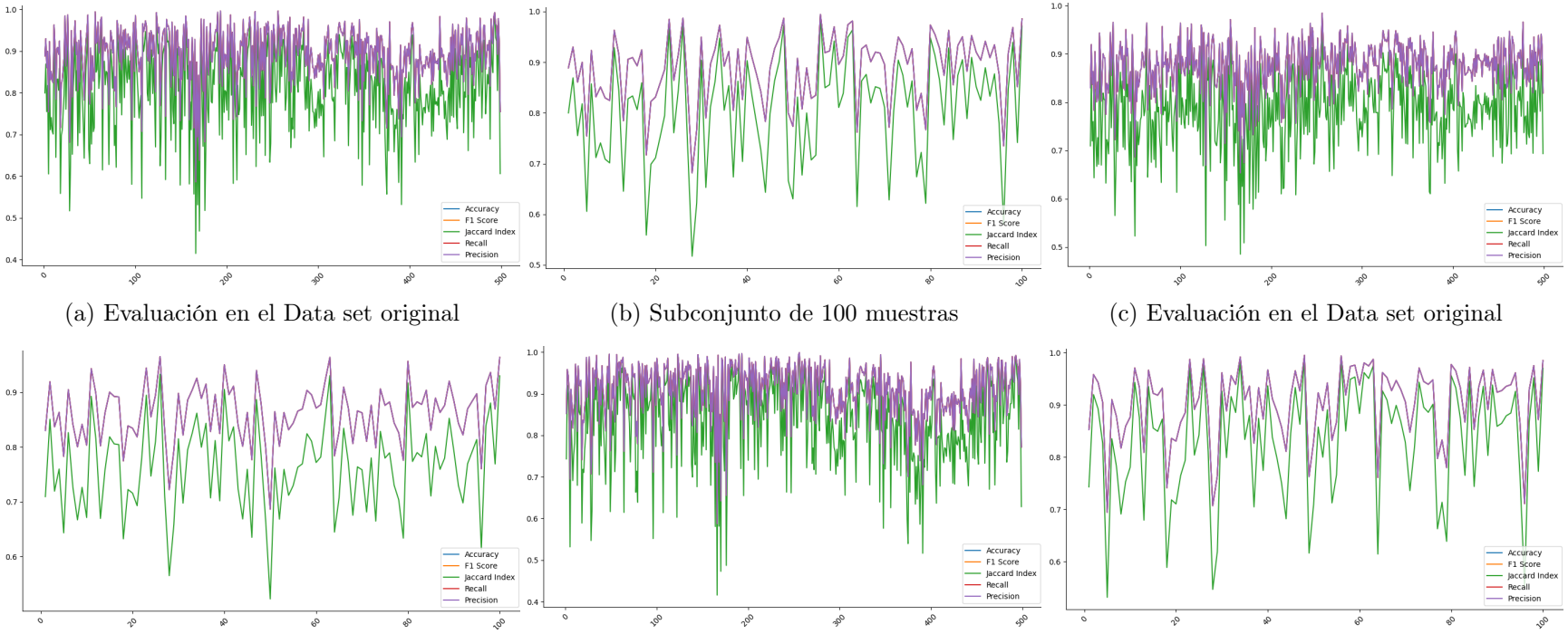
preliminar permitió medir el desempeño de cada modelo en términos de cuán bien las máscaras de segmentación predichas coinciden con las etiquetas reales.

Para cuantificar esta coincidencia, se utilizaron varias métricas, incluyendo la precisión general (accuracy), la armonía entre la precisión y la exhaustividad (F1), la similitud entre las categorías predichas y reales (IoU o jaccard), la proporción de verdaderos positivos entre todas las predicciones positivas (precision) y la proporción de verdaderos positivos entre todas las instancias reales positivas (recall). Estas métricas proporcionan una evaluación integral del rendimiento de cada modelo desde diferentes perspectivas, la tabla 2.14 presenta el promedio de las métricas de evaluación, para cada versión de los modelos U-Net2D, SegNet y DeepLabV3+ sobre el conjunto de entrenamiento base.

| **Versión** | **Modelo** | **Accuracy** | **F1** | **IoU** | **Recall** | **Precision** |
|---|---|---|---|---|---|---|
| 1 | U-Net2D | 89.94 % | 89.94 % | 82.25 % | 89.94 % | 89.94 % |
| 2 | U-Net2D | 89.19 % | 89.19 % | 81.07 % | 89.19 % | 89.19 % |
| 3 | U-Net2D | 86.52 % | 86.52 % | 76.71 % | 86.52 % | 86.52 % |
| 1 | SegNet | 89.83 % | 89.83 % | 82.01 % | 89.83 % | 89.83 % |
| 2 | SegNet | 88.84 % | 88.84 % | 80.4 % | 88.84 % | 88.84 % |
| 3 | SegNet | 87.45 % | 87.45 % | 78.05 % | 87.45 % | 87.45 % |
| 1 | DeepLabV3+ | 90.54 % | 90.54 % | 83.33 % | 90.54 % | 90.54 % |
| 2 | DeepLabV3+ | 90.64 % | 90.64 % | 83.5 % | 90.64 % | 90.64 % |

Tabla 2.14: Métricas de evaluación promedio obtenidas (500 imágenes)

En la figura 2.23a, se presenta la evolución de las métricas de rendimiento a lo largo del tiempo en el conjunto de datos original. Además de evaluar el rendimiento en el conjunto de datos básico completo, también se realizó un análisis en un subconjunto más reducido de 100 datos, como se muestra en la figura 2.23. Este enfoque permitió una visión detallada de las métricas y su comportamiento. En términos generales, las métricas de evaluación de los tres modelos estudiados en las 500 imágenes del conjunto de entrenamiento principal muestran fluctuaciones en un rango que oscila entre 0.6 y 1.0. Estos valores indican un rendimiento variable, aunque en su mayoría se observa una tendencia al aumento en las imágenes evaluadas.




(a) Evaluación en el Data set original (b) Subconjunto de 100 muestras (c) Evaluación en el Data set original

(d) Subconjunto de 100 muestras (e) Evaluación en el Data set original (f) Subconjunto de 100 muestras

Figura 2.23: Evaluación U-Net2D (a-b), SegNet(c-d), DeepLabV3+(e-f)

Además de las métricas de evaluación utilizadas para calificar las predicciones de los modelos de segmentación semántica, también se consideraron variables cruciales, como el tiempo de entrenamiento real y teórico, así como los tiempos de inferencia por lote y por imagen para cada versión generada de los modelos. Esta información, representada en la figura 2.24, revela una diferencia notable entre el tiempo de entrenamiento teórico y el tiempo de entrenamiento real, fenómeno que se puede atribuir a diversos factores:

- Carga de la CPU/GPU: El tiempo de entrenamiento real puede aumentar si la CPU o la GPU están ocupadas con otras tareas simultáneas. En el contexto de Google Colab, la disponibilidad de recursos computacionales y la posibilidad de superar la cuota asignada influyen de manera significativa en los tiempos de entrenamiento e inferencia. Por lo tanto, resulta fundamental garantizar que el entorno de Colab tenga acceso a la GPU para optimizar los tiempos de ejecución y entrenamiento.
- Operaciones de E/S: Las operaciones de lectura y escritura de datos en el disco pueden requerir tiempo, y estos aspectos no se reflejan en los tiempos de inferencia por lote. Por esta razón, la configuración adecuada del generador de datos de entrenamiento y la implementación de un callback para guardar los mejores modelos al finalizar cada época son consideraciones importantes.
- Variabilidad en los tiempos de inferencia: Los tiempos de inferencia por lote pueden fluctuar durante el proceso de entrenamiento. En las etapas iniciales, cuando los pesos del modelo se encuentran lejos de una solución óptima, los pasos de entrenamiento pueden ser más rápidos. Conforme el modelo converge, es normal que cada paso tome más tiempo.

Por lo tanto, aunque el tiempo teórico proporciona una estimación inicial, es crucial reconocer que el tiempo real puede ser sustancialmente mayor debido a estos y otros factores que influyen en el proceso de entrenamiento y evaluación de los modelos de segmentación semántica.

En este contexto, las máscaras de segmentación semántica se refieren a las salidas de cada modelo (U-Net2D, SegNet, DeeplLabV3+) que predicen la ubicación de la herida en una imagen, como se puede observar en los mosaicos de las figuras 2.25, 2.26, 2.27. Estas máscaras son comparadas con las etiquetas verdaderas, que son las ubicaciones reales de las heridas en las imágenes, determinadas por un experto y etiquetadas manualmente. Al comparar estas máscaras de segmentación con las etiquetas reales, podemos obtener una medida cuantitativa del rendimiento de cada modelo.

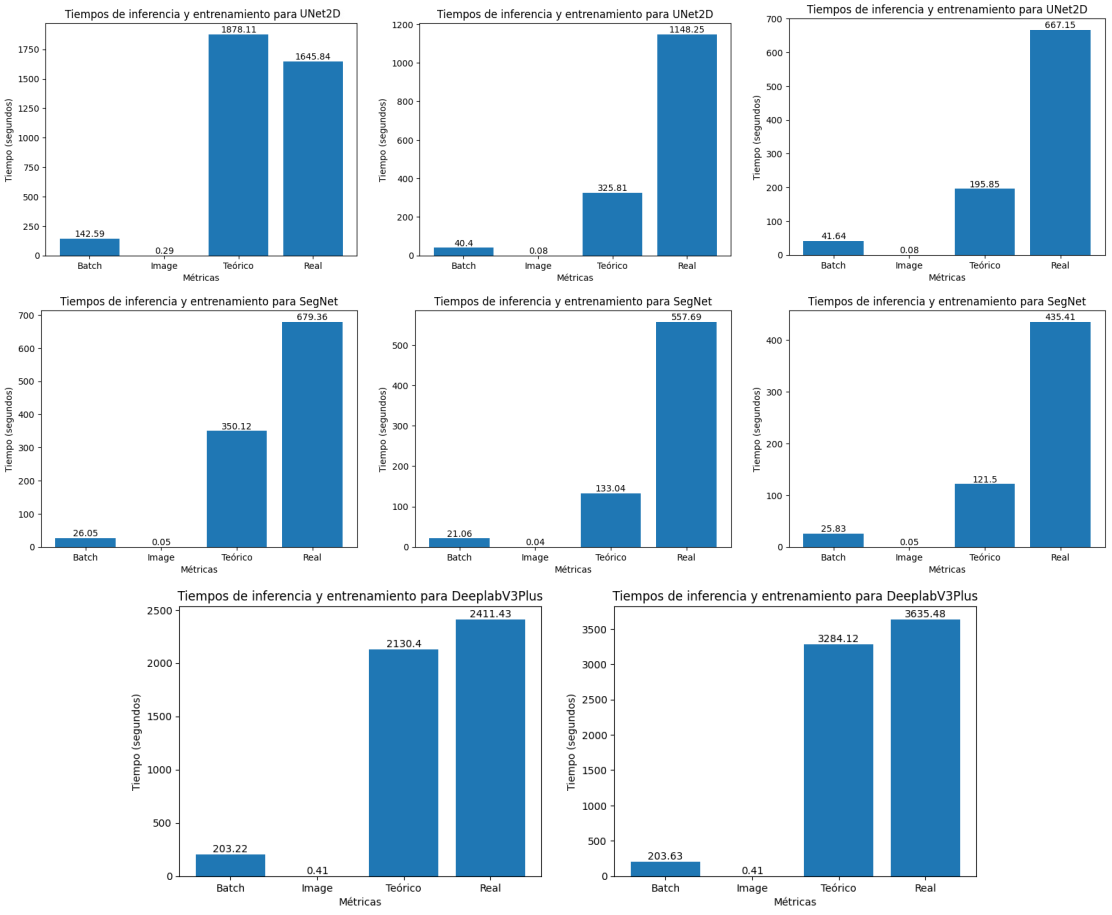



Figura 2.24: Tiempos de entrenamiento e inferencia para los modelos

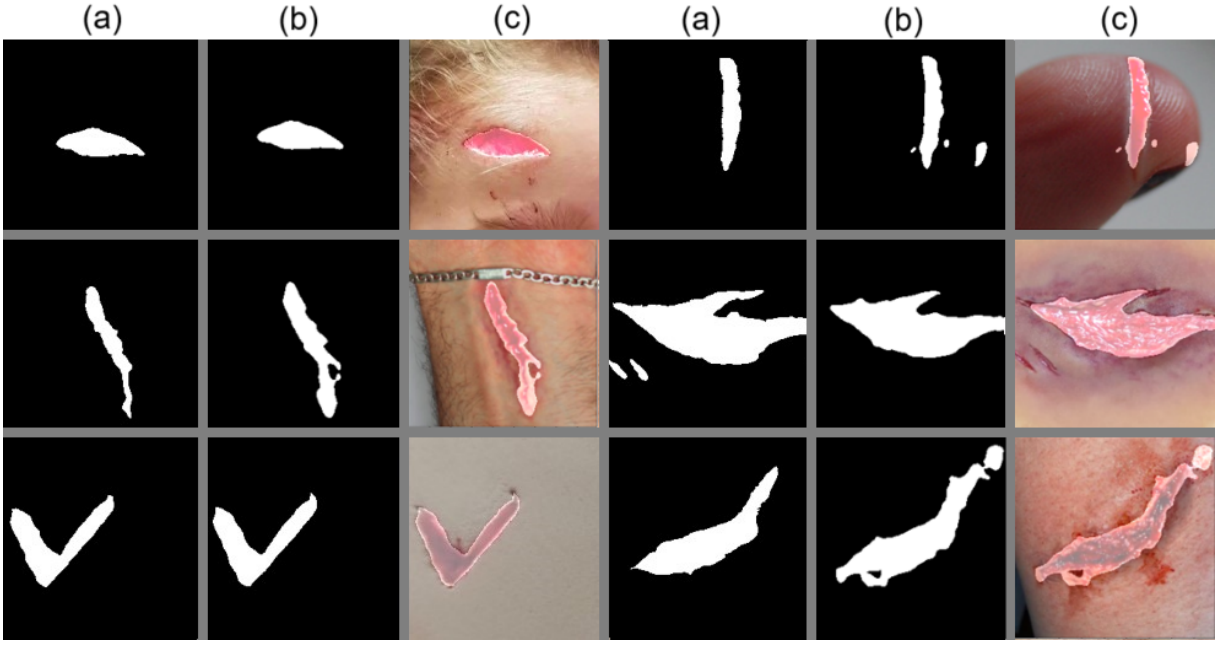



Figura 2.25: (a) etiqueta, (b) predicción y (c) superposición (U-Net2D)

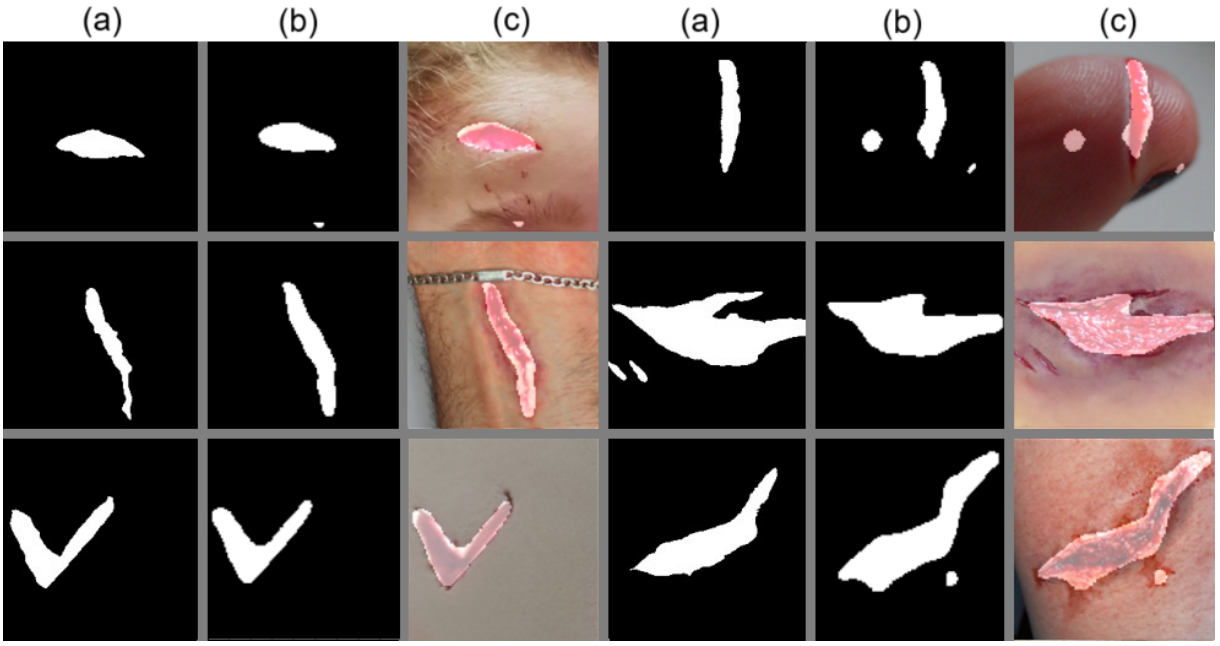



Figura 2.26: (a) etiqueta, (b) predicción y (c) superposición (SegNet)

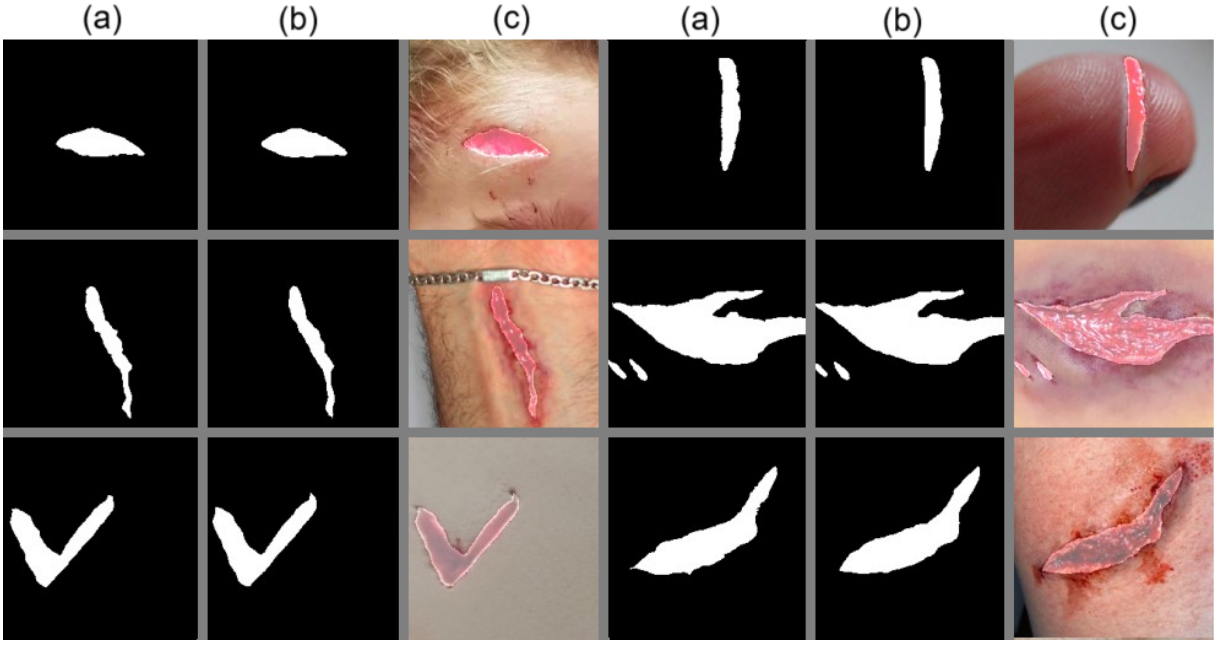



Figura 2.27: (a) etiqueta, (b) predicción y (c) superposición (DeepLabV3+)

### 2.4.3. Conclusiones preliminares de los modelos

En el marco de esta investigación, que se centra en la segmentación y reconocimiento de heridas corto punzantes, se ha llevado a cabo una validación del rendimiento de tres modelos de aprendizaje automático: U-Net2D, SegNet y DeepLabV3+. A partir de esta evaluación, se han extraído varias conclusiones preliminares que podrían tener implicaciones significativas para la investigación futura:

- **U-Net2D:** Este modelo demostró ser prometedor en la tarea de segmentación de heridas corto punzantes. Durante el proceso de entrenamiento, se observó que el modelo fue capaz de aprender características relevantes y mejorar su rendimiento a medida que avanzaban las épocas. Sin embargo, también se observaron fluctuaciones en las métricas, lo que sugiere cierta sensibilidad a la variabilidad de los datos. Aunque el modelo demostró ser eficaz en la identificación de los bordes de las heridas y detalles finos, su capacidad para generalizar y mantener un rendimiento estable podría beneficiarse de ajustes en sus hiperparámetros o la ampliación del conjunto de datos para aportar más variabilidad, lo que permitiría al modelo representar mejor las heridas en el mundo real.

- **SegNet:** Este modelo mostró resultados similares a U-Net2D en términos de aprendizaje y mejora durante el entrenamiento. Sin embargo, se observaron patrones de fluctuación en las métricas, lo que indica cierta variabilidad en el rendimiento. A pesar de su buena ubicación espacial de las heridas, el modelo no delimita los bordes con efectividad, clasificando partes de piel sana como herida. Al igual que con U-Net2D, SegNet podría beneficiarse de una optimización adicional para lograr un rendimiento más consistente. Sin embargo, su falta de precisión en los detalles finos y los bordes lo hace inviable para aplicaciones de sutura.

- **DeepLabV3+ con ResNet50:** Entre los modelos evaluados, DeepLabV3+ con ResNet50 como codificador se destacó por su rendimiento más estable y consistente. Las métricas de entrenamiento y validación mostraron una tendencia de mejora constante, con valores superiores al 80 %. Además, las gráficas de pérdida no indicaron signos de sobre ajuste, lo que sugiere que el modelo estuvo aprendiendo características generalizables en lugar de memorizar los datos de entrenamiento. Estos hallazgos respaldan la idoneidad de DeepLabV3+ con ResNet50 para aplicaciones de segmentación de heridas corto punzantes en el contexto de sutura.

# Capítulo 3

# Evaluación del desempeño de los modelos

En este capítulo, se evalúa el desempeño de los modelos U-Net2D, SegNet y DeepLabv3+ en la tarea de segmentación semántica de heridas corto punzantes. Para ello, se utilizó una cámara web encargada de capturar los frames de video en vivo y se aplicaron cada uno de los modelos para segmentar y reconocer las heridas presentes en los frames del video generado en vivo.

El modelo DeepLabV3+, considerado como el mejor modelo entrenado en la tarea de segmentación hasta el momento, es utilizado como referencia o experto para comparar y calificar las predicciones de los otros modelos. Esta comparación nos permite medir no solo la capacidad de estos modelos para segmentar heridas en tiempo real, sino también la similitud o diferencia entre sus resultados y los del modelo de referencia.

Para cuantificar el desempeño de cada modelo, se hizo uso de una serie de métricas que incluyen accuracy_score, f1_score, jaccard_score, precision_score y recall_score (1.3 pág. 40). Estas métricas ofrecen una evaluación completa del desempeño de cada modelo desde distintos ángulos, permitiendo identificar las fortalezas y debilidades de cada uno.

Al final de este capítulo y a partir de los resultados de las pruebas, se tiene una visión clara del desempeño relativo de estos modelos en la tarea de segmentación semántica binaria, lo que facilitará tomar decisiones informadas sobre qué modelo utilizar en aplicaciones médicas futuras.

## 3.1. Componentes de la Evaluación

Para evaluar el desempeño real de los modelos seleccionados en la tarea de segmentación semántica de heridas suturables, se diseñó, en primer lugar, un escenario de pruebas, como se puede observar en la figura 3.1, que incluye varios componentes clave listados a continuación:

| **Componente** | **Descripción** |
|---|---|
| Trozo de Látex | Material utilizado para simular un torso humano. Se realizaron cortes de diferentes profundidades y se usó colorante artificial de alimentos para simular la presencia de sangre en la laceración, proporcionando una representación realista de las heridas suturables. |
| Cámara Web Logitech C920s PRO | Encargada de capturar los frames de video en vivo, bajo condiciones de iluminación variable y que servirán de imagen de entrada para los modelos. Fue ubicada a 30 cm del objetivo. |
| Aro de Luz LED | Utilizado para introducir condiciones de ruido lumínicas durante la segmentación en vivo. Lo cual permite evaluar la robustez de los modelos en condiciones menos que ideales. |

Tabla 3.1: Escenario de evaluación

Figura 3.1: Escenario de pruebas diseñado

### 3.1.1. Plan de Pruebas

Una vez definido el escenario de evaluación, que permita inducir condiciones de variabilidad en la iluminación, rotación y desplazamiento del objetivo durante la segmentación de heridas corto punzantes en vivo. Se llevó a cabo una evaluación del desempeño de los modelos, como se observa en la tabla 3.2, siguiendo el plan de pruebas estructurado a continuación:

1. **Segmentación en vivo:** Se realizó la segmentación de heridas en tiempo real, utilizando la cámara web Logitech C920s PRO, conectada vía USB al ordenador, mientras se ejecuta el script **Pred_live.py** en un entorno local de Conda 23.7.2. Esto permitió evaluar el rendimiento de los modelos en un entorno dinámico y en condiciones de iluminación cambiantes, simulando una situación práctica de toma de imágenes en el campo médico.

2. **Generación de video y predicciones:** Se generó un video de aproximadamente 30 segundos a una velocidad de fotogramas de 10 FPS, a partir de los frames capturados en vivo por la cámara web. Este video representa un registro de la segmentación en tiempo real realizada por cada modelo. Además, se generó un archivo NumPy que contiene las predicciones del modelo evaluado, en los fotogramas del video generado.

3. **Selección del modelo de referencia:** Basados en la segmentación en vivo a través de la cámara web y en las métricas de entrenamiento y validación previamente analizadas, se identificó que el modelo DeepLabV3+ con ResNet50 como endocer se destacó por su precisión en la delimitación de los bordes de las heridas y su capacidad para detectar múltiples instancias de heridas corto punzantes en la piel simulada. En consecuencia, se optó por utilizar dicho modelo como el modelo de referencia para etiquetar los fotogramas del video generado.

4. **Comparación de predicciones:** Las predicciones de los modelos secundarios (SegNet y U-Net2D) se calificaron en relación con las etiquetas generadas por el modelo DeepLabV3+ (modelo de referencia). Esto permitió cuantificar la precisión y el rendimiento relativo de los modelos secundarios en comparación con el modelo principal.

5. **Visualización y análisis:** Se llevó a cabo un análisis de las métricas de evaluación, incluyendo `accuracy_score`, `f1_score`, `jaccard_score`, `precision_score` y `recall_score`. Estas métricas se calcularon durante los fotogramas del video generado, lo que permitió observar la tendencia de rendimiento a lo largo de los frames

del video y proporcionó información valiosa sobre la calidad de la segmentación en situaciones dinámicas.

| **Prueba** | **Descripción** |
|---|---|
| Segmentación en vivo | Utilización de la webcam para segmentación en tiempo real. Evaluación del rendimiento bajo condiciones de uso en vivo. |
| Generación de video | Creación de un video de 30 segundos a 10 FPS que registre la segmentación en vivo. Guardado de las predicciones del modelo en cada frame. |
| Comparación de modelos | Uso del mejor modelo elegido para etiquetar los frames del video generado. |
| Calificación de predicciones | Evaluación de las predicciones de cada modelo, en comparación con las etiquetas reales generadas por el modelo de referencia DeepLabV3+. |
| Visualización de métricas | Análisis visual del rendimiento de los modelos a lo largo de los frames del video generado. |

Tabla 3.2: Resumen del plan de Pruebas

En resumen, este enfoque de pruebas meticuloso y basado en la simulación proporcionó una evaluación completa y realista del rendimiento de los modelos de segmentación seleccionados para la tarea de reconocimiento y segmentación de heridas corto punzantes. Los resultados de esta evaluación son fundamentales para comprender cómo se desempeñarían estos modelos en aplicaciones médicas prácticas del mundo real.

## 3.2. Resultados y comparación de los modelos

Para evaluar el desempeño de los modelos entrenados para la segmentación semántica de heridas corto punzantes bajo condiciones realistas y determinar su robustez ante la variabilidad de iluminación y desplazamiento, se realizó la segmentación de heridas en vivo, utilizando el escenario dispuesto para tal fin y hallando los siguientes resultados:

1. **DeepLabV3+** demostró desde la etapa de entrenamiento su superioridad respecto a U-Net2D y SegNet. Sus métricas de entrenamiento y validación mostraron una

tendencia de mejora constante, con valores superiores al 80 % (2.22). Además, las gráficas de pérdida no indicaron signos de sobre ajuste. Su buen rendimiento quedó evidenciado en la segmentación en vivo, como se observa en la secuencia de frames 3.2.

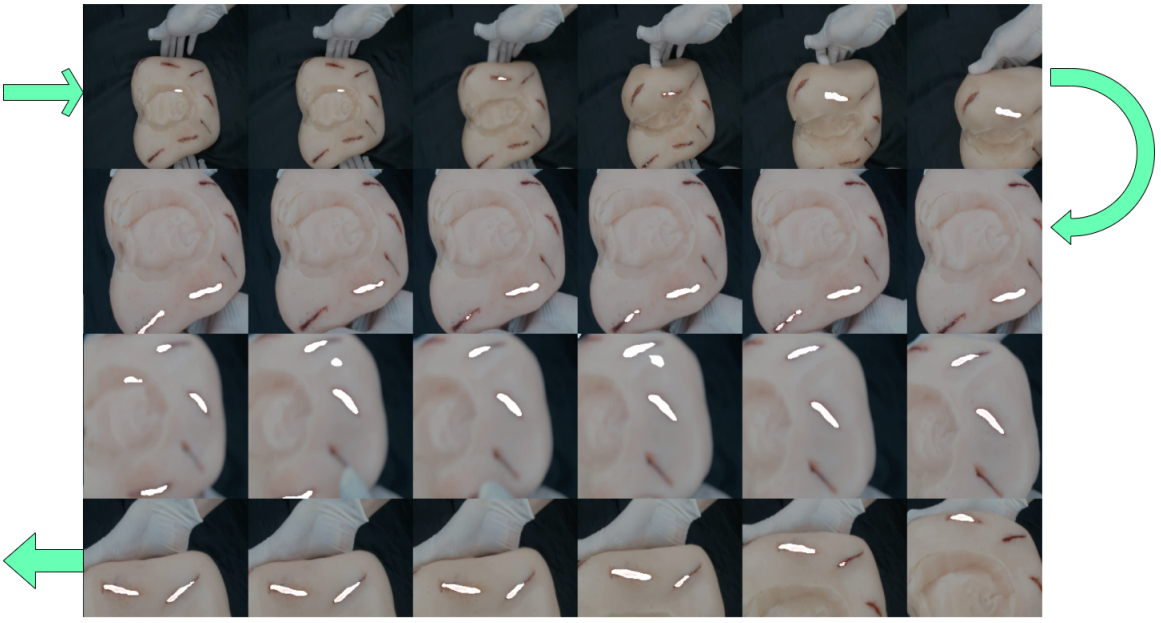


Figura 3.2: Secuencia de segmentación en vivo con DeepLabV3+, **(Ver video aquí)**

Este modelo sobresale por su capacidad para delimitar con precisión los bordes de las heridas y detectar múltiples instancias de heridas en la piel simulada, incluso cuando los datos de entrenamiento en su mayoría presentan una sola instancia de herida. Este logro indica que el modelo ha alcanzado un alto nivel de generalización. Estos factores respaldan su elección como modelo de referencia para evaluar las predicciones de los modelos secundarios (U-Net2D y SegNet) en los fotogramas del video generado.

2. **U-Net2D** por su parte, evidenció un buen rendimiento en la segmentación de heridas en vivo, sorprendiendo su capacidad para detectar detalles pequeños o segmentar heridas con muy poco nivel de abertura, en este aspecto podría llegar a ser superior a DeepLabV3+. Sin embargo, se confirmó la conclusión preliminar sobre este modelo respecto a la sensibilidad a la variabilidad de los datos de entrada. Este modelo es mucho más sensible ante las variaciones de iluminación o desplazamiento, llegando a clasificar erróneamente sombras como heridas, como se puede evidenciar en la secuencia de frames registrada 3.3.

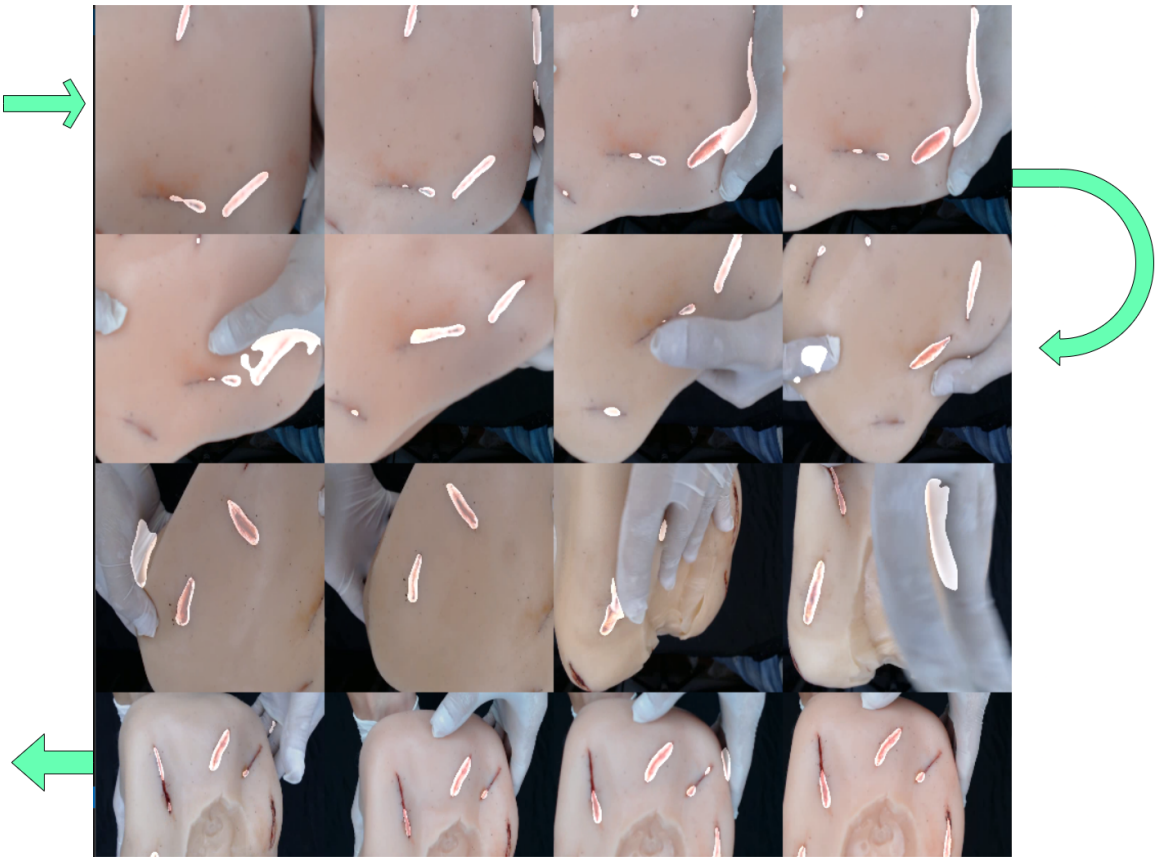


Figura 3.3: Secuencia de segmentación en vivo con U-Net2D, **(Ver video aquí)**

Estos errores en la clasificación de píxeles por parte de U-Net2D, quedan demostrados por las tendencias, las gráficas de pérdida durante el entrenamiento de las tres versiones generadas (2.20). Donde se logra evidenciar la tendencia al sobre ajuste del modelo, lo cual es un indicador de un bajo nivel de generalización de los patrones de características significativas aprendidas.

3. **SegNet** a su vez, demostró un rendimiento similar al modelo U-Net2D desde las etapas tempranas de entrenamiento y, de hecho, exhibió una mayor estabilidad, como se evidencia en la gráfica 2.20. Sin embargo, a través de la prueba de segmentación en tiempo real, ilustrada en la secuencia de fotogramas 3.4, se pudo confirmar la conclusión preliminar. A pesar de que este modelo logra una ubicación espacial precisa de las heridas, presenta dificultades en la delimitación efectiva de los bordes, lo que resulta en la clasificación errónea de áreas de piel sana como herida. Este comportamiento subraya su tendencia al sobre ajuste, lo cual afecta negativamente su capacidad de generalización en la predicción de imágenes nuevas. Estas limitaciones hacen que este modelo sea inadecuado para aplicaciones médicas de alta precisión, como las suturas.

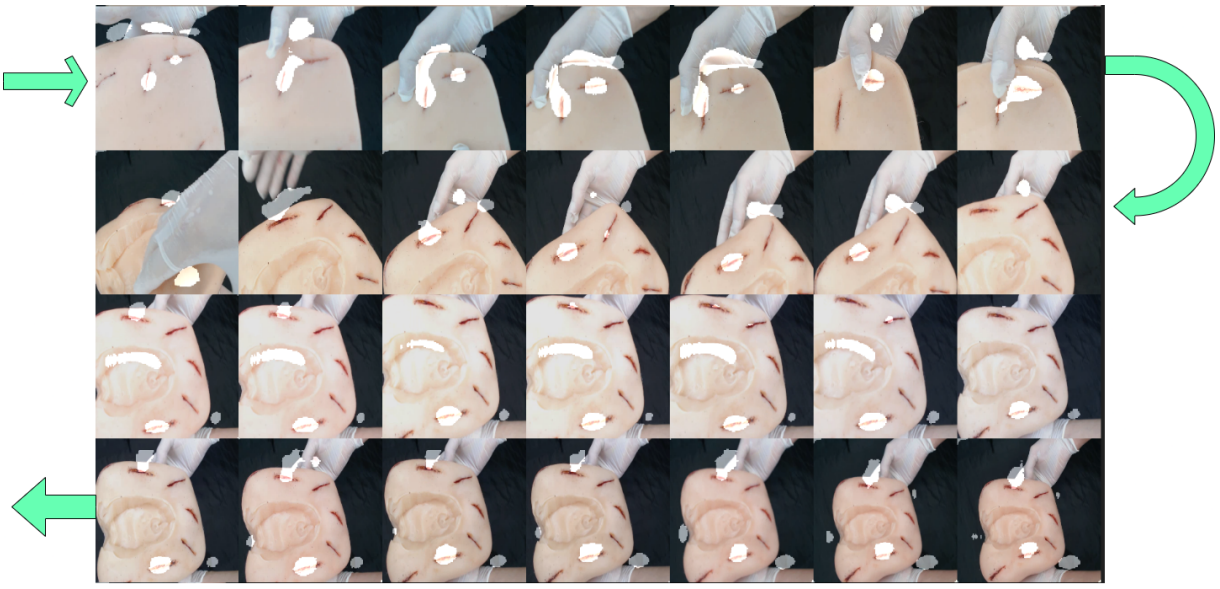


Figura 3.4: Secuencia de segmentación en vivo con SegNet, **(Ver video aquí)**

Con base en los resultados de la segmentación en vivo de heridas corto punzantes, donde se demuestra el rendimiento superior del modelo DeepLabV3+ con ResNet50 como codificador en términos de precisión en la segmentación y su resistencia ante la variabilidad en los datos de entrada, se designa este modelo como referencia para calificar a sus contra partes U-Net2D y SegNet. Para cuantificar esta calificación, se usó cada video generado durante el registro de la segmentación en vivo y su archivo de predicciones por frame asociado a cada modelo. Esta base de datos permitió inferir con el modelo de referencia DeepLabV3+ sobre los videos de registro para cada modelo secundario (U-Net2D y SegNet) y así comparar sus máscaras de segmentación predichas como se observa en la figura 3.5 y 3.7.

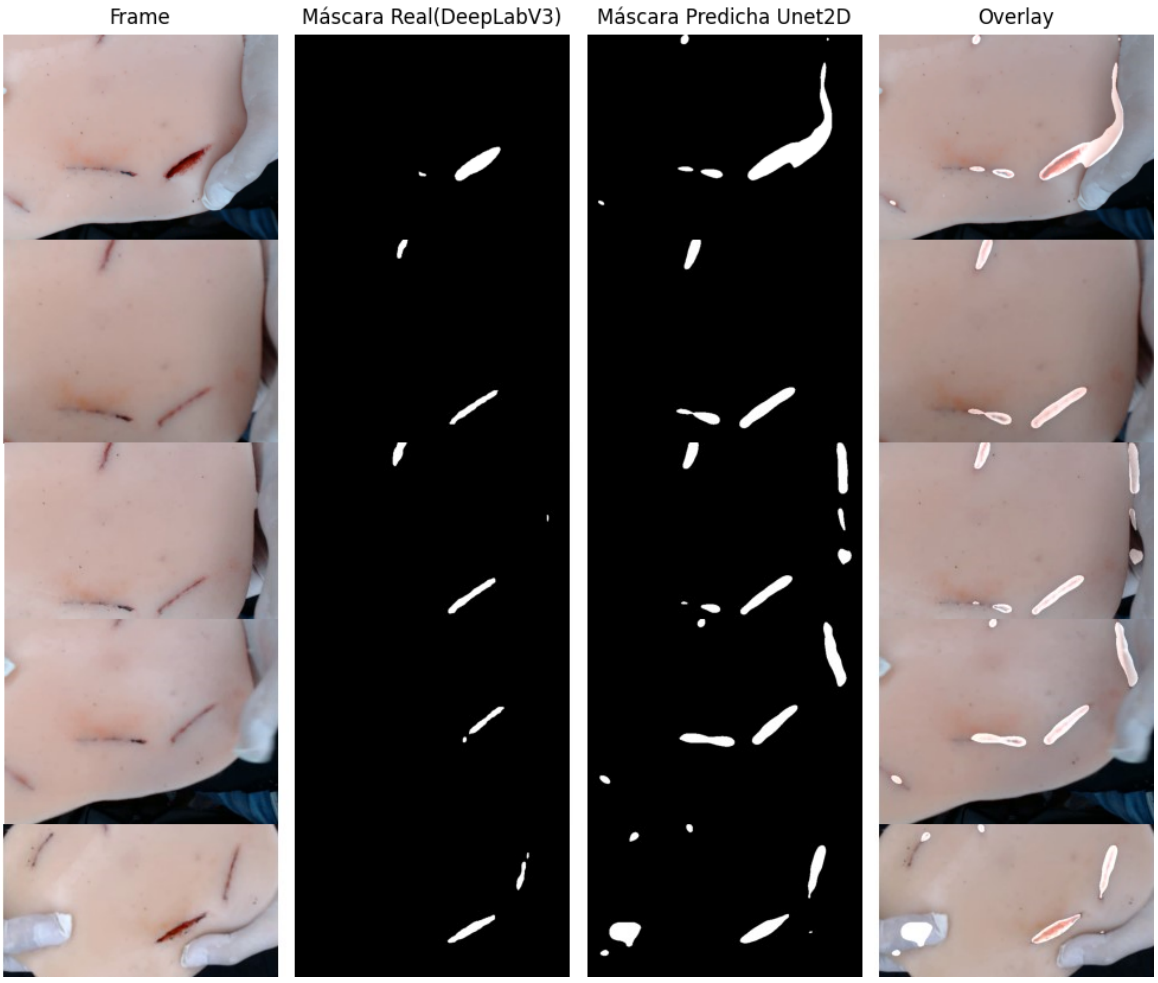



Figura 3.5: Comparación de las predicciones por frame (U-Net2D)

La figura 3.5 muestra ejemplos visuales de las máscaras de segmentación predichas por U-Net2D y DeepLabV3+ para las heridas en el video generado. Se pone en evidencia la superioridad el modelo U-Net2D en cuanto a preservar los detalles finos, pues ha calificado correctamente píxeles más pequeños que pertenecen a heridas con muy bajo nivel de abertura, mientras que DeepLabV3+ como modelo de referencia no pudo detectarlas. A pesar de lo anterior, U-Net2D demuestra su alta sensibilidad con las sombras, donde la zona de contacto de la mano y la piel, o el pliegue entre los dedos del cirujano, son clasificados erróneamente como heridas.

De igual forma, la tabla 3.3 muestra el promedio de las métricas de evaluación en los frames del video generado. Adicionalmente, la gráfica 3.6 muestra la tendencia de varias métricas de evaluación para el modelo U-Net2D, en comparación con las predicciones del modelo DeepLabV3+ como referencia. En primer lugar, el índice de Jaccard es mayormente alto, lo que nos indica una buena similitud entre ambas predicciones. Además, se puede

observar una precisión general por encima del 90 %, lo que indica que el modelo U-Net2D generalmente hace predicciones correctas, apoyado por la tendencia de la puntuación F1 que sugiere que el modelo tiene un buen equilibrio entre la precisión y la sensibilidad.

| **Modelo** | **Accuracy** | **F1** | **IoU** | **Recall** | **Precision** |
|---|---|---|---|---|---|
| U-Net2D | 97.67 % | 97.67 % | 95.49 % | 97.67 % | 97.67 % |

Tabla 3.3: Métricas de evaluación promedio (video en vivo)

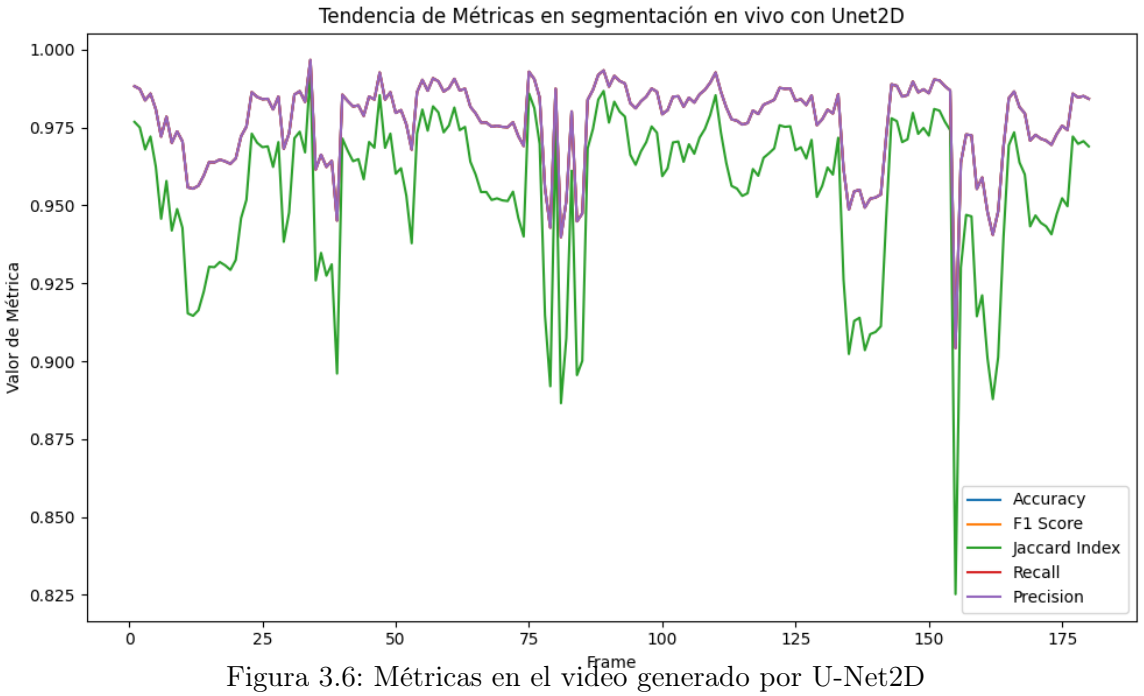



Figura 3.6: Métricas en el video generado por U-Net2D

Por otro lado, en la figura 3.7 se presentan ejemplos visuales de las máscaras de segmentación generadas por SegNet y DeepLabV3+ para las heridas en el video generado en tiempo real. En estas imágenes, se evidencia la notoria sensibilidad de SegNet ante las variaciones en la iluminación y el desplazamiento, lo que se traduce en una clasificación deficiente de píxeles y una segmentación de bordes abrupta. En este contexto, a pesar de que SegNet registra tiempos de inferencia más rápidos, su idoneidad para la tarea de segmentación semántica de heridas en el contexto de sutura se ve comprometida, ya que la correcta delimitación de los bordes resulta fundamental.

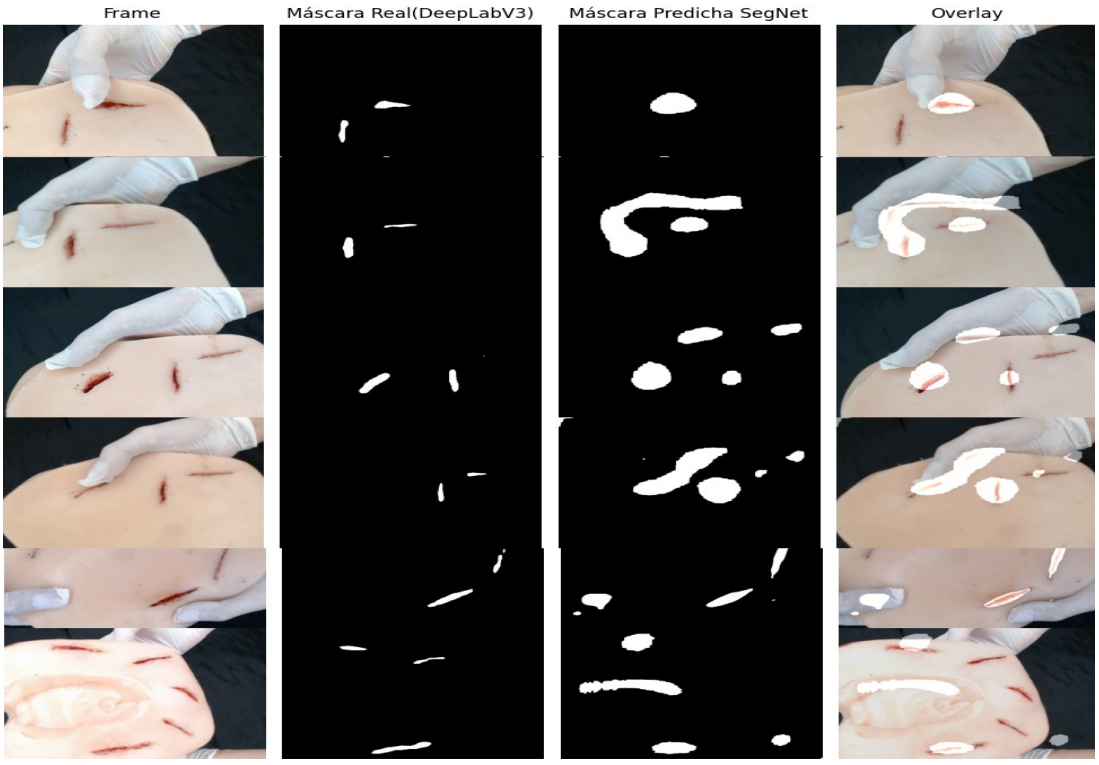



Figura 3.7: Comparación de las predicciones por frame (SegNet)

Adicionalmente, la tabla 3.4 muestra el promedio de las métricas de evaluación en los frames del video generado por SegNet. Además, en la figura 3.8, se analizan las métricas de evaluación correspondientes a dicho modelo, en comparación con las predicciones del modelo de referencia. A pesar de que las métricas reflejan una precisión promedio superior al 84 %, esta cifra no se corresponde con la apreciación visual de la calidad de la segmentación. SegNet muestra una alta sensibilidad ante las variaciones de iluminación y desplazamiento, lo que lleva a una clasificación deficiente de píxeles y a una segmentación de bordes notoriamente brusca. Esta discrepancia entre las métricas y la calidad visual podría explicarse por el hecho de que las métricas de evaluación no siempre capturan de manera completa los aspectos visuales de la segmentación. Por ejemplo, si SegNet comete pequeños errores en varias ubicaciones distintas, esto podría dar lugar a una segmentación visualmente imprecisa, a pesar de que aún obtenga una puntuación alta en las métricas debido a la gran cantidad de aciertos.

| **Modelo** | **Accuracy** | **F1** | **IoU** | **Recall** | **Precision** |
|---|---|---|---|---|---|
| SegNet | 95.15 % | 95.15 % | 90.8 % | 95.15 % | 95.15 % |

Tabla 3.4: Métricas de evaluación promedio (video en vivo)

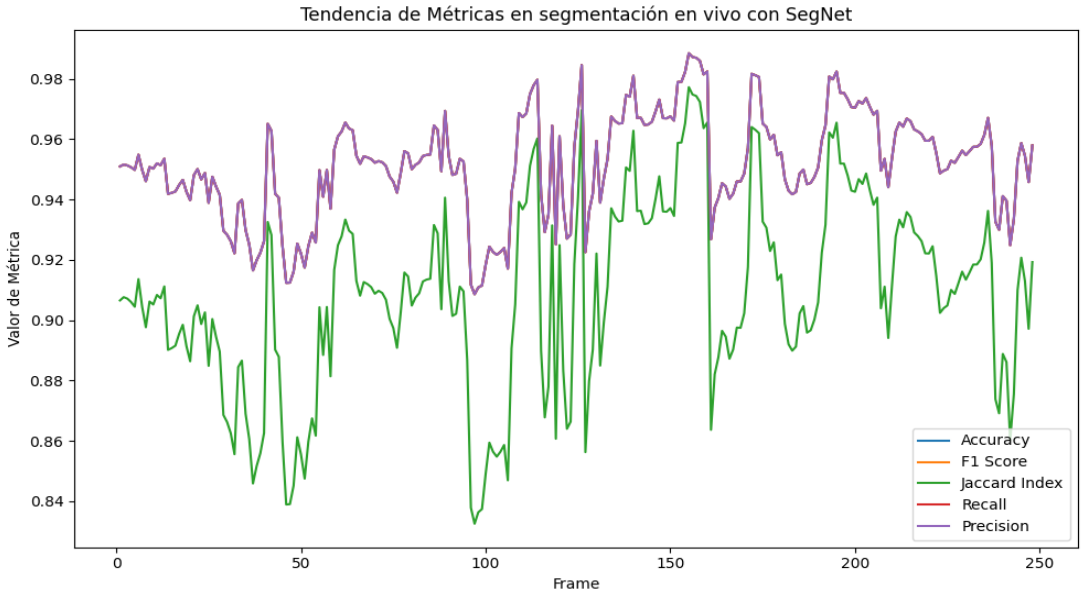



Figura 3.8: Métricas en el video generado por SegNet

En resumen, los resultados de las métricas de evaluación de la segmentación de heridas en tiempo real han revelado una precisión promedio superior al 90 % para los modelos secundarios U-Net2D y SegNet en comparación con el modelo de referencia DeepLabV3+ con ResNet50 como codificador. Sin embargo, es fundamental comprender que esta estimación de precisión no necesariamente refleja la calidad real de la segmentación ni la percepción visual de las imágenes segmentadas. Esta discrepancia se debe principalmente al enfoque de comparación que se ha empleado. En la evaluación, se ha medido el nivel de coincidencia de las predicciones de los modelos secundarios con las predicciones del modelo de referencia. Sin embargo, esta coincidencia no garantiza que el modelo de referencia haya segmentado de manera correcta todas las heridas presentes en la piel sintética. Es decir, incluso si las predicciones de los modelos secundarios coinciden con las del modelo de referencia, existe la posibilidad de que tanto el modelo de referencia como los modelos secundarios hayan cometido errores en la segmentación. Esto puede deberse a desafíos en la detección precisa de los bordes de las heridas o a dificultades para diferenciar entre áreas de piel sana y áreas afectadas por heridas corto punzantes.

Por lo tanto, es importante tener en cuenta que las métricas utilizadas, si bien proporcionan información valiosa, no ofrecen una evaluación completa de la calidad de la segmentación en vivo. La evaluación visual, la consideración de la precisión en la delimitación de bordes y la discriminación entre heridas y piel sana son aspectos igualmente esenciales para evaluar la idoneidad de estos modelos en aplicaciones médicas de alta precisión, como las relacionadas con suturas y procedimientos quirúrgicos.

# Capítulo 4

# Discusión, Conclusiones y Trabajos futuros

## 4.1. Discusión y limitaciones

Para verificar el rendimiento de los modelos entrenados, se realizó una comparación de nuestros resultados de validación en la segmentación de imágenes médicas con estudios previos relacionados con laceraciones en la piel. Esta comparación se basó en el análisis de métricas compartidas con el presente trabajo de segmentación de heridas corto punzantes. La realización de esta comparación tiene varios objetivos importantes. En primer lugar, permite evaluar cómo se desempeñan los modelos entrenados en el contexto de la segmentación de heridas en la piel en comparación con otros esfuerzos previos. Esto proporciona una perspectiva valiosa sobre la efectividad y la competitividad del enfoque adoptado. Además, esta comparación también arroja luz sobre las posibles áreas de mejora en los modelos. Si se observan discrepancias significativas entre nuestros resultados y los resultados anteriores, se pueden identificar aspectos específicos que requieren atención y refinamiento.

La tabla 4.1 muestra los resultados de estudios anteriores que han empleado arquitecturas de CNN similares y métricas comparables. Mientras que la tabla 4.2 ofrece una visión cualitativa de los resultados obtenidos en esta investigación en comparación con trabajos previos de imágenes biomédicas.

| Modelo | Métricas | Descripción |
|---|---|---|
| U-Net[12] | Dice = 0.93 | Segmentación de células nerviosas en imágenes de microscopía electrónica |
| U-Net[13] | Dice = 0.82 | Segmentación de vasos sanguíneos en imágenes de retina (IoU = 0.71, Accuracy = 0.95) |
| U-Net[15] | IoU = 0.93 | Segmentación de membranas celulares en imágenes de microscopía electrónica |
| U-Net[16] | Dice = 0.85 | Segmentación de tumores cerebrales en imágenes de resonancia magnética (Precision = 0.86, Recall = 0.87) |
| ResNet[17] | Top-1 = 0.22 | Clasificación de imágenes en 1000 categorías (Top-5 = 0.06) |
| ResNet[18] | Acc. = 0.95 | Clasificación de imágenes en 10 categorías |
| RefineNet[19] | mIoU = 0.81 | Segmentación de escenas urbanas en 19 categorías |
| RefineNet[20] | mIoU = 0.83 | Segmentación de objetos en 20 categorías |
| FCN[21] | mIoU = 0.62 | Segmentación de objetos en 20 categorías |
| FCN[21] | mIoU = 0.29 | Segmentación de escenas interiores en 40 categorías |
| Mask-RCNN[22] | AP = 0.38 | Segmentación de instancias en 80 categorías (AP50 = 0.59, AP75 = 0.41) |
| Mask-RCNN[23] | Dice = 0.83 | Segmentación de nódulos pulmonares en imágenes de tomografía computarizada (Precision = 0.84, Recall = 0.84) |
| SegNet[24] | mIoU = 0.28 | Segmentación de escenas interiores en 37 categorías |
| SegNet[25] | mIoU = 0.46 | Segmentación de escenas urbanas en 11 categorías |
| PSPNet[16] | mIoU = 0.43 | Segmentación de escenas en 150 categorías |
| DeepLabv3+[28] | mIoU = 0.89 | Segmentación de objetos en 20 categorías |
| DeepLabv3+[27] | mIoU = 0.82 | Segmentación de escenas urbanas en 19 categorías |

Tabla 4.1: Resultados de investigaciones previas en diversas tareas de segmentación.

| **Modelo** | **Precisión** | **Recall** | **Dice** | **Jaccard** |
|---|---|---|---|---|
| DeepLabv3+ (Nuestro) | 0.899 | 0.859 | 0.863 | 0.806 |
| SegNet (Nuestro) | 0.791 | 0.688 | 0.655 | 0.573 |
| U-Net (Nuestro) | 0.806 | 0.766 | 0.587 | 0.659 |
| SegNet (Badrinarayanan et al., 2017 [62]) | — | — | 0.886 | 0.752 |
| DeeplabV3+ (Chen et al., 2018b [28]) | 0.892 | — | 0.857 | 0.762 |
| U-Net (Ezad et al., 2015[14]) | 0.899 | — | 0.850 | 0.881 |

Tabla 4.2: Comparación de modelos de segmentación usados en casos de estudio biomédicos

Esta comparación es esencial, ya que proporciona una base sólida para contextualizar la contribución de nuestro trabajo en el campo de la segmentación de heridas en la piel y aporta información relevante para futuras investigaciones y aplicaciones clínicas. A través de esta comparación cualitativa, se busca proporcionar una comprensión más completa de la eficacia de los modelos entrenados en el contexto de heridas corto-punzantes o suturables. La comparación de los resultados en la tabla 4.2 muestran que nuestro modelo generado para DeepLabv3+, supera en todas las métricas a investigaciones previas en imágenes biomédicas, lo que demuestra su eficacia y robustez para segmentar las regiones de interés de las imágenes. Por otro lado, los modelos U-Net y SegNet presentan métricas inferiores que los estudios previos, lo que sugiere ajustes para mejorar su desempeño en entornos clínicos.

Este estudio representa un avance significativo en el campo de la segmentación de heridas corto-punzantes en tiempo real a través de video en vivo. A diferencia de las investigaciones previas que se centran en imágenes estáticas, nuestro trabajo ha logrado procesar secuencias de video en vivo utilizando una cámara web de alta resolución y un algoritmo de inferencia sobre los frames capturados. Este enfoque permite procesar las imágenes de las heridas en tiempo real, lo que representa un gran desafío técnico y computacional. Hemos optimizado el rendimiento de nuestros modelos de segmentación para reducir el tiempo de inferencia y mejorar la precisión, lo que demuestra la eficacia y robustez de nuestros modelos. Puede observar el registro de la segmentación en vivo con nuestro mejor modelo, DeepLabV3+ con ResNet50 como codificador en este enlace: **Video segmentación**.

La segmentación de heridas en tiempo real tiene una gran relevancia para el desarrollo de sistemas de sutura autónoma, una aplicación potencial de la inteligencia artificial en el campo médico. Nuestro trabajo contribuye a este desarrollo al proporcionar un método eficaz para identificar la región de la piel que necesita ser suturada y guiar el movimiento

del instrumento quirúrgico. La sutura autónoma tiene el potencial de reducir los errores humanos, mejorar la calidad de la cicatrización y disminuir los costos y tiempos de operación. Además, en este estudio, se utilizan métricas como Dice, IoU, F1, precisión y recall para medir la consistencia temporal de las predicciones de segmentación en el video en vivo. Estas métricas indican la similitud entre la máscara del frame anterior y la del frame actual, lo que refleja la estabilidad y la robustez de nuestros modelos frente a cambios en la iluminación, el ángulo y el movimiento del objetivo.

A pesar de los resultados prometedores obtenidos a partir del entrenamiento de las tres redes mencionadas y la segmentación de heridas corto-punzantes en tiempo real, es importante reconocer que existen ciertas limitaciones que deben ser consideradas. Estas limitaciones pueden agruparse en dos categorías principales: limitaciones de las redes en sí y limitaciones relacionadas con la infraestructura y recursos computacionales disponibles.

En cuanto a las limitaciones de las redes, uno de los desafíos principales en el entrenamiento de modelos de segmentación semántica es la disponibilidad de un conjunto de datos lo suficientemente grande y diverso. Aunque se construyó un conjunto de datos específico para este estudio, su tamaño podría limitar la capacidad de generalización de los modelos en situaciones clínicas más amplias. Además, la sensibilidad a la calidad de la etiquetación es crucial, ya que errores en la etiquetación pueden afectar negativamente la generalización correcta de los modelos.

Por otro lado, los requerimientos de hardware también son un aspecto importante a considerar. Los modelos de segmentación profunda, como DeepLabV3+ con ResNet50, pueden ser computacionalmente intensivos. Esto implica que su implementación y ejecución pueden requerir hardware con recursos significativos, como GPU de alto rendimiento, lo que puede no estar disponible en todos los entornos clínicos y limita la capacidad para utilizar estos modelos. Además, estos modelos pueden llegar a ser voluminosos en términos de almacenamiento, especialmente cuando se trata de modelos preentrenados.

A pesar de estas limitaciones, los modelos de segmentación de heridas corto punzantes basados en redes neuronales convolucionales representan un avance significativo en la capacidad de automatizar procedimientos quirúrgicos como la sutura. Con un conjunto de datos adecuado que verdaderamente represente la variabilidad de escenarios reales y con suficientes recursos computacionales disponibles, estos modelos pueden desempeñar un papel importante en la asistencia médica al mejorar la precisión y eficiencia de la segmentación de heridas en la piel.

## 4.2. Conclusiones

La respuesta a la pregunta de investigación planteada se basa en los resultados obtenidos de los modelos de Redes Neuronales Convolucionales (CNN) U-Net2D, SegNet y DeepLabV3+ con ResNet50 como codificador, entrenados en un conjunto de datos construido específicamente para este propósito.

Las características del algoritmo de aprendizaje de máquina y procesamiento de imágenes que permiten la segmentación y reconocimiento de heridas corto punzantes fueron las siguientes:

1. **Arquitectura del modelo:** Una estructura encoder-decoder como los modelos U-Net2D, SegNet y DeepLabV3+ demostraron ser efectivos para la tarea de segmentación. Es importante destacar que el modelo DeepLabV3+ con ResNet50 como codificador se destacó como el más preciso en términos de rendimiento de segmentación, aunque U-Net2D mostró una destacable capacidad para preservar detalles finos.

2. **Preprocesamiento de imágenes:** Antes del entrenamiento, las imágenes requieren cierto preprocesamiento, como la normalización y el aumento de datos, para mejorar el rendimiento del modelo.

3. **Entrenamiento supervisado:** Los modelos se entrenaron en un conjunto de datos construido específicamente para la tarea, lo que implica que el algoritmo aprende a segmentar y reconocer heridas corto punzantes a partir de ejemplos etiquetados. Esto sugiere que la extensión del conjunto de datos de entrenamiento, radicará en una mejora en la precisión de segmentación de los modelos.

4. **Segmentación en tiempo real:** El algoritmo es capaz de realizar segmentación en tiempo real a partir de la cámara web, lo que indica que puede procesar y analizar imágenes en vivo con un tiempo de respuesta adecuado.

Estas características fueron fundamentales para el algoritmo y su capacidad para segmentar y reconocer heridas corto punzantes. Sin embargo, es importante tener en cuenta que el rendimiento del algoritmo puede variar dependiendo de varios factores, como la calidad y diversidad del conjunto de datos, los hiperparámetros utilizados durante el entrenamiento y la arquitectura específica del modelo elegido.

En este estudio, se exploraron las CNN para lograr segmentación semántica de heridas corto punzantes. Se entrenaron los modelos U-Net2D, SegNet y DeepLabV3+ con ResNet50 como encoder y de los resultados obtenidos se concluye que:

- El sistema de reconocimiento de heridas corto punzantes desarrollado, logró una precisión de predicción superior al 80 % en condiciones ambientales variables.
- Entre los modelos implementados, DeepLabV3+ logró la mejor precisión de segmentación en términos de generalización, mientras que U-Net2D demostró precisión en detalles finos pero no en generalización. Por otro lado, SegNet fue eficiente en su tiempo de respuesta y en la ubicación espacial de las heridas. Pero menos efectivo en la detección de bordes.
- La calidad y la cantidad de datos de entrenamiento fueron factores críticos para el rendimiento de los modelos. La construcción de un conjunto de datos diverso y etiquetado con precisión es esencial para la correcta generalización de las características en el contexto de aprendizaje automático.
- Este estudio también contribuye al campo de la atención médica al ofrecer una solución automatizada y precisa para segmentación de heridas corto punzantes. Estos modelos pueden ayudar a profesionales de la salud en la evaluación y el tratamiento de heridas.

## 4.3. Trabajos Futuros

- Expandir el conjunto de datos de entrenamiento con mayor variabilidad de los datos, con el objetivo de mejorar el rendimiento real del algoritmo, además de aprovechar la capacidad de modelos como U-Net2D para detectar detalles finos, en el contexto de sutura.
- Aplicar las técnicas desarrolladas para la segmentación de otras patologías cutáneas, como úlceras por presión o quemaduras, ampliando la utilidad clínica de estos modelos.
- Validar clínicamente los modelos con la colaboración de profesionales médicos con el objetivo de lograr la adopción exitosa de estos modelos en entornos médicos reales.
- Entrenar un detector de objetos con el conjunto de datos disponible, para integrarlo con el modelo Segment Anything Model **(SAM)** de Meta, con el objetivo de mejorar exponencialmente la precisión de la segmentación.

# Bibliografía


[1] Y. Mintz and R. Brodie, "Introduction to artificial intelligence in medicine," *Minimally Invasive Therapy and Allied Technologies*, vol. 28, no. 2, pp. 73–81, feb 2019.

[2] N. Médicas. (2016) La visión por computador en la atención de la salud: ¿qué puede ofrecer a los proveedores? [Online]. Available: https://www.noticiasmedicas.es/noticias/la-vision-por-computador-en-la-atencion-de-la-salud-que-puede-ofrecer-a-los-proveedores/

[3] M. Miliard. (2016, Nov.) Ibm watson health, merge launch new personalized imaging tools at rsna. HealtcareITNews. [Online]. Available: https://www.healthcareitnews.com/news/ibm-watson-health-merge-launch-new-personalized-imaging-tools-rsna

[4] J. F. Ávila Tomás, M. A. Mayer-Pujadas, and V. J. Quesada-Varela, "La inteligencia artificial y sus aplicaciones en medicina ii: importancia actual y aplicaciones prácticas," *Atención Primaria*, vol. 53, no. 1, pp. 81–88, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0212656720301463

[5] D. Distefano. (2021, May) Cirugía laparoscópica: cómo la tecnología mejora la técnica quirúrgica. 13/05/21. Cirugía Argentina. Ciudad de Buenos Aires, Argentina. [Online]. Available: https://contenidos.cirugiaargentina.com/blog/cirugia-laparoscopica-como-la-tecnologia-mejora-la-tecnica-quirurgica

[6] C. Martínez Ramos, "Robótica y cirugía laparoscópica," *Cirugía Española*, vol. 80, no. 4, pp. 189–194, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0009739X06709563

[7] P. Caro Allendes, E. Cerda Diaz, C. Rodriguez-Herrera, P. Navarrete Rey, and I. Miranda-Mendoza, "Ergonomia en cirugia laparoscopica ginecologica," *Revista chilena de obstetricia y ginecologia*, vol. 85, pp. 222 – 235, 06 2020. [Online]. Available: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0717-75262020000300222&nrm=iso

[8] H. A. Naffakh, R. Ghazali, N. E. Abbadi, and A. Razzaq, "A review of human skin detection applications based on image processing," *Bulletin of Electrical*

*Engineering and Informatics*, vol. 10, no. 1, pp. 129–137, 2021. [Online]. Available: https://beei.org/index.php/EEI/article/view/2497

[9] I. de ciencias de la computacion. (2016) Imagenes y robotica. ICC. Pabellón Cero Infinito – Ciudad Universitaria. [Online]. Available: https://icc.fcen.uba.ar/imagenes-y-robotica/

[10] J. Pelegri. (2019, May) Ventajas y aplicaciones de visión arificial en los robots colaborativos. 22/05/19. Universal Robots. 08019 Barcelona España. [Online]. Available: https://www.universal-robots.com/es/blog/vision-artificial-en-robots/

[11] T. Sun, M. Liu, H. Ye, and D.-Y. Yeung, "Point-cloud-based place recognition using cnn feature extraction," *IEEE Sensors Journal*, vol. 19, no. 24, pp. 12 175–12 186, 2019.

[12] G. Du, X. Cao, J. Liang, X. Chen, and Y. Zhan, "Medical image segmentation based on u-net: A review," *Journal of Imaging Science and Technology*, vol. 64, 03 2020.

[13] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE Access*, vol. 9, pp. 82 031–82 057, 2021.

[14] R. Azad, E. K. Aghdam, A. Rauland, Y. Jia, A. H. Avval, A. Bozorgpour, S. Karimijafarbigloo, J. P. Cohen, E. Adeli, and D. Merhof, "Medical image segmentation review: The success of u-net," *arXiv preprint arXiv:2211.14830*, 2022.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[16] D. Gupta, "Image segmentation keras : Implementation of segnet, fcn, unet, pspnet and other models in keras," *arXiv preprint arxiv.org/abs/2307.13215*, 2023.

[17] T. Zhou, Y. Liu, H. Lu, X. Ye, and X. Chang, "Resnet and its application to medical image processing: Research progress and challenges," *Journal of Electronics and Information Technology*, vol. 44, no. 1, pp. 149–167, 2022. [Online]. Available: https://jeit.ac.cn/cn/article/doi/10.11999/JEIT210914

[18] A. Saha, Y.-D. Zhang, and S. C. Satapathy, "Brain tumour segmentation with a muti-pathway resnet based unet," *Journal of Grid Computing*, vol. 19, no. 1, p. 43, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10723-021-09590-y

[19] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 5168–5177. [Online]. Available: https://arxiv.org/abs/1611.06612

[20] K. Fu, Q. Zhao, and I. Y.-H. Gu, "Refinet: A deep segmentation assisted refinement network for salient object detection," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 457–469, Feb 2019.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[22] R. Anantharaman, M. Velazquez, and Y. Lee, "Utilizing mask r-cnn for detection and segmentation of oral diseases," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec 2018, pp. 2197–2204.

[23] Z. Zhang, Y. Li, S. Wang, X. Zhang, J. Zhang, Y. Zhang, Y. Wang, and J. Zhang, "Automatic lung nodule segmentation based on mask r-cnn," *Journal of Digital Imaging*, vol. 34, no. 4, pp. 1026–1035, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10278-021-00490-x

[24] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017.

[25] N. Saxena, K. B. N., and B. Raman, "Semantic segmentation of multispectral images using res-seg-net model," in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, Feb 2020, pp. 154–157.

[26] J. Chen, J. Lu, X. Zhu, and L. Zhang, "Generative Semantic Segmentation," 2023. [Online]. Available: http://arxiv.org/abs/2303.11316

[27] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and Hartwig, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018. [Online]. Available: https://paperswithcode.com/lib/detectron2/deeplabv3-1#

[28] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 833–851.

[29] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982. [Online]. Available: https://www.pnas.org

[30] F. Izaurieta and C. Saavedra, "Redes Neuronales Artificiales," *Charlas de fisica*, pp. 1–15, 1999.

[31] S. Jeswal and S. Chakraverty, "Chapter 10 - fuzzy eigenvalue problems of structural dynamics using ann," in *New Paradigms in Computational*

*Modeling and Its Applications*, S. Chakraverty, Ed. Academic Press, 2021, pp. 145–161. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128221334000104

[32] V. Jana. (2023) Potential action. Acceso: 19 septiembre 2023. [Online]. Available: https://www.kenhub.com/en/library/anatomy/action-potential

[33] P. Cheeseman and W. Gevarter, "Introduction to artificial inteligence." in *AIAA Paper*, 1986, pp. 109–128. [Online]. Available: www.springer.com/series/7592

[34] C. Camelot, "The unit that makes neural networks neural: Perceptron," 2022. [Online]. Available: https://blog.camelot-group.com/2022/01/neural-networks-perceptron/

[35] J. Martín and A. Serrano, "Problema 1: Resolución del problema XOR." 2010. [Online]. Available: https://studylib.es/doc/4814380/problema-1--resoluci%C3%B3n-del-problema-xor.-problema-2---ocw-uv

[36] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[37] J. Markus and S. Balaji, "Convolutional neural networks (cnns): concepts and applications in pharmacogenomics," *Molecular Diversity*, vol. 25, no. 3, pp. 1569–1584, 2021.

[38] B. Astuto, I. Flament, N. K. Namiri, R. Shah, U. Bharadwaj, T. M. Link, M. D. Bucknor, V. Pedoia, and S. Majumdar, "Automatic deep learning–assisted detection and grading of abnormalities in knee mri studies," *Radiology: Artificial Intelligence*, vol. 3, no. 3, p. e200165, 2021.

[39] R. Singh *et al.*, "Image quality and lesion detection on deep learning reconstruction (dlr) and iterative reconstruction (ir) images of submillisievert chest and abdominopelvic ct," *AJR Am J Roentgenol*, vol. 214, no. 2, pp. 319–328, 2020.

[40] DataScientest. (2021) Convolutional neural network: definición y funcionamiento. [Online]. Available: https://datascientest.com/es/convolutional-neural-network-es

[41] B. Juan. (2019, 2) Understanding semantic segmentation with unet. [Online]. Available: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[42] O. Toro and C. Antonio, "Algoritmos de segmentación semántica para anotación de imágenes," 2019, no Publicado. [Online]. Available: https://oa.upm.es/55407/

[43] A. Gautam and H. Bhadauria, "Classification of white blood cells based on morphological features," *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, no. May, pp. 2363–2368, 2014.

[44] S. Mircic and N. Jorgovanovic, "Automatic classification of leukocytes," *Journal of Automatic Control*, vol. 16, no. 1, pp. 29–32, 2006.

[45] H. Lamba. (2019, 2) Understanding semantic segmentation with unet. [Online]. Available: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[46] T. Anwar. (2020, 12) Classification with localization: Convert any keras classifier to a detector. [Online]. Available: https://learnopencv.com/classification-with-localization/

[47] MathWorks. (2020) Procesamiento de imágenes y visión artificial. [Online]. Available: https://www.mathworks.com/videos/image-processing-and-computer-vision-with-matlab-1597884648964.html

[48] R. Rodríguez. (2018) Segmentación de instancia. [Online]. Available: https://lamaquinaoraculo.com/computacion/segmentacion-de-instancia-mask-r-cnn/

[49] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "PolarMask: Single shot instance segmentation with polar representation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 12 190–12 199, 2020.

[50] S. Naoki. (2018, 11) Up-sampling with transposed convolution. [Online]. Available: https://zzsza.github.io/data/2018/06/25/upsampling-with-transposed-convolution/

[51] V. Dumoulin and F. Visin, "Object detection : A guide to convolution arithmetic for deep learning," *arXiv:1603.07285 [cs, stat]*, pp. 1–31, 2018.

[52] K. Wei, "Understand transposed convolutions," 2020. [Online]. Available: https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967

[53] Z. Villa, M. Dominguez, and S. Aljama. (2017, 8) Cuidados de enfermería en las heridas suturables. [Online]. Available: https://www.revista-portalesmedicos.com/revista-medica/cuidados-de-enfermeria-heridas-suturables/

[54] V. N. Shenoy, E. Foster, L. Aalami, B. Majeed, and O. Aalami, "Deepwound: Automated Postoperative Wound Assessment and Surgical Site Surveillance through Convolutional Neural Networks," *Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018*, pp. 1017–1021, 2019.

[55] P. Arya and U. Shankar Modani, "Applications of Artificial Neural Network in Image Processing: A Survey," *International Journal of Scientific and Engineering Research*, pp. 329–333, 2019. [Online]. Available: http://www.ijser.org

[56] D. M. Anisuzzaman, C. Wang, B. Rostami, S. Gopalakrishnan, J. Niezgoda, and Z. Yu, "Image-Based Artificial Intelligence in Wound Assessment: A Systematic

Review," *Advances in Wound Care*, vol. 11, no. 12, pp. 687–709, 2022.

[57] B. Carpenter, J. G. Surles, and S. N. Lahiri, "Google colaboratory as a platform for teaching statistics and data science in the cloud," *The American Statistician*, vol. 74, no. 1, p. 55–63, 2020.

[58] H. Singh, "Pytorch vs. tensorflow for deep learning in 2023," *Analytics India Magazine*, 2021. [Online]. Available: https://analyticsindiamag.com/pytorch-vs-tensorflow-for-deep-learning-in-2023/

[59] L. D. Nguyen, R. Gao, D. Lin, and Z. Lin, "Biomedical image classification based on a feature concatenation and ensemble of deep cnns," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2019.

[60] V. Kumar, M. A. Saleem, N. Senan, F. Wahid, M. Aamir, A. Samad, and M. Khan, "Comparative analysis of recent architecture of convolutional neural network," *Mathematical Problems in Engineering*, vol. 2022, p. 7313612, 03 2022. [Online]. Available: https://doi.org/10.1155/2022/7313612

[61] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[62] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[63] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A review of deep learning in medical image segmentation," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[64] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," in *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4. IEEE, 2017, p. 834–848.

[65] Z. Liu, Y. Liu, Z. Wang, Y. Wang, Y. Zhang, and Y. Zhang, "Deep learning-based wound segmentation using a modified deeplabv3+ model with a squeeze-and-excitation module," *Sensors*, vol. 21, no. 2, p. 409, 2021.

# Apéndice A

# Anexos

## A.1. Documentación software

Para acceder a los recursos usados en este proyecto (ver figura A.2) siga las instrucciones del diagrama A.1, a continuación:

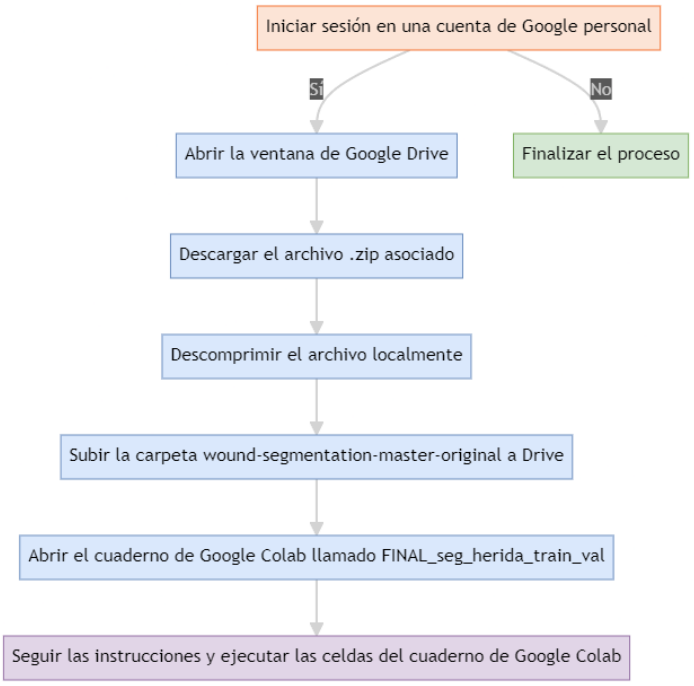



Figura A.1: Preparación del proyecto de segmentación semántica

Ingresa el siguiente enlace en tu navegador o haz clic aquí: *Repositorio Drive* (`https:`

//drive.google.com/drive/folders/1FVyBf5GsGiGl6G6IVU_4DyppfDU8cDMQ?usp=sharing)

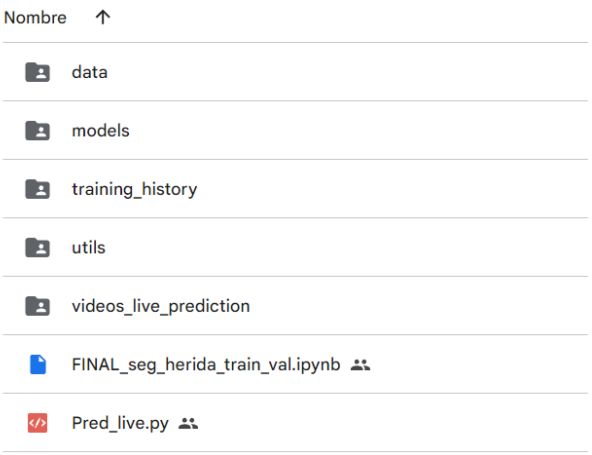



Figura A.2: Repositorio del proyecto de segmentación semántica

Si se encuentra una sesión activa en una cuenta de Google personal, se abrirá una ventana de Google Drive, que contiene la carpeta `wound-segmentation-master-original` con todas las dependencias asociadas al proyecto en su interior. Descargue el archivo .zip asociado, como observa en la figura: A.3.

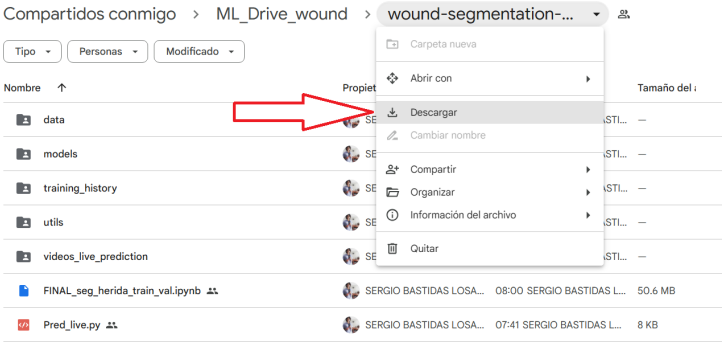



Figura A.3: Descarga del proyecto de segmentación semántica

Se recomienda descomprimir el archivo localmente y posteriormente subir la carpeta `wound-segmentation-master-original` a su cuenta de Drive personal para que pueda empezar con el entrenamiento de las CNN y aprovechar los recursos computacionales que asigna Google a su sesión de Colab. Tenga en cuenta la ruta en la cual alojará el proyecto, pues será necesaria para la carga de los datos en el cuaderno de Google Colab. Si desea mantener la nomenclatura del proyecto original, cree una carpeta en su unidad de Drive llamada `ML_Drive_wound` y allí suba la carpeta `wound-segmentation-master-original` resultante de la descompresión.

Una vez la carpeta de proyecto master se encuentra en su cuenta de Google Drive, a continuación, puede abrir el Cuaderno de Google Colab llamado `FINAL_seg_herida_train_val`, se abrirá una ventana como la que observa en la figura A.4. Solamente resta seguir las instrucciones y ejecución secuencial de cada una de las celdas.

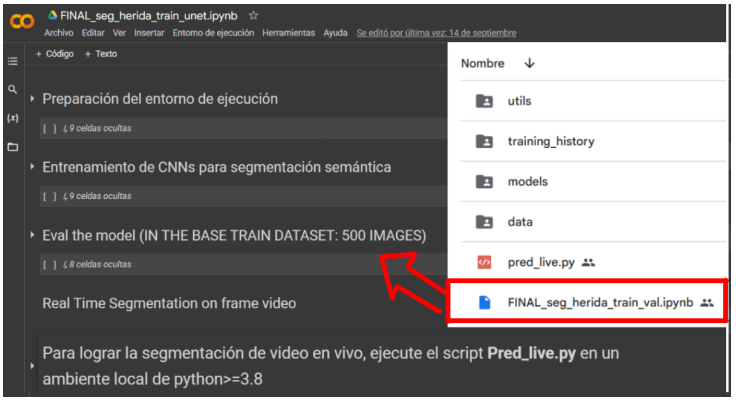



Figura A.4: Cuaderno de Colab usado para entrenamiento-validación

## A.2. Documentación de los modelos

La tabla A.1 muestra el número de parámetros de cada uno de los modelos analizados en esta investigación:

| **Parámetros** | **U-Net2D** | **SegNet** | **DeepLabV3+** |
|---|---|---|---|
| **Total params:** | 1,209,807 | 87,009 | 17,869,697 |
| **Trainable params:** | 1,209,807 | 87,009 | 17,834,913 |
| **Non-trainable params:** | 0 | 0 | 34,784 |

Tabla A.1: Parámetros de los modelos seleccionados

La tabla A.2 presenta el resumen de los parámetros de modelo U-Net2D implementado. Esta implementación está inspirada en el trabajo de Qing Yuan en *ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection*, adaptado a las necesidades del proyecto.

| **Layer (type)** | **Output Shape** | `Param #` | **Connected to** |
|---|---|---|---|
| input_1 (InputLayer) | (None, 224, 224, 3) | 0 | [] |
| Continúa en la siguiente página | | | |

Tabla A.2 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | `Param #` | **Connected to** |
|---|---|---|---|
| conv2d | (None, 224, 224, 32) | 896 | ['input_1'] |
| conv2d_1 | (None, 224, 224, 32) | 9248 | ['conv2d'] |
| max_pooling2d | (None, 112, 112, 32) | 0 | ['conv2d_1'] |
| conv2d_2 | (None, 112, 112, 64) | 18496 | ['max_pooling2d'] |
| conv2d_3 | (None, 112, 112, 64) | 36928 | ['conv2d_2'] |
| max_pooling2d_1 | (None, 56, 56, 64) | 0 | ['conv2d_3'] |
| conv2d_4 | (None, 56, 56, 128) | 73856 | ['max_pooling2d_1'] |
| conv2d_5 | (None, 56, 56, 128) | 147584 | ['conv2d_4'] |
| conv2d_6 | (None, 56, 56, 128) | 147584 | ['conv2d_5'] |
| max_pooling2d_2 | (None, 28, 28, 128) | 0 | ['conv2d_6'] |
| conv2d_7 | (None, 28, 28, 256) | 295168 | ['max_pooling2d_2'] |
| conv2d_8 | (None, 28, 28, 256) | 590080 | ['conv2d_7'] |
| conv2d_9 | (None, 28, 28, 256) | 590080 | ['conv2d_8'] |
| max_pooling2d_3 | (None, 14, 14, 256) | 0 | ['conv2d_9'] |
| conv2d_10 | (None, 14, 14, 256) | 590080 | ['max_pooling2d_3'] |
| conv2d_11 | (None, 14, 14, 256) | 590080 | ['conv2d_10'] |
| conv2d_12 | (None, 14, 14, 256) | 590080 | ['conv2d_11'] |
| up_sampling2d | (None, 28, 28, 256) | 0 | ['conv2d_12'] |
| conv2d_14 | (None, 28, 28, 128) | 295040 | ['conv2d_9'] |
| conv2d_13 | (None, 28, 28, 128) | 131200 | ['up_sampling2d'] |
| concatenate | (None, 28, 28, 256) | 0 | ['conv2d_14', 'conv2d_13'] |
| conv2d_15 | (None, 28, 28, 128) | 295040 | ['concatenate'] |
| conv2d_16 | (None, 28, 28, 128) | 147584 | ['conv2d_15'] |
| up_sampling2d_1 | (None, 56, 56, 128) | 0 | ['conv2d_16'] |
| conv2d_18 | (None, 56, 56, 64) | 73792 | ['conv2d_6'] |
| conv2d_17 | (None, 56, 56, 64) | 32832 | ['up_sampling2d_1'] |
| concatenate_1 | (None, 56, 56, 128) | 0 | ['conv2d_18', 'conv2d_17'] |
| conv2d_19 | (None, 56, 56, 64) | 73792 | ['concatenate_1'] |
| conv2d_20 | (None, 56, 56, 64) | 36928 | ['conv2d_19'] |
| up_sampling2d_2 | (None, 112, 112, 64) | 0 | ['conv2d_20'] |
| conv2d_22 | (None, 112, 112, 32) | 18464 | ['conv2d_3'] |
| conv2d_21 | (None, 112, 112, 32) | 8224 | ['up_sampling2d_2'] |

Continúa en la siguiente página

Tabla A.2 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | `Param #` | **Connected to** |
|---|---|---|---|
| concatenate_2 | (None, 112, 112, 64) | 0 | ['conv2d_22', 'conv2d_21'] |
| conv2d_23 | (None, 112, 112, 32) | 18464 | ['concatenate_2'] |
| conv2d_24 | (None, 112, 112, 32) | 9248 | ['conv2d_23'] |
| up_sampling2d_3 | (None, 224, 224, 32) | 0 | ['conv2d_24'] |
| conv2d_26 | (None, 224, 224, 16) | 4624 | ['conv2d_1'] |
| conv2d_25 | (None, 224, 224, 16) | 2064 | ['up_sampling2d_3'] |
| concatenate_3 | (None, 224, 224, 32) | 0 | ['conv2d_26', 'conv2d_25'] |
| conv2d_27 | (None, 224, 224, 16) | 4624 | ['concatenate_3'] |
| conv2d_28 | (None, 224, 224, 16) | 2320 | ['conv2d_27'] |
| conv2d_29 | (None, 224, 224, 3) | 435 | ['conv2d_28'] |
| conv2d_30 | (None, 224, 224, 1) | 4 | ['conv2d_29'] |

Tabla A.2: Parámetros de la red U-Net2D

La tabla A.3 presenta el resumen de las capas del modelo DeepLabV3+ con ResNet50 como backbone, esta implementación está inspirada en la competencia de Kaggle *Understanding Clouds from Satellite Images*, adaptado a las necesidades del proyecto.

| **Layer (type)** | **Output Shape** | **Connected to** |
|---|---|---|
| input_1 | [(None,224,224,3)] | [] |
| conv1_pad | (None,230,230,3) | ['input_1'] |
| conv1_conv | (None,112,112,64) | ['conv1_pad'] |
| conv1_bn | (None,112,112,64) | ['conv1_conv'] |
| conv1_relu | (None,112,112,64) | ['conv1_bn'] |
| pool1_pad | (None,114,114,64) | ['conv1_relu'] |
| pool1_pool | (None,56,56,64) | ['pool1_pad'] |
| conv2_block1_1_conv | (None,56,56,64) | ['pool1_pool'] |
| conv2_block1_1_bn | (None,56,56,64) | ['conv2_block1_1_conv'] |
| conv2_block1_1_relu | (None,56,56,64) | ['conv2_block1_1_bn'] |
| conv2_block1_2_conv | (None,56,56,64) | ['conv2_block1_1_relu'] |
| conv2_block1_2_bn | (None,56,56,64) | ['conv2_block1_2_conv'] |
| conv2_block1_2_relu | (None,56,56,64) | ['conv2_block1_2_bn'] |

Tabla A.3 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | **Connected to** |
|---|---|---|
| conv2_block1_0_conv | (None,56,56,256) | ['pool1_pool'] |
| conv2_block1_3_conv | (None,56,56,256) | ['conv2_block1_2_relu'] |
| conv2_block1_0_bn | (None,56,56,256) | ['conv2_block1_0_conv'] |
| conv2_block1_3_bn | (None,56,56,256) | ['conv2_block1_3_conv'] |
| conv2_block1_add | (None,56,56,256) | ['conv2_block1_0_bn','conv2_block1_3_bn'] |
| conv2_block1_out | (None,56,56,256) | ['conv2_block1_add'] |
| conv2_block2_1_conv | (None,56,56,64) | ['conv2_block1_out'] |
| conv2_block2_1_bn | (None,56,56,64) | ['conv2_block2_1_conv'] |
| conv2_block2_1_relu | (None,56,56,64) | ['conv2_block2_1_bn'] |
| conv2_block2_2_conv | (None,56,56,64) | ['conv2_block2_1_relu'] |
| conv2_block2_2_bn | (None,56,56,64) | ['conv2_block2_2_conv'] |
| conv2_block2_2_relu | (None,56,56,64) | ['conv2_block2_2_bn'] |
| conv2_block2_3_conv | (None,56,56,256) | ['conv2_block2_2_relu'] |
| conv2_block2_3_bn | (None,56,56,256) | ['conv2_block2_3_conv'] |
| conv2_block2_add | (None,56,56,256) | ['conv2_block1_out','conv2_block2_3_bn'] |
| conv2_block2_out | (None,56,56,256) | ['conv2_block2_add'] |
| conv2_block3_1_conv | (None,56,56,64) | ['conv2_block2_out'] |
| conv2_block3_1_bn | (None,56,56,64) | ['conv2_block3_1_conv'] |
| conv2_block3_1_relu | (None,56,56,64) | ['conv2_block3_1_bn'] |
| conv2_block3_2_conv | (None,56,56,64) | ['conv2_block3_1_relu'] |
| conv2_block3_2_bn | (None,56,56,64) | ['conv2_block3_2_conv'] |
| conv2_block3_2_relu | (None,56,56,64) | ['conv2_block3_2_bn'] |
| conv3_block3_3_conv | (None,28,28,512) | ['conv3_block3_2_relu'] |
| conv3_block3_3_bn | (None,28,28,512) | ['conv3_block3_3_conv'] |
| conv3_block3_add | (None,28,28,512) | ['conv3_block2_out','conv3_block3_3_bn'] |
| conv3_block3_out | (None,28,28,512) | ['conv3_block3_add'] |
| conv3_block4_1_conv | (None,28,28,128) | ['conv3_block3_out'] |
| conv3_block4_1_bn | (None,28,28,128) | ['conv3_block4_1_conv'] |
| conv3_block4_1_relu | (None,28,28,128) | ['conv3_block4_1_bn'] |
| conv3_block4_2_conv | (None,28,28,128) | ['conv3_block4_1_relu'] |
| conv3_block4_2_bn | (None,28,28,128) | ['conv3_block4_2_conv'] |
| conv3_block4_2_relu | (None,28,28,128) | ['conv3_block4_2_bn'] |
| conv3_block4_3_conv | (None,28,28,512) | ['conv3_block4_2_relu'] |

Tabla A.3 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | **Connected to** |
|---|---|---|
| conv3_block4_3_bn | (None,28,28,512) | ['conv3_block4_3_conv'] |
| conv3_block4_add | (None,28,28,512) | ['conv3_block3_out','conv3_block4_3_bn'] |
| conv3_block4_out | (None,28,28,512) | ['conv3_block4_add'] |
| conv4_block1_1_conv | (None,14,14,256) | ['conv3_block4_out'] |
| conv4_block1_1_bn | (None,14,14,256) | ['conv4_block1_1_conv'] |
| conv4_block1_1_relu | (None,14,14,256) | ['conv4_block1_1_bn'] |
| conv4_block1_2_conv | (None,14,14,256) | ['conv4_block1_1_relu'] |
| conv4_block1_2_bn | (None,14,14,256) | ['conv4_block1_2_conv'] |
| conv4_block1_2_relu | (None,14,14,256) | ['conv4_block1_2_bn'] |
| conv4_block1_0_conv | (None,14,14,1024) | ['conv3_block4_out'] |
| conv4_block1_3_conv | (None,14,14,1024) | ['conv4_block1_2_relu'] |
| conv4_block1_0_bn | (None,14,14,1024) | ['conv4_block1_0_conv'] |
| conv4_block1_3_bn | (None,14,14,1024) | ['conv4_block1_3_conv'] |
| conv4_block1_add | (None,14,14,1024) | ['conv4_block1_0_bn','conv4_block1_3_bn'] |
| conv4_block1_out | (None,14,14,1024) | ['conv4_block1_add'] |
| conv4_block2_1_conv | (None,14,14,256) | ['conv4_block1_out'] |
| conv4_block2_1_bn | (None,14,14,256) | ['conv4_block2_1_conv'] |
| conv4_block2_1_relu | (None,14,14,256) | ['conv4_block2_1_bn'] |
| conv4_block2_2_conv | (None,14,14,256) | ['conv4_block2_1_relu'] |
| conv4_block2_2_bn | (None,14,14,256) | ['conv4_block2_2_conv'] |
| conv4_block2_2_relu | (None,14,14,256) | ['conv4_block2_2_bn'] |
| conv4_block2_3_conv | (None,14,14,1024) | ['conv4_block2_2_relu'] |
| conv4_block2_3_bn | (None,14,14,1024) | ['conv4_block2_3_conv'] |
| conv4_block2_add | (None,14,14,1024) | ['conv4_block1_out','conv4_block2_3_bn'] |
| conv4_block2_out | (None,14,14,1024) | ['conv4_block2_add'] |
| conv4_block3_1_conv | (None,14,14,256) | ['conv4_block2_out'] |
| conv4_block3_1_bn | (None,14,14,256) | ['conv4_block3_1_conv'] |
| conv4_block3_1_relu | (None,14,14,256) | ['conv4_block3_1_bn'] |
| conv4_block3_2_conv | (None,14,14,256) | ['conv4_block3_1_relu'] |
| conv4_block3_2_bn | (None,14,14,256) | ['conv4_block3_2_conv'] |
| conv4_block3_2_relu | (None,14,14,256) | ['conv4_block3_2_bn'] |
| conv4_block3_3_conv | (None,14,14,1024) | ['conv4_block3_2_relu'] |
| conv4_block3_3_bn | (None,14,14,1024) | ['conv4_block3_3_conv'] |

Tabla A.3 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | **Connected to** |
|---|---|---|
| conv4_block3_add | (None,14,14,1024) | ['conv4_block2_out','conv4_block3_3_bn'] |
| conv4_block3_out | (None,14,14,1024) | ['conv4_block3_add'] |
| conv4_block4_1_conv | (None,14,14,256) | ['conv4_block3_out'] |
| conv4_block4_1_bn | (None,14,14,256) | ['conv4_block4_1_conv'] |
| conv4_block4_1_relu | (None,14,14,256) | ['conv4_block4_1_bn'] |
| conv4_block4_2_conv | (None,14,14,256) | ['conv4_block4_1_relu'] |
| conv4_block4_2_bn | (None,14,14,256) | ['conv4_block4_2_conv'] |
| conv4_block4_2_relu | (None,14,14,256) | ['conv4_block4_2_bn'] |
| conv4_block4_3_conv | (None,14,14,1024) | ['conv4_block4_2_relu'] |
| conv4_block4_3_bn | (None,14,14,1024) | ['conv4_block4_3_conv'] |
| conv4_block4_add | (None,14,14,1024) | ['conv4_block3_out','conv4_block4_3_bn'] |
| conv4_block4_out | (None,14,14,1024) | ['conv4_block4_add'] |
| conv4_block5_1_conv | (None,14,14,256) | ['conv4_block4_out'] |
| conv4_block5_1_bn | (None,14,14,256) | ['conv4_block5_1_conv'] |
| conv4_block5_1_relu | (None,14,14,256) | ['conv4_block5_1_bn'] |
| conv4_block5_2_conv | (None,14,14,256) | ['conv4_block5_1_relu'] |
| conv4_block5_2_bn | (None,14,14,256) | ['conv4_block5_2_conv'] |
| conv4_block5_2_relu | (None,14,14,256) | ['conv4_block5_2_bn'] |
| conv4_block5_3_conv | (None,14,14,1024) | ['conv4_block5_2_relu'] |
| conv4_block5_3_bn | (None,14,14,1024) | ['conv4_block5_3_conv'] |
| conv4_block5_add | (None,14,14,1024) | ['conv4_block4_out','conv4_block5_3_bn'] |
| conv4_block5_out | (None,14,14,1024) | ['conv4_block5_add'] |
| conv4_block6_1_conv | (None,14,14,256) | ['conv4_block5_out'] |
| conv4_block6_1_bn | (None,14,14,256) | ['conv4_block6_1_conv'] |
| conv4_block6_1_relu | (None,14,14,256) | ['conv4_block6_1_bn'] |
| conv4_block6_2_conv | (None,14,14,256) | ['conv4_block6_1_relu'] |
| conv4_block6_2_bn | (None,14,14,256) | ['conv4_block6_2_conv'] |
| conv4_block6_2_relu | (None,14,14,256) | ['conv4_block6_2_bn'] |
| conv4_block6_3_conv | (None,14,14,1024) | ['conv4_block6_2_relu'] |
| conv4_block6_3_bn | (None,14,14,1024) | ['conv4_block6_3_conv'] |
| conv4_block6_add | (None,14,14,1024) | ['conv4_block5_out','conv4_block6_3_bn'] |
| conv4_block6_out | (None,14,14,1024) | ['conv4_block6_add'] |
| average_pooling2d | (None,1,1,1024) | ['conv4_block6_out'] |

Tabla A.3 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | **Connected to** |
|---|---|---|
| conv2d | (None,1,1,256) | ['average_pooling2d'] |
| batch_normalization | (None,1,1,256) | ['conv2d'] |
| conv2d_1 | (None,14,14,256) | ['conv4_block6_out'] |
| conv2d_2 | (None,14,14,256) | ['conv4_block6_out'] |
| conv2d_3 | (None,14,14,256) | ['conv4_block6_out'] |
| conv2d_4 | (None,14,14,256) | ['conv4_block6_out'] |
| activation | (None,1,1,256) | ['batch_normalization'] |
| batch_normalization_1 | (None,14,14,256) | ['conv2d_1'] |
| batch_normalization_2 | (None,14,14,256) | ['conv2d_2'] |
| batch_normalization_3 | (None,14,14,256) | ['conv2d_3'] |
| batch_normalization_4 | (None,14,14,256) | ['conv2d_4'] |
| up_sampling2d | (None,14,14,256) | ['activation'] |
| activation_1 | (None,14,14,256) | ['batch_normalization_1'] |
| activation_2 | (None,14,14,256) | ['batch_normalization_2'] |
| activation_3 | (None,14,14,256) | ['batch_normalization_3'] |
| activation_4 | (None,14,14,256) | ['batch_normalization_4'] |
| concatenate | (None,14,14,1280) | ['up_sampling2d','activation(1,2,3,4)] |
| conv2d_5 | (None,14,14,256) | ['concatenate'] |
| batch_normalization_5 | (None,14,14,256) | ['conv2d_5'] |
| conv2d_6 | (None,56,56,48) | ['conv2_block2_out'] |
| activation_5 | (None,14,14,256) | ['batch_normalization_5'] |
| batch_normalization_6 | (None,56,56,48) | ['conv2d_6'] |
| up_sampling2d_1 | (None,56,56,256) | ['activation_5'] |
| activation_6 | (None,56,56,48) | ['batch_normalization_6'] |
| concatenate_1 | (None,56,56,304) | ['up_sampling2d_1','activation_6'] |
| global_average_pooling2d | (None,304) | ['concatenate_1'] |
| reshape | (None,1,1,304) | ['global_average_pooling2d'] |
| dense | (None,1,1,38) | ['reshape'] |
| dense_1 | (None,1,1,304) | ['dense'] |
| tf.math.multiply | (None,56,56,304) | ['concatenate_1','dense_1'] |
| conv2d_7 | (None,56,56,256) | ['tf.math.multiply'] |
| batch_normalization_7 | (None,56,56,256) | ['conv2d_7'] |
| activation_7 | (None,56,56,256) | ['batch_normalization_7'] |

Tabla A.3 – Continuación de la página anterior

| **Layer (type)** | **Output Shape** | **Connected to** |
|---|---|---|
| conv2d_8 | (None,56,56,256) | ['activation_7'] |
| batch_normalization_8 | (None,56,56,256) | ['conv2d_8'] |
| activation_8 | (None,56,56,256) | ['batch_normalization_8'] |
| global_average_pooling2d1 | (None,256) | ['activation_8'] |
| reshape_1 | (None,1,1,256) | ['global_average_pooling2d1'] |
| dense_2 | (None,1,1,32) | ['reshape_1'] |
| dense_3 | (None,1,1,256) | ['dense_2'] |
| tf.math.multiply_1 | (None,56,56,256) | ['activation_8','dense_3'] |
| up_sampling2d_2 | (None,224,224,256) | ['tf.math.multiply_1'] |
| conv2d_9 | (None,224,224,1) | ['up_sampling2d_2'] |
| activation_9 | (None,224,224,1) | ['conv2d_9'] |

Tabla A.3: Parámetros de la red DeepLabV3+ con ResNet50

## A.3. Origen de los datos de entrenamiento

La tabla A.4 recopila las fuentes de los 349 datos recuperados de la web y que forman parte del conjunto de entrenamiento base empleado en esta investigación:

| **Fuente** | **Enlace** | **Descripción** |
|---|---|---|
| YouTube - Dr. Thomas McClellan | Enlace 1, Enlace 2, Enlace 3, Enlace 4 | Aprovechando la colección de videos del Dr. Thomas McClellan sobre cirugía en vivo, donde explica cómo realizar una sutura en la piel y ejemplifica distintos tipos como sub cuticular y simple. Se lograron obtener tomas de distintos ángulos, de heridas abiertas reales y en un entorno quirúrgico. |

Tabla A.4 – Continuación de la página anterior

| **Fuente** | **Enlace** | **Descripción** |
|---|---|---|
| YouTube - Dr. Zenn | Enlace 5 | Sirviéndose del vídeo donde se presentan las mejores técnicas básicas de sutura con el cirujano plástico certificado y mundialmente reconocido Dr. Michael Zenn. Se logró recopilar varias tomas de heridas durante su proceso de sutura, esto asegura un ambiente quirúrgico uniforme. |
| YouTube - RichardsonsFaceHospitals | Enlace 6 | Gracias a este breve video que muestra cómo se debe realizar precisamente la sutura interrumpida simple para obtener finalmente buenos resultados de cicatrización. Se recopilaron capturas del procedimiento de sutura en la zona facial de un paciente, estos datos aportan variabilidad a la recopilación, abarcando más zonas del cuerpo con heridas. |
| YouTube - Playlist | Enlace 7 | Esta playlist llamada ‘ Primeros Auxilios - Aprende a salvar vidas con la Cruz Roja’, muestra distintos tutoriales de primeros auxilios para distintos tipos de lesión. Se pudo obtener un buen número de tomas, de distintos ángulos, y en condiciones normales fuera de un ambiente quirúrgico. |
| YouTube - EFE Salud | Enlace 8, Enlace 9 | Gracias a este canal se logró aumentar la variabilidad de los datos recopilados, pues en primer lugar proporcionó diferentes tomas de heridas abiertas, simples y graves, haciendo uso del un maniquí de entrenamiento médico. Además, también se obtuvieron algunas tomas de heridas reales muy profundas, durante el procedimiento quirúrgico de prótesis para la articulación del codo. |

Tabla A.4 – Continuación de la página anterior

| **Fuente** | **Enlace** | **Descripción** |
|---|---|---|
| Kaggle - yasinpratomo/wound-dataset y mohamadtaher/wound-data | Enlace 10, Enlace 11 | Estos conjuntos de datos disponibles en el repositorio público de Kaggle, fueron diseñados para otras tareas de segmentación, pero se lograron extraer algunos datos específicamente de laceraciones por corte. |
| Google Images | Enlace 12 | Adicionalmente se complementaron los hallazgos en el buscador de Google Images, bajo el filtro de 'herida abierta'. |
| Simuladores médicos: MaxPreven, SIMULESC, Laerdal, Simulando, PRANATEC, More Than Simulators, ADARO Shop | Enlace 13, Enlace 14, Enlace 15, Enlace 16, Enlace 17, Enlace 18, Enlace 19 | Gracias a los diferentes simuladores listados, que proporcionan entrenadores para el tratamiento de diferentes lesiones, se filtraron los entrenadores con presencia de laceraciones por corte, aunque son heridas artificiales, aportan a la variabilidad de los datos, en cuento a diferentes ubicaciones de las heridas en el cuerpo humano. |

Tabla A.4: Fuentes de imágenes y videos relacionados con heridas suturables