

UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO



**UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA**

**APLICACIÓN MÓVIL PARA LA CLASIFICACIÓN DE QUEMADURAS
EN PACIENTES BASADA EN MODELOS DE APRENDIZAJE
PROFUNDO**

POR:

IVAN AVENA CARO

TESIS PRESENTADA COMO REQUISITO PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA EN COMPUTACIÓN

DIRECTORA DE TESIS: DRA. GRACIELA MARÍA DE JESÚS RAMÍREZ ALONSO

CHIHUAHUA, CHIH., MÉXICO

12 de junio de 2026



Aplicación Móvil para la Clasificación de Quemaduras en Pacientes Basada en Modelos de Aprendizaje Profundo. Tesis presentada por Ivan Avena Caro como requisito parcial para obtener el grado de Maestro en Ingeniería en Computación, ha sido aprobado y aceptado por:

M.I. Fabián Vinicio Hernández Martínez
Director de la Facultad de Ingeniería

M.I. Rodrigo de la Garza Aguilar
Secretario de Investigación y Posgrado

M.S.I Karina Requena Yáñez
Coordinador(a) Académico

Dra. Graciela María de Jesús Ramírez Alonso
Director(a) de Tesis

Junio 2026

Fecha

COMITÉ

Dra. Graciela María de Jesús Ramírez Alonso
Dr. David Ricardo López Flores
Dr. Fernando Martínez Reyes
Dr. Alain Manzo Martínez



UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

17 de junio de 2026.

ING. IVAN AVENA CARO

Presente. -

En atención a su solicitud relativa al trabajo de tesis para obtener el grado de Maestría en Ingeniería en Computación, nos es grato transcribirle el tema aprobado por esta Dirección, propuesto y dirigido por la directora **Dra. Graciela María de Jesús Ramírez Alonso** para que lo desarrolle como Tesis, con el título **“Aplicación móvil para la clasificación de quemaduras en pacientes basada en modelos de aprendizaje profundo”**.

Índice de Contenido

Agradecimientos

Resumen

Abstract

Índice de tablas

Índice de figuras

Índice de ilustraciones

APLICACIÓN MÓVIL PARA LA CLASIFICACIÓN DE QUEMADURAS EN PACIENTES BASADA EN MODELOS DE APRENDIZAJE PROFUNDO

Capítulo 1: Introducción

1.1 Introducción a la problemática

1.2 Antecedentes y estado del arte

1.3 Definición del problema

1.4 Hipótesis

1.5 Objetivos

1.6 Justificación y originalidad

Capítulo 2: Fundamentación teórica

2.1 Clasificación clínica y profundidad de quemaduras

2.2 Fundamentos de visión por computadora en imágenes médicas

2.3 Modelos supervisados para segmentación

2.4 Métricas de evaluación

2.5 Análisis estadístico

2.6 Despliegue en dispositivos móviles

Capítulo 3: Metodología

3.1 Diseño general del estudio

3.2 Descripción del conjunto de datos

3.3 Preprocesamiento de imágenes

3.4 Modelo supervisado YOLOv11-seg



UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

3.5 Benchmark multi-seed y análisis estadístico
3.6 Recursos computacionales
3.7 Implementación en aplicación móvil
Capítulo 4: Resultados
4.1 Resultados de la experimentación con YOLOv11-seg
4.2 Resultados del benchmark comparativo multi-seed
4.3 Resultados del despliegue en aplicación móvil
4.4 Discusión
Capítulo 5: Conclusiones
Capítulo 6: Trabajo futuro
ANEXOS
Anexo A Curvas de entrenamiento del modelo YOLOm11-seg
Anexo B Resultados cualitativos adicionales de YOLO11m-seg
Anexo C Backlog de la aplicación móvil
C.1 Sprints del sistema
C.2 Backlog de historias de usuario de la aplicación móvil
Anexo D Arquitectura en capas de la aplicación móvil
Anexo E Visualización en video de la aplicación móvil
Anexo F Registros de ejecución del sistema de evaluación de dispositivos móviles (Logcat)
Curriculum Vitae

ATENTAMENTE

"naturam subiecit aliis"

EL DIRECTOR

**M.I. FABIÁN VINICIO HERNÁNDEZ
MARTÍNEZ**

**FACULTAD DE
INGENIERÍA
U.A.CH.**



**SECRETARIO DE INVESTIGACIÓN
Y POSGRADO**

M.I. RODRIGO DE LA GARZA AGUILAR



Agradecimientos

Agradezco primeramente a Dios por permitirme concluir esta etapa académica y profesional.

Agradezco a la Dra. Graciela María de Jesús Ramírez Alonso por la dirección, orientación y apoyo brindado durante el desarrollo de esta investigación.

Asimismo, agradezco a los integrantes del comité de tesis por sus observaciones, recomendaciones y aportaciones realizadas durante este trabajo.

Agradezco especialmente al Dr. Norman Alejandro Rendón Mejía por su colaboración en la obtención de imágenes clínicas y en el proceso de revisión de las anotaciones y máscaras utilizadas en el conjunto de datos.

También agradezco al personal de la Unidad de Quemados del Hospital General de Chihuahua por el apoyo brindado durante el desarrollo de este proyecto.

A la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua, al Laboratorio de Visión y Análisis de Datos y a mis compañeros del laboratorio por las facilidades, el espacio de trabajo y el acompañamiento académico brindado durante mis estudios de posgrado.

Agradezco a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por el apoyo otorgado mediante la beca de posgrado que permitió el desarrollo de esta investigación.

Finalmente, agradezco a mi esposa, hijos y padres por su apoyo, comprensión y motivación durante esta etapa académica y profesional.

Resumen

Las quemaduras representan un problema de salud relevante debido a las complicaciones asociadas con la evaluación clínica de la profundidad de la lesión y la necesidad de una atención oportuna. En este trabajo se desarrolló un sistema basado en aprendizaje profundo para la segmentación y clasificación automática de quemaduras de segundo y tercer grado en imágenes clínicas, considerando además su implementación en dispositivos móviles. Inicialmente, se realizaron experimentos de optimización utilizando diferentes configuraciones del modelo YOLO11m-seg con el objetivo de seleccionar la arquitectura y parámetros de entrenamiento más adecuados. Posteriormente, la configuración seleccionada fue utilizada en un *benchmark* comparativo *multi-seed* frente a arquitecturas de segmentación semántica basadas en CNN, incluyendo U-Net, Attention U-Net y U-Net++. Cada arquitectura fue evaluada mediante 10 corridas independientes utilizando los mismos datos, particiones y semillas predefinidas. La evaluación incluyó métricas globales y métricas por clase, análisis estadístico mediante pruebas no paramétricas de Kruskal-Wallis y comparaciones post-hoc de Dunn con corrección de Bonferroni, además de un análisis de eficiencia computacional considerando parámetros, FLOPs, FPS, latencia y consumo de memoria GPU. Los resultados mostraron que las arquitecturas basadas en U-Net obtuvieron mejores métricas globales de segmentación en Dice e IoU durante el *benchmark multi-seed*. Sin embargo, YOLO11m-seg alcanzó el mayor Recall global (0.8643), además de mantener menor consumo de memoria GPU y tiempos de inferencia competitivos respecto a las arquitecturas de segmentación semántica. Por otra parte, las arquitecturas basadas en U-Net presentaron mejores métricas por clase, particularmente en la diferenciación entre quemaduras de segundo y tercer grado. Finalmente, el modelo YOLO11m-seg seleccionado fue convertido e integrado en una aplicación móvil Android mediante TensorFlow Lite para realizar inferencia local sobre dispositivos reales. Los resultados obtenidos muestran la viabilidad del uso de modelos basados en segmentación de instancias como alternativa eficiente para sistemas móviles de apoyo clínico en el análisis automático de quemaduras.

Palabras clave: quemaduras, segmentación de imágenes médicas, aprendizaje profundo, YOLO11m-seg, segmentación semántica, implementación móvil.

Abstract

Burn injuries represent a significant healthcare problem due to the complications associated with clinical assessment of burn depth and the need for timely medical attention. In this work, a deep learning-based system was developed for the automatic segmentation and classification of second- and third-degree burns from clinical images, also considering its deployment on mobile devices. Initially, optimization experiments were conducted using different YOLO11-seg configurations in order to determine the most suitable architecture and training parameters. Subsequently, the selected configuration was used in a comparative multi-seed benchmark against CNN-based semantic segmentation architectures, including U-Net, Attention U-Net, and U-Net++. Each architecture was evaluated through 10 independent runs using the same dataset, data partitions, and controlled random seeds. The evaluation included global and per-class metrics, statistical analysis using non-parametric Kruskal-Wallis tests and Dunn post-hoc comparisons with Bonferroni correction, as well as a computational efficiency analysis considering parameters, FLOPs, FPS, latency, and GPU memory consumption. The results showed that U-Net-based architectures achieved better global segmentation metrics in terms of Dice and IoU during the multi-seed benchmark. However, YOLO11m-seg obtained the highest global Recall (0.8643), while also maintaining lower GPU memory consumption and competitive inference times compared with semantic segmentation architectures. In contrast, U-Net-based architectures achieved better per-class metrics, particularly for differentiating second- and third-degree burns. Finally, the selected YOLO11m-seg model was converted and integrated into an Android mobile application using TensorFlow Lite to perform on-device inference on real smartphones. The results demonstrate that instance segmentation models such as YOLO11m-seg are feasible for mobile clinical decision-support systems aimed at automatic burn wound analysis, as the model achieved the lowest parameter count (22.36M) and memory consumption (261 MB), making it an efficient deployment alternative.

Keywords: burn wounds, medical image segmentation, deep learning, YOLO11m-seg, semantic segmentation, mobile deployment.

Índice general

Agradecimientos	I
Resumen	II
Abstract	III
Índice de tablas	VIII
Índice de figuras	X
Índice de ilustraciones	XIII
APLICACIÓN MÓVIL PARA LA CLASIFICACIÓN DE QUEMADURAS EN PA- CIENTES BASADA EN MODELOS DE APRENDIZAJE PROFUNDO	1
Capítulo 1: Introducción	1
1.1 Introducción a la problemática	1
1.2 Antecedentes y estado del arte	3
1.2.1 Fundamentos clínicos de las quemaduras	3
1.2.2 Análisis automático de quemaduras mediante visión por computadora	3
1.2.3 Modelos supervisados para segmentación y clasificación de quemaduras	5
1.2.3.1 Arquitecturas CNN para segmentación médica	5
1.2.3.2 Segmentación y clasificación de quemaduras mediante <i>deep learning</i>	7
1.2.3.3 Enfoques basados en YOLO para análisis de quemaduras	10
1.2.3.4 Aplicaciones móviles para apoyo clínico en quemaduras	14
1.2.4 Síntesis del estado del arte y posicionamiento del trabajo	16
1.3 Definición del problema	17
1.4 Hipótesis	17
1.5 Objetivos	18

1.5.1	Objetivo general	18
1.6	Justificación y originalidad	19
Capítulo 2:	Fundamentación teórica	20
2.1	Clasificación clínica y profundidad de quemaduras	20
2.1.1	Estimación de superficie corporal afectada (TBSA)	21
2.1.2	Importancia clínica de la segmentación	22
2.2	Fundamentos de visión por computadora en imágenes médicas	22
2.2.1	Redes neuronales convolucionales (CNNs)	23
2.2.2	Segmentación de imágenes médicas	24
2.2.2.1	Segmentación semántica vs segmentación de instancia	24
2.2.3	Arquitecturas encoder-decoder	25
2.3	Modelos supervisados para segmentación	25
2.3.1	Modelos basados en CNN	25
2.3.2	Enfoques basados en YOLO para segmentación médica	30
2.3.2.1	YOLO para tareas médicas.	30
2.3.3	Funciones de pérdida	33
2.3.4	Ventajas y limitaciones de los modelos supervisados de segmentación	36
2.4	Métricas de evaluación	36
2.4.1	Coeficiente Dice	37
2.4.2	Intersección sobre Unión (IoU)	37
2.4.3	Precisión y exhaustividad (<i>Precision / Recall</i>)	38
2.4.4	Tiempo de inferencia y eficiencia computacional	39
2.5	Análisis estadístico	40
2.5.1	Pruebas no paramétricas (Kruskal-Wallis, Dunn, Bonferroni)	40
2.6	Despliegue en dispositivos móviles	41
Capítulo 3:	Metodología	43
3.1	Diseño general del estudio	43
3.2	Descripción del conjunto de datos	43
3.2.1	Recolección de los datos	43

3.2.2	Estructura y características del dataset	46
3.3	Preprocesamiento de imágenes	46
3.3.1	División del conjunto de datos	47
3.4	Modelo supervisado YOLOv11-seg	51
3.5	Benchmark multi-seed y análisis estadístico	55
3.5.1	Arquitecturas evaluadas	55
3.5.2	Preparación del conjunto de datos para el <i>benchmark</i>	55
3.5.3	Configuración experimental	55
3.5.4	Estrategias de augmentación durante el <i>benchmark</i>	56
3.5.5	Funciones de pérdida	56
3.5.6	Protocolo multi-seed	56
3.5.7	Cálculo de métricas de segmentación	57
3.5.8	Matrices de confusión entre clases	58
3.5.9	Análisis estadístico	59
3.5.10	Benchmark de eficiencia computacional	59
3.6	Recursos computacionales	60
3.7	Implementación en aplicación móvil	61
3.7.1	Enfoque metodológico	62
3.7.2	Arquitectura general del sistema	63
3.7.3	Caso de uso del sistema	64
3.7.4	Pipeline de procesamiento e inferencia	67
Capítulo 4:	Resultados	76
4.1	Resultados de la experimentación con YOLOv11-seg	76
4.1.1	Resultados del experimento 1	76
4.1.2	Resultados del experimento 2 y selección del modelo final	77
4.2	Resultados del benchmark comparativo multi-seed	78
4.2.1	Resultados globales promedio	78
4.2.2	Distribución de métricas entre corridas	79
4.2.3	Análisis estadístico	80

4.2.4	Resultados por clase	81
4.2.5	Análisis de matrices de confusión entre clases	81
4.2.6	Resultados de eficiencia computacional	83
4.2.7	Resultados cualitativos	84
4.2.8	Capacidad funcional de las arquitecturas	85
4.3	Resultados del despliegue en aplicación móvil	89
4.3.1	Resultados visuales de segmentación	89
4.3.2	Evaluación del rendimiento computacional	94
4.3.3	Estabilidad del sistema	100
4.4	Discusión	101
Capítulo 5: Conclusiones		106
Capítulo 6: Trabajo futuro		108
ANEXOS		109
Anexo A	Curvas de entrenamiento del modelo YOLOm11-seg	109
Anexo B	Resultados cualitativos adicionales de YOLO11m-seg	109
Anexo C	Backlog de la aplicación móvil	112
C.1	Sprints del sistema	112
C.2	Backlog de historias de usuario de la aplicación móvil	113
Anexo D	Arquitectura en capas de la aplicación móvil	128
Anexo E	Visualización en video de la aplicación móvil	128
Anexo F	Registros de ejecución del sistema de evaluación de dispositivos móviles (Logcat)	129
Curriculum vitae		140

Índice de tablas

Tabla 1.1:	Resumen de trabajos relacionados en detección, segmentación y clasificación automática de quemaduras mediante aprendizaje profundo. Det. = detección; Seg. = segmentación; Clas. = clasificación; aug. = imágenes generadas mediante aumentación de datos.	13
Tabla 3.1:	Distribución de datos por etapas del dataset de quemaduras.	46
Tabla 3.2:	Clases del dataset de quemaduras.	46
Tabla 3.3:	Distribución final de imágenes utilizadas en cada subconjunto del dataset. .	48
Tabla 3.4:	Configuraciones evaluadas durante el experimento 1 para la selección inicial de hiperparámetros.	53
Tabla 3.5:	Configuraciones evaluadas durante el experimento 2 utilizando 300 épocas de entrenamiento.	54
Tabla 3.6:	Entorno computacional utilizado para entrenamiento y evaluación experimental.	60
Tabla 4.1:	Resultados obtenidos en el experimento 1.	76
Tabla 4.2:	Resultados obtenidos en el experimento 2 bajo el protocolo experimental definitivo.	77
Tabla 4.3:	Resultados globales promedio (10 corridas) del <i>benchmark multi-seed</i> utilizando métricas binarias de quemadura frente a fondo.	78
Tabla 4.4:	Resultados de la prueba de Kruskal-Wallis para el <i>benchmark multi-seed</i> con métricas globales binarias.	80
Tabla 4.5:	Comparaciones post-hoc significativas mediante Dunn con corrección de Bonferroni.	80

Tabla 4.6:	Resultados promedio por clase obtenidos durante el <i>benchmark multi-seed</i> . La clase <i>background</i> fue excluida del análisis.	81
Tabla 4.7:	Comparación de eficiencia computacional entre YOLO11m-seg y arquitecturas CNN.	83
Tabla 4.8:	Capacidad funcional de los modelos evaluados en el contexto de segmentación de quemaduras e implementación en dispositivos móviles.	88
Tabla 4.9:	Tiempo promedio de procesamiento por etapa del <i>pipeline</i> de inferencia. . .	97
Tabla 4.10:	Comparación entre la latencia del modelo en entorno de <i>benchmark</i> y la latencia del sistema completo en la aplicación móvil.	99
Tabla 4.11:	Clasificación estándar de latencia en sistemas interactivos (adaptado de [1,2]).	99
Tabla 4.12:	Comparación del desempeño del sistema entre dispositivos móviles con diferentes capacidades computacionales.	100
Tabla C.1:	Distribución de historias de usuario por sprints durante el desarrollo de la aplicación móvil	112
Tabla C.2:	Backlog de historias de usuario de la aplicación móvil	113

Índice de figuras

Figura 2.1:	Arquitectura U-Net. El camino codificador (izquierda) extrae características mediante convoluciones y <i>max-pooling</i> , mientras que el camino decodificador (derecha) recupera la resolución espacial. Las flechas grises representan las conexiones de salto (<i>skip connections</i>) que concatenan mapas de características del encoder al decoder. Adaptado de [3].	27
Figura 2.2:	Arquitectura U-Net++. Los nodos intermedios (círculos verdes) entre el encoder y el decoder forman caminos de salto anidados y densos que mejoran la relación semántica. La supervisión profunda se aplica en los nodos de salida de cada columna del decoder. Adaptado de [4].	28
Figura 2.3:	Arquitectura Attention U-Net. Las puertas de atención (AG) se insertan en las conexiones de salto y ponderan espacialmente las características del encoder según la señal de contexto proveniente del decoder, suprimiendo regiones irrelevantes para la segmentación. Adaptado de [5].	30
Figura 2.4:	Arquitectura general del modelo YOLOv11-seg empleado en este trabajo. El diagrama muestra los módulos <i>backbone</i> , <i>neck</i> y <i>head</i> utilizados para la extracción de características, fusión multiescala y segmentación de instancias. Elaboración propia con base en [6].	32
Figura 3.1:	Distribución de clases en el conjunto de datos clínico.	49
Figura 3.2:	Distribución del número de objetos anotados por imagen en el conjunto de datos clínico.	50
Figura 3.3:	Arquitectura general de la aplicación móvil. Elaboración propia	65
Figura 3.4:	Caso de uso de la aplicación móvil: Elaborado por el autor	66
Figura 3.5:	<i>Pipeline</i> de inferencia de la aplicación móvil: Elaborado por el autor . . .	70

Figura 3.6:	Árbol de decisión clínica utilizado para la generación de recomendaciones en el sistema. Elaboración propia	74
Figura 4.1:	Distribución de métricas obtenidas durante las 10 corridas independientes para cada arquitectura evaluada.	79
Figura 4.2:	Matrices de confusión normalizadas para las arquitecturas evaluadas. Las filas representan la clase de referencia y las columnas representan la clase predicha: fondo, quemadura de segundo grado y quemadura de tercer grado.	82
Figura 4.3:	Relación entre desempeño global de segmentación, medido mediante Dice global, y complejidad computacional (GFLOPs) de las arquitecturas evaluadas.	84
Figura 4.4:	Relación entre latencia promedio de inferencia y consumo máximo de memoria GPU (VRAM) de las arquitecturas evaluadas.	85
Figura 4.5:	Comparación cualitativa entre arquitecturas utilizando imágenes del conjunto de prueba. Las regiones de quemadura de segundo grado se muestran en naranja y las de tercer grado en rojo.	86
Figura 4.6:	Mapas de error generados a partir de las predicciones de los modelos sobre imágenes clínicas del conjunto de prueba; falsos positivos (cian), falsos negativos (amarillo) y errores de clasificación entre grados de quemadura (magenta).	87
Figura 4.7:	Tiempo promedio de procesamiento por etapa del <i>pipeline</i> de inferencia.	97
Figura 4.8:	Distribución de latencia por etapa del <i>pipeline</i> de inferencia.	98
Figura A.1:	Curvas de entrenamiento y validación obtenidas durante el entrenamiento del modelo YOLOv11m-seg seleccionado como configuración final en la experimentación de distintas configuraciones.	109
Figura B.1:	Resultados cualitativos obtenidos por el modelo YOLO11m-seg sobre imágenes clínicas del conjunto de prueba. Cada ejemplo muestra la imagen original, la anotación <i>ground truth</i> y la predicción generada por el sistema. Las regiones segmentadas corresponden a quemaduras de segundo grado (naranja) y tercer grado (rojo).	110

Figura B.2:	Continuación de los resultados cualitativos obtenidos por el modelo YOLO11m-seg sobre imágenes clínicas del conjunto de prueba.	111
Figura B.3:	Mapas de error generados por el modelo YOLO11m-seg sobre imágenes del conjunto de prueba. Las regiones coloreadas representan discrepancias entre la segmentación predicha y la anotación <i>ground truth</i> , incluyendo falsos positivos (rojo), falsos negativos (amarillo) y errores de clasificación (púrpura).	127
Figura D.1:	Arquitectura en capas de la aplicación móvil	128

Índice de ilustraciones

Ilustración 2.1	Clasificación de la profundidad de quemaduras [7]	21
Ilustración 3.1	Ejemplo del proceso de anotación manual de una imagen clínica de quemadura mediante polígonos en la herramienta LabelMe.	45
Ilustración 4.1	Capturas de pantalla del flujo del sistema principal en dispositivo móvil. .	90
Ilustración 4.2	Capturas de pantalla del del sistema de almacenamiento de datos en dispositivo móvil.	91
Ilustración 4.3	Ejemplo de resultado de inferencia del sistema en el dispositivo móvil. A la izquierda se muestra la imagen original y a la derecha el resultado generado por el modelo, incluyendo la segmentación y la detección de la región afectada.	92
Ilustración 4.4	Ejemplo de resultado de inferencia del sistema en el dispositivo móvil. A la izquierda se muestra la imagen original y a la derecha el resultado generado por el modelo, incluyendo la segmentación y la detección de la región afectada.	93
Ilustración 4.5	Recomendaciones de primeros auxilios mostrados por el sistema	93
Ilustración 4.6	Ejemplo de reporte en formato PDF generado por el sistema	95



APLICACIÓN MÓVIL PARA LA CLASIFICACIÓN DE QUEMADURAS EN PACIENTES BASADA EN MODELOS DE APRENDIZAJE PROFUNDO

Capítulo 1: Introducción

1.1. Introducción a la problemática

Las quemaduras en la piel representan un problema de salud a nivel global; aproximadamente 180,000 personas pierden la vida cada año por esta causa, con mayor frecuencia en países con recursos limitados [8]. En México, alrededor de 13,000 personas sufren algún tipo de quemadura y aproximadamente 1,000 casos se relacionan con población infantil; además, las quemaduras se encuentran entre las causas relevantes de mortalidad reportadas en el país [9]. Las lesiones no mortales también pueden generar consecuencias importantes, como hospitalización prolongada, discapacidad, desfiguración y secuelas funcionales [8].

La evaluación de la profundidad y extensión de una quemadura juega un papel importante en la toma de decisiones clínicas, especialmente cuando se requiere determinar la necesidad de tratamiento especializado o intervenciones como injertos de piel. Sin embargo, esta evaluación puede ser compleja durante las primeras etapas de la lesión, debido a la variabilidad visual de las heridas, la presencia de regiones con distinta severidad dentro de una misma quemadura y la posible variación entre observadores [10].

En este contexto, los métodos basados en inteligencia artificial y visión por computadora representan una alternativa para apoyar la evaluación visual de quemaduras a partir de imágenes clínicas. En particular, los modelos de aprendizaje profundo permiten segmentar regiones lesionadas y clasificar su grado de severidad, generando información visual que puede complementar la valoración realizada por el personal clínico [11].

Sin embargo, todavía existen desafíos relacionados con la integración simultánea de segmentación, clasificación y eficiencia computacional en sistemas orientados a escenarios de uso clínico real [11]. Todavía existen desafíos relacionados con la integración simultánea de segmen-



tación, clasificación y eficiencia computacional en sistemas orientados a escenarios de uso clínico real. Diversos trabajos han reportado resultados prometedores en el análisis automático de quemaduras y el desarrollo de aplicaciones móviles de apoyo clínico, muchos de estos enfoques se centran en tareas específicas, como la estimación de la superficie corporal afectada, la telemedicina o la segmentación de lesiones, sin evaluar de manera conjunta el desempeño del modelo y su viabilidad de despliegue en dispositivos móviles [12, 13].

En este trabajo se desarrolla un sistema basado en aprendizaje profundo para la segmentación y clasificación automática de quemaduras de segundo y tercer grado en imágenes clínicas, con énfasis en la comparación experimental entre la arquitectura de segmentación de instancias YOLOv11-seg y arquitecturas de segmentación semántica de la familia U-Net, así como en la integración del modelo seleccionado en una aplicación móvil Android.

Las principales contribuciones de este trabajo son las siguientes:

- Implementación y optimización de un modelo basado en YOLOv11-seg para la segmentación y clasificación de quemaduras en la piel de segundo y tercer grado
- Comparación experimental entre YOLOv11-seg y arquitecturas de segmentación semántica basadas en CNN, incluyendo U-Net, Attention U-Net y U-Net++
- Desarrollo de un *benchmark multi-seed* con 10 corridas independientes por arquitectura y análisis estadístico mediante pruebas no paramétricas
- Evaluación cuantitativa mediante métricas globales y métricas por clase para quemaduras de segundo y tercer grado
- Análisis de eficiencia computacional considerando parámetros, FLOPs, FPS, latencia y consumo de memoria GPU
- Conversión e integración del modelo YOLOv11-seg seleccionado en una aplicación móvil Android mediante TensorFlow Lite para inferencia local en dispositivos reales
- Evaluación del desempeño del sistema móvil mediante un análisis de latencia en dispositivos Android de diferentes capacidades computacionales



1.2. Antecedentes y estado del arte

1.2.1. Fundamentos clínicos de las quemaduras

Las quemaduras corresponden a lesiones tisulares producidas por agentes térmicos, químicos, eléctricos o radiación. Clínicamente, se clasifican según la profundidad del daño en quemaduras de primer, segundo y tercer grado [14]. Las quemaduras de segundo grado profundo y tercer grado suelen requerir intervención quirúrgica mediante injertos de piel, por lo que la estimación precisa de la profundidad representa un factor importante en la toma de decisiones clínicas [15]. La evaluación visual de la severidad puede resultar compleja debido a que las lesiones presentan variaciones espaciales dentro de una misma herida y pueden evolucionar durante los primeros días posteriores a la quemadura [7]. Por ello, los métodos automáticos de análisis de imágenes orientados a segmentar regiones específicas según su profundidad han adquirido interés en aplicaciones médicas asistidas por inteligencia artificial.

1.2.2. Análisis automático de quemaduras mediante visión por computadora

Diversos estudios han explorado el uso de técnicas de visión por computadora en el ámbito médico para el análisis automático de quemaduras en la piel. En particular, se han propuesto enfoques basados en redes neuronales convolucionales (CNN) para la clasificación de la severidad de las lesiones a partir de imágenes clínicas. Trabajos como los de Suha y Sanam (2021, 2022) destacan el potencial de estas soluciones como herramientas de apoyo al personal de salud, al permitir diagnósticos más oportunos y facilitar su aplicación en escenarios de telemedicina, especialmente en contextos con recursos limitados [7]. No obstante, estos enfoques presentan limitaciones cuando una misma imagen contiene regiones con distintos grados de severidad, lo que dificulta una evaluación precisa a nivel regional.

Las CNN no solo han sido empleadas para tareas de clasificación, sino también para la segmentación de heridas por quemadura, permitiendo la delimitación automática de las regiones afectadas dentro de una imagen. Por ejemplo, Liu et al. (2021) [16] propusieron un marco basado en aprendizaje profundo para la segmentación automática de regiones quemadas en imágenes clínicas, reportando un buen desempeño en la identificación de áreas lesionadas. No obstante, el enfoque



de segmentación empleado se centra principalmente en la delimitación binaria de la lesión, por lo que la diferenciación precisa entre distintos grados de severidad continúa representando un desafío importante en escenarios clínicos reales.

Además de los enfoques basados exclusivamente en redes neuronales profundas, la literatura también muestra la combinación de métodos de preprocesamiento de imágenes con modelos de aprendizaje automático para el análisis de quemaduras. Por ejemplo, Rangaiah et al. (2025) [17] proponen un sistema de recuperación de imágenes médicas que integra técnicas de mejora de contraste, como CLAHE, con clasificadores basados en redes neuronales artificiales. Si bien estos enfoques híbridos pueden mejorar la calidad visual de las imágenes y favorecer la discriminación de características relevantes, generalmente requieren múltiples etapas independientes para llevar a cabo tareas como la mejora de imagen, la localización de la lesión y la clasificación de la severidad. En escenarios clínicos reales, esta fragmentación del proceso limita la robustez del sistema, especialmente cuando las imágenes presentan alta variabilidad en condiciones de iluminación, adquisición, anatomía del paciente y coexistencia de distintos grados de severidad dentro de una misma lesión.

Un aspecto comúnmente presentado en visión por computadora en el análisis de imágenes médicas es la escasez de datos clínicos etiquetados. A diferencia de dominios como las imágenes naturales, donde existen abundantes imágenes anotadas disponibles públicamente, los conjuntos de datos médicos son costosos de obtener y requieren anotación por especialistas médicos, lo que limita el tamaño de los conjuntos de entrenamiento disponibles [18]. Frente a esta restricción, el aprendizaje por transferencia (*transfer learning*) a partir de modelos previamente entrenados en grandes conjuntos de imágenes naturales, como ImageNet, se ha consolidado como una estrategia habitual para inicializar los pesos de modelos de segmentación y clasificación médica [19]. No obstante, estudios previos han señalado que existe una diferencia sustancial entre el dominio de imágenes naturales y médicas, ya que estas últimas, a menudo, tienen escalas, texturas y características inherentemente diferentes a las primeras [19]. Esta limitación es especialmente relevante en el análisis de quemaduras, donde los modelos deben aprender a discriminar patrones visuales específicos de daño tisular que no tienen equivalente en ningún dataset de preentrenamiento de propósito general, lo que refuerza la necesidad de estrategias de entrenamiento que maximicen el aprovechamiento de conjuntos de datos reducidos y heterogéneos.



En consecuencia, la literatura evidencia la necesidad de enfoques más integrados que permitan realizar de manera conjunta la segmentación y la clasificación de las áreas quemadas, reduciendo la dependencia de múltiples etapas manuales o heurísticas y mejorando la aplicabilidad de estos sistemas en contextos clínicos reales.

1.2.3. Modelos supervisados para segmentación y clasificación de quemaduras

El análisis automático de quemaduras mediante aprendizaje profundo ha evolucionado desde enfoques centrados únicamente en segmentación o clasificación hacia modelos capaces de integrar ambas tareas dentro de una misma arquitectura. A continuación se describen algunos de los principales enfoques supervisados reportados en la literatura para segmentación y clasificación de quemaduras.

1.2.3.1. Arquitecturas CNN para segmentación médica

El desarrollo de arquitecturas convolucionales orientadas a la segmentación de imágenes médicas ha evolucionado significativamente a partir de las redes completamente convolucionales (Fully Convolutional Networks, FCN), propuestas por Long et al. (2015) [20], quienes demostraron que una red convolucional entrenada para clasificación podía adaptarse para producir mapas de activación densos a escala de píxel mediante la supresión de las capas completamente conectadas. Esta idea sentó las bases para los enfoques encoder-decoder, en los que una rama de codificación extrae representaciones semánticas progresivamente abstractas y una rama de decodificación las recupera espacialmente para producir una máscara de segmentación.

Más adelante, Badrinarayanan et al. (2017) [21] presentaron SegNet, una arquitectura encoder-decoder orientada a tareas de comprensión de escenas en tiempo real. A diferencia de las FCN, SegNet reutiliza los índices obtenidos durante las operaciones de *max-pooling* en la etapa de codificación para realizar el muestreo ascendente en el decoder. De esta manera, evita aprender operaciones adicionales de reconstrucción espacial, reduciendo así el consumo de memoria durante el entrenamiento e inferencia. Aunque este enfoque permite una arquitectura más eficiente, la recuperación de detalles espaciales finos es más limitada en comparación con modelos que



incorporan conexiones de salto. Aun así, SegNet representó un avance importante en el desarrollo de arquitecturas encoder-decoder eficientes para segmentación semántica y contribuyó a establecer varias de las ideas que posteriormente influenciaron a arquitecturas encoder-decoder como U-Net y sus variantes.

La arquitectura U-Net, introducida por Ronneberger et al. [22] para la segmentación de imágenes histológicas, extendió este paradigma mediante la incorporación de conexiones de salto (skip connections) entre capas simétricas (al mismo nivel) del encoder y el decoder. Estas conexiones preservan información espacial de alta resolución que se perdería durante el submuestreo progresivo, lo que permite generar máscaras con bordes precisos incluso con conjuntos de datos reducidos. Debido a su buen desempeño incluso con conjuntos de datos reducidos, U-Net se convirtió en una de las arquitecturas más utilizadas en segmentación biomédica. En el contexto de quemaduras, múltiples trabajos han adoptado variantes de U-Net para la segmentación de heridas, obteniendo resultados competitivos en métricas de solapamiento como Dice e IoU [23].

La variante Attention U-Net, propuesta por Oktay et al. [24], introdujo mecanismos de atención en los caminos de skip connection, con el objetivo de suprimir activaciones irrelevantes y enfocarse selectivamente en las regiones de interés. Este mecanismo puede ser útil en escenarios donde las lesiones presentan alta variabilidad morfológica, como ocurre en quemaduras de diferente profundidad y extensión, ya que permite al modelo ponderar de forma adaptativa las regiones más informativas sin incrementar sustancialmente el costo computacional. Por su parte, U-Net++ [25] propuso un esquema de conexiones densas anidadas entre los nodos del encoder y el decoder, reemplazando los puentes directos de U-Net por rutas intermedias que agregan representaciones a distintas profundidades semánticas. Este rediseño permite una recuperación gradual del detalle espacial y reduce la brecha semántica entre las características codificadas y decodificadas. En distintos trabajos de segmentación médica multiclase, U-Net++ reportó mejoras respecto a U-Net estándar, especialmente cuando las clases presentan tamaños o apariencias heterogéneas.

A pesar de su consolidación en segmentación médica, las arquitecturas basadas en U-Net presentan limitaciones relevantes para ciertos escenarios clínicos. Estas arquitecturas realizan segmentación semántica, asignando una clase a cada píxel de la imagen. Aunque esto permite diferenciar regiones correspondientes a distintos grados de quemadura, no separa explícitamente instancias individuales de una misma clase clínica. En escenarios con múltiples lesiones cercanas o



superpuestas, esta limitación puede dificultar la delimitación independiente de cada región afectada. En contraste, los modelos de segmentación de instancias integran mecanismos de detección y segmentación que permiten identificar regiones individuales dentro de la imagen.

En años recientes, algunos trabajos han comenzado a incorporar mecanismos de autoatención global mediante *Vision Transformers* (ViT) en tareas de segmentación médica. Un ejemplo es TransUNet, propuesto por Chen et al. [26], que combina capas convolucionales con bloques Transformer dentro de un codificador híbrido. Esta integración permite capturar relaciones entre regiones distantes de la imagen, algo que las CNN tradicionales suelen modelar de forma más limitada debido a la naturaleza local de las convoluciones. A pesar de sus ventajas en la representación global de características, estos modelos suelen requerir una mayor capacidad computacional, más memoria y grandes volúmenes de datos para preentrenamiento. Estas condiciones dificultan su implementación en dispositivos móviles o en escenarios con conjuntos de datos relativamente reducidos, como ocurre frecuentemente en imágenes de quemaduras. Por ello, en este trabajo se optó por arquitecturas basadas en CNN ligeras basadas en arquitecturas U-Net y modelos de la familia YOLO, ya que ofrecen un balance más adecuado entre desempeño, costo computacional y viabilidad de despliegue en dispositivos con recursos limitados.

1.2.3.2. Segmentación y clasificación de quemaduras mediante *deep learning*

La aplicación de redes neuronales profundas al análisis de imágenes clínicas de quemaduras ha seguido dos líneas principales: la segmentación de la región lesionada y la clasificación de su profundidad. Durante mucho tiempo ambas tareas se abordaron de forma independiente, aunque los trabajos más recientes han comenzado a integrarlas en sistemas unificados con mayor utilidad clínica.

En el ámbito de la segmentación, uno de los trabajos más citados en contexto clínico real es el de Chang et al. [27], quienes propusieron un método basado en aprendizaje profundo para la detección y segmentación de quemaduras a nivel de píxel, con el objetivo de estimar automáticamente el porcentaje de superficie corporal quemada (%TBSA). Empleando U-Net y Mask R-CNN con diferentes *backbones* sobre un conjunto de 2,591 imágenes clínicas etiquetadas por cirujanos especialistas, obtuvieron un coeficiente Dice de 0.9496 con Mask R-CNN y ResNet101, superando incluso la consistencia de evaluadores clínicos en la estimación del %TBSA. Este trabajo muestra



las dificultades de trabajar con imágenes clínicas reales, donde existen variaciones importantes en iluminación, captura y apariencia de la lesión.

Con el objetivo de abordar simultáneamente la segmentación y la clasificación de profundidad de quemaduras (primero, segundo y tercer grado), Liu et al. [16] propusieron un marco de extremo a extremo basado en redes encoder-decoder con diferentes *backbones*, capaz de segmentar la región quemada y clasificar su grado en una sola etapa. Su sistema fue entrenado sobre un conjunto de datos clínico propio con 1200 imágenes, etiquetado por especialistas, incorporando aumentación de datos y funciones de pérdida híbridas para compensar la escasez de muestras anotadas. Los resultados mostraron un IoU de 0.8467 para la segmentación de zona quemada/no quemada y un IoU promedio de 0.5144 para la segmentación de múltiples áreas de profundidad de quemadura. Los autores argumentan que el uso de imágenes capturadas con cámaras convencionales o teléfonos inteligentes representa una vía viable para automatizar la evaluación de quemaduras y reducir tanto el costo del diagnóstico como el impacto del error humano, lo cual guarda relación con el enfoque propuesto en esta tesis.

Un avance relevante hacia la integración de segmentación y clasificación de profundidad de quemaduras en la piel, lo representa el trabajo de Cai et al. [23], quienes propusieron GL-FusionNet: un sistema en dos etapas que primero utiliza U-Net para segmentar binariamente la herida (herida/ vs. no herida) obteniendo un Dice de 0.9258 e IoU de 0.8872. Posteriormente, aplica un modelo de clasificación que fusiona características globales y locales mediante ResNet50 y ResNet101. El objetivo fue distinguir entre quemaduras de espesor parcial superficial y profundo, dos subtipos del segundo grado de quemadura, sin considerar quemaduras de primer o tercer grado. El modelo alcanzó una exactitud de 0.9352 en esta tarea binaria, evaluada mediante validación cruzada de 5 pliegues sobre un conjunto de aproximadamente 500 imágenes clínicas balanceadas entre ambas clases. Cabe señalar que el hecho de que cada imagen contenga únicamente una profundidad de herida, y que el clasificador opere exclusivamente sobre la región ya segmentada (eliminando el fondo), simplifica considerablemente la tarea respecto a escenarios clínicos reales, lo que puede contribuir en las métricas de los resultados. Este trabajo muestra que integrar segmentación y clasificación puede aportar información más útil para la interpretación clínica. No obstante, su arquitectura en cascada implica dos modelos independientes, lo que incrementa la complejidad de despliegue y la latencia de inferencia.



Desde una perspectiva clínica, el review 2024 de machine learning sobre severidad de quemaduras de Wilson et al. [15] se señala que una misma lesión por quemadura puede presentar regiones con diferente profundidad, donde algunas áreas requieren injerto mientras otras pueden sanar sin intervención quirúrgica. Esta variabilidad en las regiones de una imagen representa una limitación para los enfoques de clasificación global de imágenes, ya que asignar una única etiqueta a toda la lesión puede no reflejar adecuadamente la complejidad clínica de la herida. En este contexto, los métodos de segmentación permiten delimitar regiones específicas y asociarlas con distintos grados de severidad dentro de una misma imagen, proporcionando información potencialmente más útil para la evaluación clínica. Esta motivación es consistente con el enfoque adoptado en la presente tesis, donde se busca generar máscaras de segmentación diferenciadas según el grado de quemadura presente en cada región de la lesión.

Recientemente, trabajos orientados a condiciones clínicas no controladas han comenzado a explorar la robustez de estos sistemas ante variabilidad de dispositivos y entornos de captura. El modelo Dfusion del 2025 [28], basado en DeepLabv3 con *backbones* ResNet101 y mecanismos de atención espacial y de canal, fue evaluado sobre un conjunto propio de 4,785 imágenes de quemaduras pediátricas capturadas con teléfonos de diferentes fabricantes en entornos hospitalarios y ambulatorios. A diferencia de enfoques previos que abordan segmentación y clasificación como etapas separadas, DFusion realiza ambas tareas mediante segmentación semántica multiclase, distinguiendo entre quemaduras de segundo grado superficial, segundo grado profundo y tercer grado. Para compensar el desbalance de clases presente en el conjunto de datos, los autores utilizaron una función de pérdida de entropía cruzada ponderada. El modelo alcanzó un Dice de $0,8766 \pm 0,012$ e IoU de $0,8052 \pm 0,015$ en segmentación, además de una exactitud de 0,9765 y un F1-score de 0,8533 en clasificación de profundidad. Los resultados mostraron que este tipo de sistemas puede mantener un desempeño estable incluso bajo condiciones clínicas no controladas, evidenciando el potencial de los enfoques basados en aprendizaje profundo para aplicaciones de apoyo clínico a partir de imágenes capturadas con dispositivos móviles.

A pesar de estos avances, todavía existen varias limitaciones en la literatura actual. La mayoría de los conjuntos de datos empleados son de acceso restringido, etiquetados manualmente por especialistas y con escasa representación de poblaciones diversas. Asimismo, trabajos como los de Chang et al. [27], Liu et al. [16] y Cai et al. [23] no reportan evaluaciones detalladas



sobre eficiencia computacional o latencia de inferencia en dispositivos reales, aspectos importantes para la transferencia clínica y el despliegue móvil. Además, enfoques como GL-FusionNet [23] continúan abordando la segmentación y la clasificación de profundidad mediante arquitecturas en múltiples etapas o modelos independientes, lo que incrementa la complejidad computacional y dificulta el intercambio de información entre tareas. Estas limitaciones motivan el desarrollo de modelos integrados de una sola etapa orientados a aplicaciones clínicas y dispositivos móviles, como el enfoque propuesto en este trabajo.

1.2.3.3. Enfoques basados en YOLO para análisis de quemaduras

Trabajos recientes han explorado modelos basados en la familia YOLO para la detección de quemaduras en la piel, debido a ventajas como su eficiencia computacional y su capacidad de inferencia en tiempo real. En particular, Ferdinand et al. (2023) [29] emplearon un modelo basado en YOLOv5 para la detección y clasificación de quemaduras, aplicando técnicas de *fine-tuning* sobre dos conjuntos de datos de 3691 imágenes obtenidos de las plataformas Kaggle y Roboflow. El mejor desempeño se obtuvo después del proceso de ajuste de hiperparámetros, alcanzando un F1-score de 0.759, un mAP@0.5 de 0.837 y un mAP@0.5:0.95 de 0.834. Los resultados mostraron que las arquitecturas YOLO pueden adaptarse de manera efectiva al dominio de imágenes de quemaduras, respaldando su potencial para aplicaciones clínicas asistidas por computadora. No obstante, el estudio también evidencia la escasez de datos anotados en el dominio de quemaduras, lo cual continúa siendo un desafío importante para el desarrollo de modelos robustos en imágenes clínicas.

En el contexto específico de la detección automática de quemaduras, Yıldız et al. [30] propusieron el uso de YOLOv7 para la detección y clasificación simultánea de regiones lesionadas mediante *bounding boxes*, distinguiendo entre quemaduras de primer, segundo y tercer grado, además del desarrollo de una aplicación móvil Android orientada a la evaluación de la gravedad de las lesiones. Los autores emplearon un conjunto de datos público de Roboflow compuesto por 7,006 imágenes originales, que tras aumentación de datos alcanzó un total de 21,018 imágenes distribuidas en tres clases.

El modelo obtuvo un mAP@0.5 de 0.751 en tareas de detección automática, mostrando un desempeño inferior en quemaduras de primer grado (mAP@0.5 = 0.639) respecto a segundo y



tercer grado (0.808 y 0.801, respectivamente). Asimismo, el trabajo demuestra el potencial de las arquitecturas basadas en YOLO para aplicaciones médicas con implementación en dispositivos móviles, reportando una tasa de acierto del 80 % en una validación clínica realizada sobre 50 pacientes reales. Sin embargo, el uso de conjuntos de datos públicos puede limitar la representatividad del dominio clínico real, especialmente debido a las diferencias entre imágenes obtenidas de internet y fotografías capturadas en entornos hospitalarios. Además, la detección basada en *bounding boxes* no proporciona una delimitación precisa a nivel de píxel, lo que restringe su utilidad para estimar el área exacta de la lesión.

En una publicación reciente, Luo et al. (2025) [31] propusieron una variante de YOLOv7 con un módulo de atención PSA para la detección automática de heridas por quemaduras. El estudio utilizó un conjunto de datos clínico propio compuesto por 492 imágenes originales de quemaduras y escaldaduras recolectadas en el Ruijin Hospital de Shanghái. Las lesiones fueron anotadas mediante consenso entre especialistas en cirugía reconstructiva y quemaduras, clasificando regiones de primer, segundo y tercer grado. Para incrementar el tamaño del conjunto de entrenamiento, los autores aplicaron técnicas de aumentación de datos como rotación, recorte, ajuste de brillo y ruido, generando aproximadamente 2952 imágenes aumentadas distribuidas en entrenamiento, validación y prueba. El modelo alcanzó un F1-score promedio de 0.894, un mAP@0.5 de 0.877 y un mAP@0.5:0.95 de 0.688 en tareas de detección automática de quemaduras. Adicionalmente, la matriz de confusión mostró tasas de detección correcta de 0.86, 0.87 y 0.91 para quemaduras de primer, segundo y tercer grado, respectivamente. Estos resultados evidencian el potencial de las arquitecturas basadas en YOLO para el análisis automático de quemaduras y constituyen un antecedente relevante para enfoques más avanzados orientados a integrar detección y segmentación dentro de una misma arquitectura. Sin embargo, las tareas basadas únicamente en detección y clasificación mediante cajas delimitadoras presentan limitaciones para la delimitación precisa de las regiones lesionadas, por lo que resulta relevante incorporar métodos de segmentación a nivel de píxel que permitan representar de manera más detallada la extensión y severidad de la quemadura.

Dentro de la familia YOLO, las capacidades de segmentación se consolidaron a partir de YOLOv8, presentado por Ultralytics en 2023 [32]. A diferencia de versiones anteriores enfocadas principalmente en detección mediante cajas delimitadoras, YOLOv8 incorporó soporte nativo para tareas como segmentación, clasificación y estimación de pose dentro de una misma arquitectura.



Además, introdujo un enfoque *anchor-free* y un mecanismo de segmentación basado en prototipos, permitiendo delimitar objetos individuales de manera más eficiente.

Posteriormente, YOLO11 [33], lanzado en 2024, extendió estas capacidades mediante mejoras en el *backbone* y el *neck* del modelo, optimizando la extracción de características y logrando una mejor relación entre precisión y costo computacional. La variante YOLO11-seg mantiene soporte para detección y segmentación en una sola pasada de inferencia, lo que la convierte en una alternativa adecuada para aplicaciones con restricciones de tiempo y recursos computacionales. Debido a estas características, YOLO11-seg fue seleccionado como configuración de referencia en el presente trabajo, ya que permite realizar detección, clasificación y segmentación de lesiones por quemadura de forma simultánea, manteniendo un nivel de eficiencia compatible con su despliegue en dispositivos móviles.

En este sentido, la presente tesis se diferencia de estos enfoques al utilizar una variante de YOLO orientada a segmentación de instancias, permitiendo obtener no solo la clase y localización aproximada de la lesión, sino también una máscara a nivel de píxel para cada región detectada.

La Tabla 1.1 muestra un resumen de las publicaciones antes mencionadas. En conjunto, estos trabajos muestran que las arquitecturas basadas en YOLO representan una alternativa prometedora para el análisis automático de quemaduras debido a su eficiencia computacional y capacidad de inferencia en tiempo real [29–31]. A pesar de estos avances, gran parte de los enfoques reportados en la literatura se centran principalmente en tareas de detección o clasificación mediante cajas delimitadoras, mientras que pocos trabajos integran simultáneamente detección, clasificación y segmentación de instancias dentro de una misma arquitectura. Además, la evaluación en dispositivos móviles reales y en escenarios clínicos no controlados continúa siendo limitada, lo que motiva el desarrollo de enfoques integrados orientados a aplicaciones clínicas prácticas, como el propuesto en esta tesis [12, 13, 30].

Tabla 1.1: Resumen de trabajos relacionados en detección, segmentación y clasificación automática de quemaduras mediante aprendizaje profundo.
Det. = detección; Seg. = segmentación; Clas. = clasificación; aug. = imágenes generadas mediante aumentación de datos.

Referencia	Año	Dataset	N.º imágenes	Clases	Det.	Seg.	Clas.	Tipo de salida	Modelo	Métricas principales
Chang et al.	2021	Propio clínico	2,591	Quemadura / no quemadura	—	✓	—	Máscara binaria	Mask R-CNN + ResNet101	Dice = 0.9496
Liu et al.	2021	Propio clínico	1,200	1.º, 2.º y 3.º grado	—	✓	✓	Segmentación multiclase	Encoder-decoder CNN	IoU binario = 0.8467, IoU multiclase = 0.5144
Cai et al.	2023	Propio clínico	500 orig. / 3,264 aug.	2.º superficial / 2.º profundo	—	✓	✓	Máscara binaria + clase	U-Net + GL-FusionNet	Dice = 0.9259, IoU = 0.8873, Acc = 0.9352
Ferdinand et al.	2023	Kaggle + Roboflow	3,691 orig. / 10,951 aug.	1.º, 2.º y 3.º grado	✓	—	✓	Bounding boxes	YOLOv5l	F1 = 0.759, mAP@0.5 = 0.837, mAP@0.5:0.95 = 0.834
Yıldız et al.	2024	Roboflow	7,006 orig. / 21,018 aug.	1.º, 2.º y 3.º grado	✓	—	✓	Bounding boxes	YOLOv7	mAP@0.5 = 0.7512, mAP@0.5:0.95 = 0.3925
Luo et al.	2025	Propio clínico	492 orig. / aug.	1.º, 2.º y 3.º grado	✓	—	✓	Bounding boxes	YOLOv7-PSA	F1 = 0.894, mAP@0.5 = 0.877, mAP@0.5:0.95 = 0.688
Li et al.	2025	Propio clínico	4,785 imágenes / 14,355 regiones	2.º superficial, 2.º profundo y 3.º grado	—	✓	✓	Segmentación semántica multiclase	DFusion	Dice = 0.8766, IoU = 0.8052, Acc = 0.9765, F1 = 0.8533



En conjunto, estos trabajos muestran el creciente interés en el uso de aplicaciones móviles como herramientas de apoyo en la evaluación y manejo inicial de quemaduras. En este contexto, las funcionalidades, el nivel de automatización y la disponibilidad clínica varían considerablemente entre propuestas. Estos resultados no son directamente comparables entre sí debido a diferencias en tarea, tipo de salida, clases evaluadas, tamaño del dataset y protocolo experimental.

1.2.3.4. Aplicaciones móviles para apoyo clínico en quemaduras

Además de los enfoques basados en modelos de aprendizaje profundo, diversos trabajos han explorado el uso de aplicaciones móviles como herramientas de apoyo en la atención de pacientes con quemaduras. Estas aplicaciones buscan agilizar la evaluación inicial, estandarizar protocolos y mejorar la comunicación entre niveles de atención.

En el ámbito de las aplicaciones móviles para apoyo clínico en quemaduras, Frederick et al. [34] desarrollaron TOBI (*Telemedicine Optimized Burn Intervention*), una aplicación orientada al seguimiento remoto de pacientes pediátricos ambulatorios con quemaduras. A diferencia de los enfoques basados en aprendizaje profundo, el sistema no incorpora modelos automáticos de segmentación o clasificación, sino que permite la captura y envío de imágenes clínicas para su evaluación remota por especialistas. En un ensayo clínico aleatorizado realizado en Estados Unidos con 65 pacientes pediátricos, los autores reportaron que los pacientes que utilizaron la aplicación presentaron menor percepción de dolor, menos visitas presenciales y menor tiempo de traslado en comparación con el seguimiento convencional cara a cara. Aunque el estudio fue realizado en un único centro, los resultados muestran el potencial de las aplicaciones móviles como herramientas de apoyo clínico remoto en pacientes con quemaduras.

Actualmente, existen herramientas de soporte clínico ampliamente utilizadas, como *Mersey Burns*, desarrollada por el National Health Service (NHS) del Reino Unido. Esta aplicación permite el ingreso de datos del paciente y la representación gráfica de la extensión de la quemadura sobre un modelo anatómico, estimando automáticamente el porcentaje de superficie corporal total afectada (%TBSA) e integrando protocolos de reanimación. Su uso contribuye a acelerar y estandarizar los pasos iniciales en la atención de pacientes quemados. La aplicación cuenta con versiones de acceso gratuito orientadas al cálculo clínico básico y funcionalidades adicionales para gestión clínica institucional [35].



Desde el punto de vista técnico, el despliegue de modelos de aprendizaje profundo en dispositivos móviles requiere estrategias que reduzcan el tamaño del modelo y el tiempo de inferencia sin afectar significativamente el desempeño. TensorFlow Lite (TFLite) es un framework desarrollado por Google para ejecutar modelos de aprendizaje automático en dispositivos con recursos limitados, como teléfonos Android e iOS [36]. Este framework permite convertir modelos entrenados en TensorFlow a un formato optimizado para inferencia local y ofrece soporte para aceleración mediante GPU y otros aceleradores de hardware.

Entre las técnicas más utilizadas para optimizar modelos móviles se encuentra la cuantización, la cual consiste en representar los pesos y activaciones de la red con formatos numéricos de menor precisión. Jacob et al. [37] reportaron que este tipo de optimización puede reducir el costo computacional y mejorar la eficiencia de inferencia en hardware móvil, manteniendo pérdidas mínimas de precisión.

Por otra parte, algunos trabajos han explorado el uso de tecnologías tridimensionales para mejorar la evaluación objetiva de las quemaduras mediante dispositivos móviles. Un ejemplo es el sistema Burn Evaluation Network (B.E.N.), el cual utiliza sensores LiDAR integrados en dispositivos inteligentes para reconstruir información tridimensional de la herida a partir de imágenes clínicas [13]. Esta estrategia permite estimar con mayor precisión la extensión de la quemadura, especialmente en regiones anatómicas con alta curvatura.

Para la segmentación automática de las heridas, el sistema emplea el modelo DeepLabV3+ con codificador ResNet101, entrenado con más de 10,000 imágenes clínicas anotadas por cirujanos plásticos experimentados [13]. El enfoque corresponde a una tarea de segmentación binaria, donde el modelo diferencia regiones de herida y tejido sano. Los resultados reportados muestran un desempeño robusto, alcanzando un coeficiente Dice cercano a 0.89 y un IoU de 0.8288, además de altos valores de precisión y exhaustividad. En la evaluación clínica, los autores observaron que las mediciones bidimensionales tienden a subestimar el área real de la quemadura en zonas anatómicas complejas, como la cabeza, el cuello y el dorso del pie, mientras que en superficies relativamente planas, como la espalda, las diferencias entre mediciones 2D y 3D fueron mínimas [13]. Asimismo, la aplicación B.E.N. se encuentra disponible públicamente para dispositivos Apple compatibles con sensores LiDAR, como algunos modelos recientes de iPhone y iPad [13].

Además, la aplicación móvil *BurnAid*, presentada por Díaz Granados Zúñiga et al. en



2024 [12], propone una herramienta de apoyo para el manejo inicial del paciente quemado. La aplicación está orientada principalmente a guiar al personal de salud en la estimación del porcentaje de superficie corporal total quemada (%TBSA) y en la toma de decisiones iniciales. Para ello, el usuario registra datos clínicos básicos del paciente, como edad, sexo, peso, talla y tiempo transcurrido desde la quemadura, además de seleccionar condiciones de riesgo como quemadura eléctrica, química, en espacio cerrado o sospecha de lesión de vía aérea.

Posteriormente, la aplicación permite cargar fotografías de las zonas afectadas según la región anatómica correspondiente, con el objetivo de estimar el porcentaje de superficie corporal comprometida mediante inteligencia artificial. Sin embargo, el artículo no especifica con detalle la arquitectura del modelo utilizado ni reporta métricas cuantitativas de desempeño para la detección o segmentación de las quemaduras. Una vez estimado el porcentaje de SCTQ, *BurnAid* genera una guía de manejo inicial que incluye recordatorios del protocolo ABCDE del *Advanced Burn Life Support* y el cálculo de reanimación hídrica mediante la fórmula de Brooke modificada [12]. De acuerdo con los autores, la aplicación fue probada con 810 fotografías de pacientes con quemaduras de primero, segundo y tercer grado, y utilizada en hospitales de los departamentos del Atlántico y Risaralda, Colombia, entre diciembre de 2023 y febrero de 2024. Al momento de la publicación, *BurnAid* se encontraba en fase beta y estaba disponible para pruebas mediante un archivo APK para dispositivos Android [12].

Cabe señalar, que en la revisión antes presentada, se priorizan aplicaciones móviles con evidencia de implementación o disponibilidad documentada en contextos clínicos.

1.2.4. Síntesis del estado del arte y posicionamiento del trabajo

La literatura revisada muestra avances importantes en el análisis automático de quemaduras mediante aprendizaje profundo, incluyendo enfoques de clasificación, detección, segmentación semántica y segmentación de instancias [16,23,27–31]. Aun así, todavía existen desafíos relacionados con la variabilidad clínica de las lesiones debido a la naturaleza de las imágenes clínicas, la escasez de datos anotados por especialistas, la comparación sistemática entre arquitecturas y la evaluación del desempeño en dispositivos móviles reales [12, 13].

En este contexto, el presente trabajo aborda la segmentación de quemaduras de segundo y tercer grado mediante una comparación experimental entre arquitecturas CNN basadas en U-Net



y un modelo YOLO11m-seg orientado a segmentación de instancias. Además, se incorpora un protocolo de evaluación multi-seed, análisis estadístico, evaluación de eficiencia computacional y despliegue en una aplicación móvil Android, con el objetivo de analizar tanto el desempeño del modelo como su viabilidad práctica en escenarios con recursos computacionales limitados.

1.3. Definición del problema

La evaluación clínica de quemaduras en la piel requiere identificar tanto la región lesionada como el grado de profundidad de la lesión. Esta tarea puede ser compleja debido a la variabilidad visual de las heridas, la presencia de distintos grados de severidad dentro de una misma imagen y la dificultad para delimitar con precisión los bordes de la lesión.

Aunque existen métodos basados en aprendizaje profundo para el análisis automático de quemaduras, muchos enfoques se centran únicamente en la segmentación de la lesión o en la clasificación global de la imagen. Esta separación limita su utilidad en escenarios clínicos donde se requiere conocer no solo si existe una quemadura, sino también su localización, extensión y grado de severidad.

El problema abordado en este trabajo consiste en desarrollar y evaluar un sistema capaz de segmentar y clasificar quemaduras de segundo y tercer grado en imágenes clínicas, manteniendo un equilibrio entre desempeño de segmentación, eficiencia computacional y viabilidad de despliegue móvil.

1.4. Hipótesis

H1. El modelo YOLO11m-seg puede segmentar regiones de quemadura de segundo y tercer grado con un desempeño competitivo frente a arquitecturas basadas en U-Net, especialmente en términos de cobertura de regiones lesionadas.

H2. El modelo YOLO11m-seg puede integrarse de forma viable en una aplicación móvil Android, manteniendo inferencia local y tiempos de ejecución compatibles con dispositivos móviles de diferentes capacidades computacionales.



1.5. Objetivos

1.5.1. Objetivo general

Desarrollar y evaluar un sistema basado en aprendizaje profundo para la segmentación y clasificación automática de quemaduras de segundo y tercer grado en imágenes clínicas, mediante la comparación de YOLO11m-seg con arquitecturas CNN basadas en U-Net y su posterior integración en una aplicación móvil Android.

Objetivos específicos

- Construir y preparar un conjunto de datos clínico propio de imágenes de quemaduras de segundo y tercer grado con anotaciones de segmentación
- Entrenar y optimizar un modelo YOLO11m-seg para la segmentación y clasificación automática de regiones de quemadura
- Implementar un *benchmark* comparativo *multi-seed* entre YOLO11m-seg y arquitecturas CNN basadas en U-Net, Attention U-Net y U-Net++
- Evaluar el desempeño de los modelos mediante métricas globales, métricas por clase, matrices de confusión y análisis estadístico
- Analizar la eficiencia computacional de las arquitecturas evaluadas considerando parámetros, FLOPs, latencia, FPS y consumo de memoria GPU
- Convertir e integrar el modelo YOLO11m-seg seleccionado en una aplicación móvil Android mediante TensorFlow Lite
- Evaluar el tiempo de ejecución del sistema móvil en dispositivos Android con diferentes capacidades computacionales



1.6. Justificación y originalidad

El desarrollo de herramientas automáticas para el análisis de quemaduras puede apoyar la evaluación visual inicial de las lesiones, especialmente en escenarios donde se requiere una estimación rápida de la región afectada y del grado de severidad. Los modelos de aprendizaje profundo permiten extraer información visual relevante a partir de imágenes clínicas y generar resultados interpretables mediante máscaras de segmentación, clases predichas y niveles de confianza.

La originalidad de este trabajo radica en la implementación y evaluación de un modelo YOLO11m-seg para la segmentación y clasificación automática de quemaduras de segundo y tercer grado, integrando detección, clasificación y segmentación de instancias dentro de una sola arquitectura. Además, se realiza una comparación experimental frente a arquitecturas CNN basadas en U-Net bajo un protocolo multi-seed, incorporando métricas globales, métricas por clase, matrices de confusión, análisis estadístico y evaluación de eficiencia computacional.

Además, el modelo seleccionado fue convertido e integrado en una aplicación móvil Android mediante TensorFlow Lite, permitiendo realizar inferencia local sin depender de servidores externos. Esto representa una contribución práctica al evaluar no solo el desempeño del modelo en escritorio, sino también su viabilidad de despliegue en dispositivos móviles con diferentes capacidades computacionales.

Capítulo 2: Fundamentación teórica

El análisis automático de quemaduras en la piel ha adquirido relevancia en los últimos años debido al crecimiento de las técnicas de visión por computadora y aprendizaje profundo aplicadas al ámbito médico. Estos enfoques buscan apoyar la evaluación clínica mediante el análisis de imágenes biomédicas, permitiendo identificar regiones lesionadas, estimar la severidad de la quemadura y asistir en tareas de segmentación y clasificación automática.

2.1. Clasificación clínica y profundidad de quemaduras

La piel humana está compuesta principalmente por dos capas: la epidermis, que constituye la capa externa y más delgada, y la dermis, una capa interna más gruesa formada por tejido conectivo elástico. En el contexto clínico de las quemaduras, estas lesiones se clasifican según la profundidad del daño tisular en quemaduras dérmicas superficiales, quemaduras dérmicas profundas y quemaduras de espesor total.

Las quemaduras dérmicas superficiales afectan principalmente la epidermis y se manifiestan como áreas enrojecidas, dolorosas y secas, frecuentemente acompañadas de ampollas. Las quemaduras dérmicas profundas comprometen parcialmente la dermis, presentando enrojecimiento intenso, formación de ampollas y dolor significativo. Por su parte, las quemaduras de espesor total afectan completamente la epidermis y la dermis, pudiendo extenderse al tejido subcutáneo y presentar una apariencia blanquecina o carbonizada [17].

De acuerdo con esta profundidad, las quemaduras se clasifican clínicamente en superficiales (primer grado), de espesor parcial superficial (SPT, segundo grado), de espesor parcial profundo (DPT, segundo grado) y de espesor completo (FT, tercer grado) [38]. Esta clasificación clínica es ampliamente utilizada como referencia para el análisis automatizado, esta clasificación se muestra en la Ilustración 2.1.

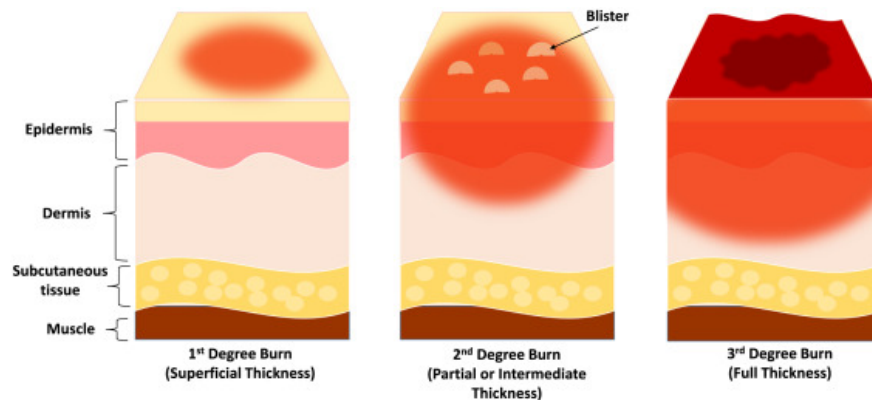


Ilustración 2.1: Clasificación de la profundidad de quemaduras [7]

2.1.1. Estimación de superficie corporal afectada (TBSA)

La superficie corporal total afectada (*Total Body Surface Area*, TBSA) corresponde a la proporción de superficie corporal comprometida por una quemadura respecto a la superficie corporal total del paciente. Esta estimación representa uno de los parámetros importantes en la evaluación clínica de pacientes con quemaduras, ya que proporciona información relevante para la determinación del manejo agudo de las heridas.

Una estimación precisa y rápida del porcentaje de TBSA es fundamental para calcular la reanimación con líquidos, el soporte nutricional y el pronóstico clínico del paciente [39]. En la práctica médica, este porcentaje se calcula considerando conjuntamente las quemaduras de segundo y tercer grado, mientras que las quemaduras de primer grado generalmente no se incluyen en la estimación.

Existen tres métodos principales utilizados por el personal médico para calcular el porcentaje de TBSA:

Regla de los Nueves de Wallace. Este método divide el cuerpo en regiones anatómicas proporcionales, donde cada región representa aproximadamente el 9 % de la superficie corporal total [40]. A partir de esta división se realiza una estimación rápida del área afectada por quemaduras. Sin embargo, este método presenta limitaciones en pacientes pediátricos y personas con sobrepeso debido a diferencias anatómicas y proporcionales.

Tabla de Lund y Browder. Es considerado el método estándar para la evaluación de TBSA en pacientes pediátricos. Este enfoque divide la superficie corporal en segmentos más pequeños y



ajusta las proporciones anatómicas de acuerdo con la edad del paciente. Por ejemplo, en un niño de un año, la cabeza representa aproximadamente el 19 % del TBSA, mientras que en un adulto representa cerca del 9 % [41]. El método ajusta las proporciones de regiones como cabeza, muslos y piernas conforme cambia la edad del paciente.

Método Palmar. Este método se utiliza principalmente para estimar quemaduras pequeñas, generalmente menores al 20 % del TBSA. Se toma como referencia la palma de la mano del paciente, la cual corresponde aproximadamente al 1 % de la superficie corporal total. Aunque estas técnicas continúan siendo ampliamente utilizadas en la práctica clínica, presentan limitaciones relacionadas con la subjetividad y la variabilidad entre evaluadores [42]. Además del porcentaje de TBSA, la profundidad de la quemadura constituye otro parámetro fundamental para la evaluación de la gravedad de las lesiones.

2.1.2. Importancia clínica de la segmentación

La segmentación precisa de las regiones afectadas por quemaduras tiene una relevancia clínica fundamental, ya que permite una evaluación localizada de la severidad de la lesión, facilitando la estimación del área comprometida y la identificación de zonas con diferente profundidad dentro de una misma herida. En la práctica clínica, las quemaduras pueden presentar variabilidad espacial significativa, lo que dificulta una evaluación global basada únicamente en la inspección visual.

En este contexto, la segmentación automática basada en imágenes médicas se propone como una herramienta de apoyo para mejorar la consistencia y objetividad de la evaluación inicial, especialmente en escenarios de atención primaria o prehospitalaria, donde la experiencia del personal puede ser limitada [7, 38].

2.2. Fundamentos de visión por computadora en imágenes médicas

La visión por computadora aplicada al análisis de imágenes médicas tiene como objetivo desarrollar métodos capaces de interpretar automáticamente información visual obtenida a partir de diferentes modalidades de imagen. En el contexto clínico, estas técnicas permiten apoyar tareas



como clasificación, detección y segmentación, facilitando el análisis de estructuras anatómicas y regiones de interés como anomalías, heridas, etc. El avance de los métodos basados en aprendizaje profundo ha permitido mejorar significativamente el desempeño en tareas de segmentación médica, particularmente mediante el uso de redes neuronales convolucionales, las cuales han demostrado una alta capacidad para extraer características espaciales complejas a partir de imágenes clínicas y decodificarlas de acuerdo a la tarea del marco de trabajo.

2.2.1. Redes neuronales convolucionales (CNNs)

El componente fundamental de una red neuronal convolucional es la capa de convolución, la cual emplea un conjunto de filtros entrenables para extraer características locales a partir de la imagen de entrada. Estos filtros se desplazan espacialmente sobre la imagen, permitiendo capturar patrones como bordes, texturas y estructuras relevantes, manteniendo invariancia a la traslación.

Las redes neuronales convolucionales combinan operaciones de convolución, funciones de activación no lineales y capas de agrupamiento (pooling), lo que permite reducir la dimensionalidad espacial y aumentar la robustez frente a pequeñas variaciones en la posición de los patrones. Estas propiedades han convertido a las CNN en el enfoque predominante para tareas de visión por computadora, particularmente en el análisis de imágenes médicas, donde las lesiones presentan variabilidad en tamaño, forma y localización [43,44].

Formulación matemática de las redes neuronales convolucionales Sea x_{ij} el vector de datos en la posición espacial (i, j) de una capa determinada, y y_{ij} la salida correspondiente en la capa siguiente. Las capas convolucionales, de agrupamiento y activación pueden describirse de manera unificada mediante la siguiente expresión:

$$y_{ij} = f_{k,s} \left(\{x_{s \cdot i + \delta_i, s \cdot j + \delta_j} \mid 0 \leq \delta_i, \delta_j \leq k\} \right) \quad (2.1)$$

donde k representa el tamaño del kernel, s el factor de stride o submuestreo, y $f_{k,s}$ define el tipo de operación aplicada. En el caso de una capa convolucional, $f_{k,s}$ corresponde a una multiplicación entre el kernel entrenable y el parche de entrada; para una capa de agrupamiento, puede representar una operación de máximo o promedio; mientras que en el caso de una función de activación, corresponde a una no linealidad aplicada elemento a elemento.



Pooling y funciones de activación Las redes neuronales convolucionales cuentan con una capa de pooling; estas condensan la información más importante obtenida en las capas de convolución, simplificando la información recogida. Hay varias formas de condensar la información, por ejemplo, **max_pooling**, donde cada grupo de puntos de entrada se transforma en el valor máximo del grupo [45]. **ReLU**: A diferencia de las unidades binarias, las unidades lineales rectificadas conservan información sobre las intensidades relativas a medida que la información viaja a través de múltiples capas de detectores de características [46]. La función ReLU se define como:

$$\text{ReLU}(z) = \max(0, z) \quad (2.2)$$

Transfer learning El aprendizaje por transferencia (transfer learning) consiste en reutilizar modelos previamente entrenados en grandes conjuntos de datos para adaptarlos a nuevas tareas con información limitada. En el contexto de imágenes médicas, esta estrategia resulta especialmente útil debido a la escasez de datos anotados, permitiendo mejorar la convergencia y el desempeño del modelo [43].

2.2.2. Segmentación de imágenes médicas

2.2.2.1. Segmentación semántica vs segmentación de instancia

La segmentación de imágenes en visión por computadora puede abordarse desde distintos enfoques, entre los que destacan la segmentación semántica y la segmentación por instancia. En la segmentación semántica, a cada píxel de la imagen se le asigna una etiqueta de clase, sin importar a que instancia pertenezca [3]. Por otro lado, la segmentación por instancia permite identificar y segmentar individualmente cada objeto presente en la imagen, incluso cuando múltiples objetos pertenecen a la misma clase [47].

Esta distinción resulta especialmente relevante en el dominio de imágenes médicas. Ronneberger et al. [3] señalan que uno de los principales desafíos en tareas de segmentación celular es específicamente la *separación de objetos tocantes de la misma clase*, este problema lo abordan mediante una función de pérdida ponderada que asigna mayor peso a los píxeles de borde entre instancias adyacentes. Este planteamiento pone de manifiesto que, aún en redes orientadas a



segmentación semántica, la distinción entre instancias constituye un problema no superficial.

2.2.3. Arquitecturas *encoder–decoder*

Las arquitecturas *encoder–decoder* constituyen el paradigma dominante en segmentación semántica de imágenes médicas. En este esquema, el *encoder* reduce progresivamente la resolución espacial para extraer representaciones de alto nivel, mientras que el *decoder* recupera la resolución original para producir el mapa de segmentación. Las conexiones de salto (*skip connections*) entre ambas ramas permiten recuperar detalles espaciales perdidos durante el submuestreo [22].

2.3. Modelos supervisados para segmentación

Los modelos supervisados de segmentación utilizan imágenes anotadas manualmente, validadas por personal médico experto en el caso de imágenes médicas, para aprender la relación entre las características visuales de entrada y las regiones de interés presentes en la imagen. Estos enfoques han demostrado un alto desempeño en tareas de delimitación de estructuras anatómicas y lesiones, especialmente mediante arquitecturas basadas en aprendizaje profundo. Dependiendo de la formulación del problema, los modelos pueden abordar tareas de segmentación semántica o segmentación de instancias, permitiendo identificar regiones anómalas y asociarlas a categorías o clases específicas.

2.3.1. Modelos basados en CNN

Las arquitecturas basadas en redes neuronales convolucionales (CNN) con estructura *encoder–decoder* constituyen el paradigma dominante en segmentación semántica de imágenes médicas. En este trabajo se emplean tres arquitecturas de esta familia como líneas base de comparación: U-Net, U-Net++ y Attention U-Net. A continuación se describe cada una.

U-Net. U-Net fue propuesta por Ronneberger et al. en 2015 [3] como una arquitectura CNN en el marco de visión por computadora diseñada específicamente para la segmentación de imágenes biomédicas. Su principal aportación fue demostrar que es posible obtener segmentaciones precisas



a nivel de píxel incluso con conjuntos de entrenamiento de tamaño reducido, que es una de las características habituales en el dominio de imágenes médicas.

La arquitectura adopta una estructura simétrica en forma de “U” compuesta por dos caminos principales [3]:

- **Camino codificador (encoder):** aplica bloques de convoluciones seguidos de operaciones de *max-pooling* para reducir progresivamente la resolución espacial e incrementar el número de mapas de características, capturando contexto semántico de alto nivel.
- **Camino decodificador (decoder):** recupera la resolución espacial mediante convoluciones transpuestas (*upsampling*), reconstruyendo la máscara de segmentación de forma progresiva.

El elemento central de U-Net son las **conexiones de salto** (*skip connections*), que concatenan los mapas de características del encoder directamente con los correspondientes de nivel de resolución del decoder. Estas conexiones permiten que el decoder recupere detalles que se habrían perdido durante el proceso de submuestreo, mejorando la precisión en la localización de los bordes de los objetos segmentados [3].

La operación de convolución en cada bloque puede expresarse como:

$$\mathbf{x}^{(l+1)} = \sigma\left(W^{(l)} * \mathbf{x}^{(l)} + b^{(l)}\right) \quad (2.3)$$

donde $\mathbf{x}^{(l)}$ es el mapa de características en la capa l , $W^{(l)}$ y $b^{(l)}$ son los pesos y sesgos del filtro convolucional, $*$ denota la operación de convolución y σ una función de activación no lineal (típicamente ReLU). La estructura general de la arquitectura se ilustra en la Figura 2.1.

U-Net++. U-Net++ fue presentada por Zhou et al. en 2018 [4] como una extensión de U-Net enfocada en reducir la *brecha semántica* entre las características generadas por el encoder y el decoder. En U-Net original, las conexiones de salto transfieren directamente mapas de características del encoder al decoder, fusionando representaciones que pueden ser semánticamente muy distintas entre sí. U-Net++ aborda esta limitación mediante **caminos de salto anidados y densos** entre las dos ramas de la red.

La arquitectura introduce nodos intermedios $\mathbf{x}^{i,j}$ entre el encoder y el decoder, definidos como [4]:

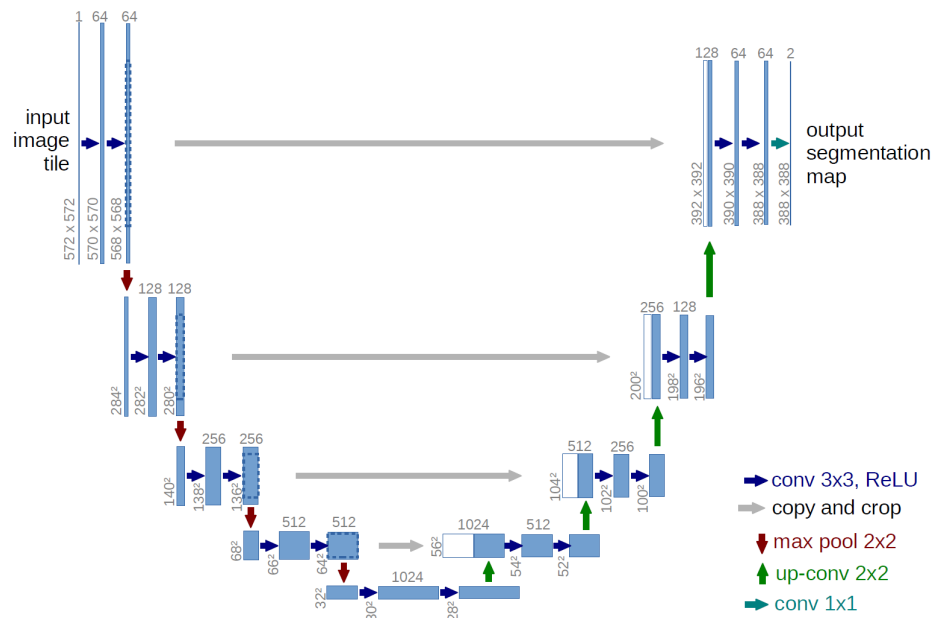


Figura 2.1: Arquitectura U-Net. El camino codificador (izquierda) extrae características mediante convoluciones y *max-pooling*, mientras que el camino decodificador (derecha) recupera la resolución espacial. Las flechas grises representan las conexiones de salto (*skip connections*) que concatenan mapas de características del encoder al decoder. Adaptado de [3].

$$\mathbf{x}^{i,j} = \mathcal{H}\left(\mathbf{x}^{i,j-1}, \left[\mathbf{x}^{i+1,k}\right]_{k=0}^{j-1}\right) \quad (2.4)$$

donde i es el índice de profundidad (nivel de resolución), j es el índice de densidad a lo largo del camino de salto, $\mathcal{H}$ representa un bloque convolucional con activación no lineal, y $[\cdot]$ indica concatenación de los nodos previos en el mismo nivel de resolución.

Gracias a estos caminos densos, las características del encoder son enriquecidas de forma gradual antes de ser fusionadas con el decoder, de modo que ambos representan información semánticamente más similar [4]. Adicionalmente, U-Net++ incorpora **supervisión profunda** (*deep supervision*), que permite obtener predicciones intermedias en distintos niveles de profundidad y facilita el proceso de entrenamiento del modelo. En inferencia, puede utilizarse un subconjunto de la red (*model pruning*), ajustando la profundidad según los requisitos de precisión o eficiencia. La estructura de caminos anidados se ilustra en la Figura 2.2.

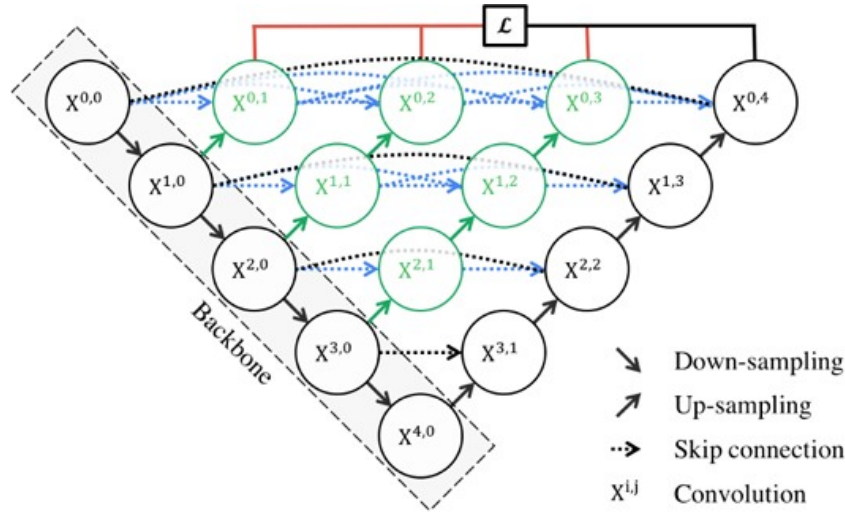


Figura 2.2: Arquitectura U-Net++. Los nodos intermedios (círculos verdes) entre el encoder y el decoder forman caminos de salto anidados y densos que mejoran la relación semántica. La supervisión profunda se aplica en los nodos de salida de cada columna del decoder. Adaptado de [4].

Attention U-Net. Attention U-Net fue propuesta por Oktay et al. en 2018 [5] como una variante de U-Net que incorpora **puertas de atención** (*Attention Gates*, AGs) dentro de las conexiones de salto. El objetivo principal es permitir que el modelo aprenda automáticamente cuáles regiones



espaciales de la imagen son más importantes para la segmentación, reduciendo la influencia de información irrelevante [5].

Las puertas de atención reciben dos entradas: las características provenientes del encoder y una señal de contexto (*gating signal*) generada en niveles más profundos del decoder. A partir de ambas señales se calculan coeficientes de atención definidos como [5]:

$$\alpha_i = \sigma_2(\psi^\top (\sigma_1(W_x \mathbf{x}_i + W_g \mathbf{g} + b_g)) + b_\psi) \quad (2.5)$$

donde $\mathbf{x}_i$ es el vector de características del encoder en la posición espacial i , $\mathbf{g}$ es la señal de contexto del decoder (señal de *gating*), W_x y W_g son transformaciones lineales aprendidas, σ_1 es la función de activación ReLU, σ_2 es la función sigmoide (que produce coeficientes $\alpha_i \in (0, 1)$), y ψ es un vector de pesos aprendido.

Los coeficientes α_i funcionan como pesos de atención que escalan las características del encoder antes de concatenarlas con el decoder:

$$\hat{\mathbf{x}}_i = \alpha_i \cdot \mathbf{x}_i \quad (2.6)$$

De esta manera, las regiones relevantes para la segmentación reciben mayor peso, mientras que las regiones menos importantes son atenuadas [5]. Esto permite que el modelo concentre su atención en estructuras específicas de interés, especialmente en regiones pequeñas o con bajo contraste.

Los autores reportan que la incorporación de puertas de atención mejora el desempeño de U-Net en tareas de segmentación médica sin incrementar significativamente el costo computacional. La estructura general del mecanismo de atención utilizado en Attention U-Net se muestra en la Figura 2.3.

Estas arquitecturas constituyen las líneas base utilizadas para comparar el modelo propuesto en este trabajo, debido a que representan enfoques ampliamente utilizados en segmentación médica supervisada basada en CNN.

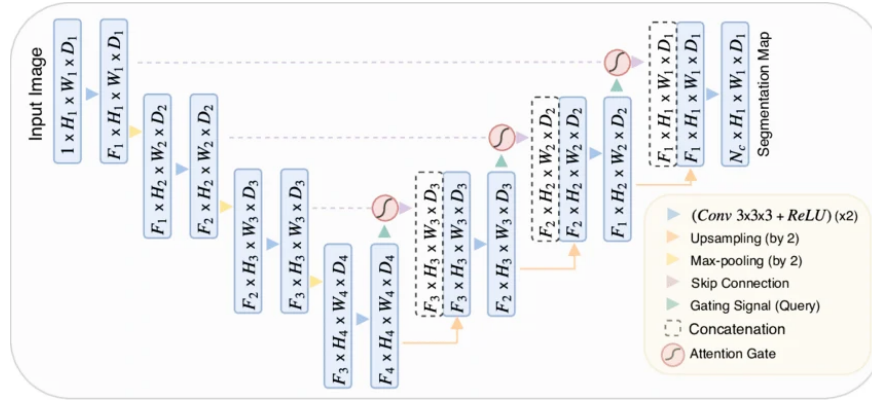


Figura 2.3: Arquitectura Attention U-Net. Las puertas de atención (AG) se insertan en las conexiones de salto y ponderan espacialmente las características del encoder según la señal de contexto proveniente del decoder, suprimiendo regiones irrelevantes para la segmentación. Adaptado de [5].

2.3.2. Enfoques basados en YOLO para segmentación médica

2.3.2.1. YOLO para tareas médicas.

Redmon et al. propusieron YOLO (You Only Look Once) como un modelo de detección de objetos basado en una sola etapa (*one-stage*), donde la localización y clasificación se realizan directamente en una única inferencia de la red neuronal convolucional [48]. A diferencia de los enfoques tradicionales de múltiples etapas, YOLO formula la detección como un problema de regresión directa desde la imagen de entrada con cajas delimitadoras y probabilidades de clase.

Gracias a esta estrategia, los modelos YOLO alcanzan tiempos de inferencia reducidos y alta eficiencia computacional, características que han favorecido su uso en aplicaciones en tiempo real, sistemas embebidos y dispositivos móviles. Las versiones recientes de YOLO han extendido sus capacidades más allá de la detección de objetos, incorporando tareas de segmentación de instancias mediante la generación simultánea de cajas delimitadoras, etiquetas de clase y máscaras de segmentación. Esto permite integrar localización y segmentación dentro de una misma arquitectura.

En el área médica, diferentes trabajos han explorado el uso de modelos YOLO para tareas de detección y segmentación en imágenes clínicas, obteniendo resultados competitivos debido a su rapidez de inferencia y capacidad para procesar imágenes de manera eficiente [49, 50]. Estas características han convertido a YOLO en una alternativa relevante para aplicaciones médicas que requieren procesamiento en tiempo real o implementación en dispositivos con recursos compu-



tacionales limitados.

Las arquitecturas más recientes de YOLO para segmentación incorporan módulos especializados para la generación de máscaras, permitiendo segmentar regiones de interés mientras se mantiene la eficiencia computacional característica de los modelos *one-stage*. Debido a ello, YOLO ha sido utilizado en tareas médicas como detección de lesiones, análisis de estructuras anatómicas y segmentación automática en imágenes clínicas.

YOLOv11-seg. YOLOv11-seg mantiene el enfoque *one-stage* de la familia YOLO, propuesto originalmente por Redmon et al. [48], y extiende la arquitectura hacia tareas de segmentación de instancias mediante predicción multiescala y generación de máscaras prototipo [6].

La arquitectura de YOLOv11-seg se compone principalmente de tres módulos: *backbone*, *neck* y *head*.

El módulo *backbone* se encarga de extraer características jerárquicas de la imagen de entrada mediante bloques convolucionales y bloques C3K2, los cuales permiten refinar y combinar características en diferentes niveles de profundidad. En las etapas finales del *backbone* se incorporan los módulos SPPF (*Spatial Pyramid Pooling Fast*) y C2PSA (*Cross Stage Partial with Position-Sensitive Attention*). El bloque SPPF permite aumentar el campo receptivo y capturar información contextual a diferentes escalas, mientras que C2PSA incorpora mecanismos de atención para mejorar la representación espacial de las características extraídas.

Posteriormente, el módulo *neck* combina información proveniente de diferentes escalas espaciales mediante operaciones de *upsampling* y concatenación. La capa *Upsample* incrementa la resolución de los mapas de características utilizando interpolación por vecino más cercano, mientras que el módulo *Concat* fusiona características profundas y superficiales sin modificar la resolución espacial del mapa resultante. Esta estrategia permite conservar información semántica y detalles espaciales relevantes para la segmentación.

Finalmente, el *head* genera simultáneamente las predicciones de detección y los coeficientes de máscara utilizados para segmentación. El modelo produce cajas delimitadoras, probabilidades de clase y coeficientes asociados a máscaras prototipo (*prototype masks*), los cuales posteriormente son combinados para obtener la máscara final de cada instancia segmentada. El módulo Proto genera dichas máscaras prototipo siguiendo un enfoque similar al utilizado en modelos de segmentación



de instancias tipo YOLACT [51]. La Figura 2.4 muestra la arquitectura del modelo.

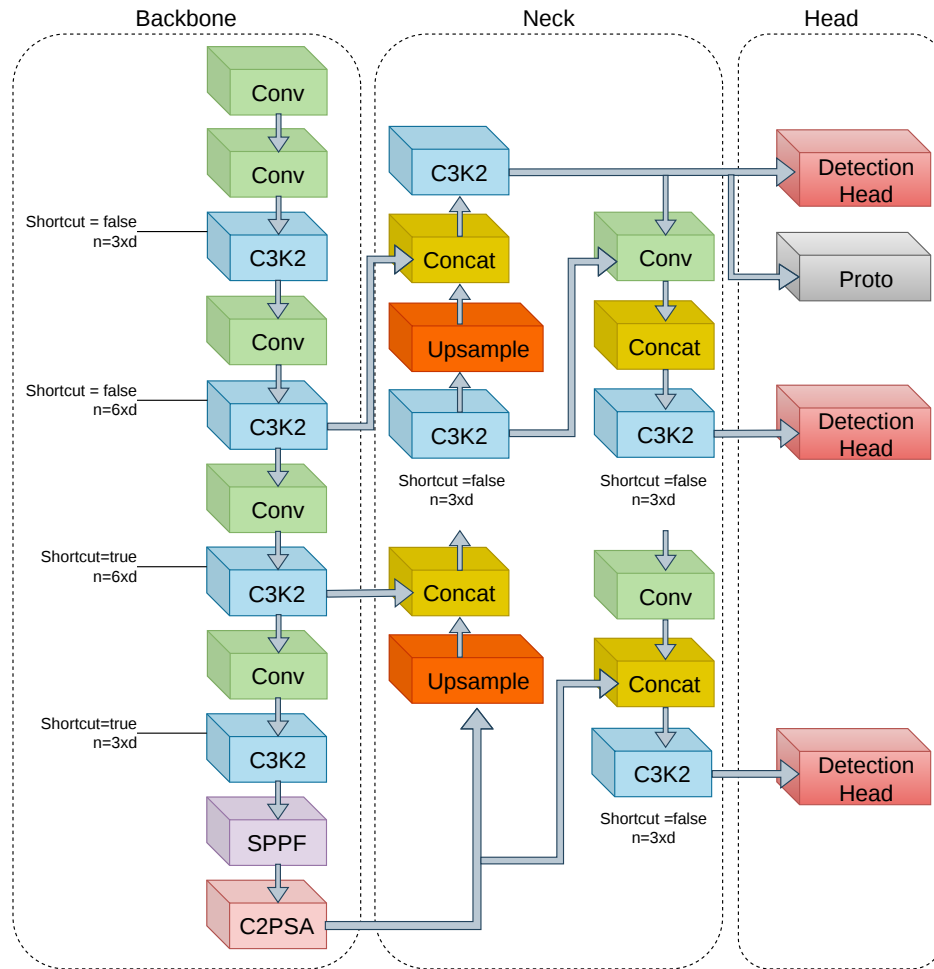


Figura 2.4: Arquitectura general del modelo YOLOv11-seg empleado en este trabajo. El diagrama muestra los módulos *backbone*, *neck* y *head* utilizados para la extracción de características, fusión multiescala y segmentación de instancias. Elaboración propia con base en [6].

A diferencia de los enfoques basados únicamente en segmentación semántica, YOLOv11-seg permite identificar regiones individuales dentro de la imagen, conservando simultáneamente información de localización y segmentación. Esta característica resulta especialmente útil en aplicaciones médicas donde es necesario delimitar lesiones específicas dentro de imágenes clínicas.

Índice de confianza en modelos YOLO. En modelos de detección basados en YOLO, el índice de confianza representa el grado de certeza asociado a una predicción de una de las clases realizada



por el modelo [52]. Este valor combina la probabilidad de existencia de un objeto en una región determinada y la probabilidad condicional de pertenencia a una clase específica.

De forma general, el índice de confianza puede expresarse como:

$$S_{\text{conf}} = P(O) \times \max_i P(C_i|O) \quad (2.7)$$

donde S_{conf} representa el puntaje de confianza asociado a la detección, $P(O)$ corresponde a la probabilidad de existencia del objeto y $P(C_i|O)$ representa la probabilidad condicional de que el objeto pertenezca a la clase C_i .

Valores altos de confianza indican una mayor certeza del modelo respecto a la detección realizada, mientras que valores bajos pueden asociarse a predicciones inciertas o regiones ambiguas.

Regresión de cajas delimitadoras. En modelos YOLO, la localización de objetos se realiza mediante la predicción de cajas delimitadoras (*bounding boxes*), las cuales representan la posición espacial 2D del objeto detectado en la imagen. Cada caja se define mediante coordenadas asociadas a su centro, ancho y altura:

$$B = (u_c, v_c, w, h) \quad (2.8)$$

donde u_c y v_c representan las coordenadas del centro de la caja delimitadora, mientras que w y h corresponden al ancho y alto de la detección, respectivamente, ampliamente utilizada en modelos de detección basados en YOLO [48].

2.3.3. Funciones de pérdida

En esta sección se presentan las principales funciones de pérdida relacionadas con las tareas de clasificación, detección y segmentación utilizadas durante el entrenamiento de modelos de aprendizaje profundo. Estas funciones permiten optimizar de manera conjunta la clasificación de las lesiones, la localización de regiones de interés y la segmentación a nivel de píxel.

Las formulaciones mostradas corresponden a funciones ampliamente utilizadas en la literatura de visión por computadora y segmentación médica [44, 53–55].



Pérdida de clasificación (Cross-Entropy). La pérdida de entropía cruzada categórica se emplea para optimizar la tarea de clasificación multiclase. Esta función penaliza la discrepancia entre la distribución de probabilidad predicha por el modelo y la distribución real de las clases:

$$\mathcal{L}_{CE} = - \sum_{c=1}^{N_c} y_c \log(p_c) \quad (2.9)$$

donde N_c representa el número total de clases consideradas en el problema de segmentación, y_c es la etiqueta real asociada a la clase c , y p_c corresponde a la probabilidad predicha por el modelo para dicha clase. Esta función es ampliamente utilizada en problemas de clasificación supervisada [44].

Pérdida Dice. La pérdida Dice es ampliamente utilizada en segmentación médica debido a su capacidad para manejar desbalances entre clases y optimizar directamente el grado de superposición entre regiones segmentadas.

$$\mathcal{L}_{Dice} = 1 - \frac{2|P \cap G|}{|P| + |G|} \quad (2.10)$$

donde P representa el conjunto de píxeles predichos como pertenecientes a la región de interés y G el conjunto de píxeles correspondientes a la anotación real. Esta formulación ha demostrado ser efectiva en aplicaciones de segmentación médica [53, 54].

Pérdida de segmentación combinada. En tareas de segmentación médica, es común emplear funciones de pérdida compuestas que combinan entropía cruzada y pérdida Dice. Esta combinación permite aprovechar la estabilidad de la entropía cruzada y la sensibilidad al solapamiento de la pérdida Dice:

$$\mathcal{L}_{seg} = \lambda_{ce} \mathcal{L}_{CE} + \lambda_{dice} \mathcal{L}_{Dice} \quad (2.11)$$

donde λ_{ce} y λ_{dice} son factores de ponderación que ajustan la contribución relativa de cada término en la función de pérdida total.

Distribution Focal Loss (DFL). La *Distribution Focal Loss (DFL)* se utiliza para mejorar la precisión en la regresión de cajas delimitadoras. Esta función modela la localización de los bordes



mediante distribuciones discretas de probabilidad, permitiendo una estimación más precisa de las coordenadas de los objetos detectados.

$$\mathcal{L}_{DFL} = - \sum_{i=1}^N \sum_{k=1}^K \alpha_{i,k} (1 - p_{i,k})^\gamma \log(p_{i,k}) \quad (2.12)$$

donde N es el número de muestras, K el número de puntos discretos utilizados para modelar la distribución, $p_{i,k}$ representa la probabilidad predicha para el punto k de la muestra i , $\alpha_{i,k}$ es un factor de ponderación y γ controla el énfasis en muestras difíciles. Esta función fue propuesta por Li et al. [55] y es utilizada en modelos modernos de detección y segmentación.

Pérdida de localización de cajas. La pérdida de localización de cajas delimitadoras tiene como objetivo minimizar la diferencia entre las cajas predichas por el modelo y las cajas reales anotadas en el conjunto de datos. Esta pérdida optimiza la posición y el tamaño de las regiones detectadas durante el entrenamiento.

De forma general, la pérdida de localización puede expresarse como:

$$\mathcal{L}_{box} = \lambda_{center} \mathcal{L}_{center} + \lambda_{wh} \mathcal{L}_{wh} + \lambda_{conf} \mathcal{L}_{conf} \quad (2.13)$$

donde $\mathcal{L}_{center}$ penaliza diferencias en las coordenadas centrales de las cajas, $\mathcal{L}_{wh}$ optimiza el ancho y altura de las cajas delimitadoras, y $\mathcal{L}_{conf}$ corresponde al error asociado al índice de confianza de detección.

Para estabilizar el entrenamiento y reducir problemas asociados a gradientes extremos, algunos enfoques basados en YOLO utilizan funciones de pérdida Smooth L1 para la regresión de coordenadas [52].

En versiones modernas de la familia YOLO, incluyendo YOLOv11-seg, la optimización de cajas delimitadoras incorpora además métricas basadas en *Intersection over Union* (IoU), permitiendo mejorar la precisión espacial de las detecciones.

Pérdida total del modelo. La función de pérdida global empleada en modelos modernos de la familia YOLO para tareas de detección y segmentación puede expresarse como:

$$\mathcal{L}_{total} = \mathcal{L}_{box} + \mathcal{L}_{cls} + \mathcal{L}_{seg} + \mathcal{L}_{DFL} \quad (2.14)$$



donde $\mathcal{L}_{box}$ corresponde a la pérdida de localización de las cajas delimitadoras, $\mathcal{L}_{cls}$ a la pérdida de clasificación, $\mathcal{L}_{seg}$ a la pérdida de segmentación y $\mathcal{L}_{DFL}$ a la pérdida de refinamiento de bordes. Esta formulación permite optimizar de manera conjunta la detección, clasificación y segmentación de las lesiones por quemaduras.

Las formulaciones anteriores representan funciones de pérdida ampliamente utilizadas en modelos de detección y segmentación profunda. En la implementación práctica de YOLOv11-seg, varias de estas funciones se integran internamente dentro del framework Ultralytics para optimizar simultáneamente la detección, clasificación y segmentación de objetos.

2.3.4. Ventajas y limitaciones de los modelos supervisados de segmentación

Los enfoques supervisados basados en aprendizaje profundo han demostrado un alto desempeño en tareas de análisis de imágenes médicas cuando se dispone de conjuntos de datos suficientemente grandes y representativos. En particular, modelos como YOLO destacan por su eficiencia computacional, capacidad de inferencia en tiempo real y facilidad de implementación, características que los hacen adecuados para aplicaciones clínicas y dispositivos con recursos limitados.

Sin embargo, estos enfoques también presentan limitaciones importantes en escenarios clínicos reales, donde la cantidad de datos anotados suele ser reducida y existen variaciones entre dominios, como diferencias en resolución, iluminación, contraste o condiciones de adquisición de las imágenes. Estas variaciones pueden afectar la capacidad de generalización del modelo hacia nuevos entornos o poblaciones de pacientes.

A pesar de estas limitaciones, los modelos supervisados continúan siendo una de las estrategias más utilizadas en tareas de segmentación médica debido a su alto desempeño cuando se cuenta con datos adecuadamente anotados y configuraciones de entrenamiento apropiadas.

2.4. Métricas de evaluación

Las métricas de evaluación se utilizan para analizar el desempeño de los modelos de detección y segmentación empleados en este trabajo. En el contexto de segmentación de imágenes médicas, estas métricas permiten cuantificar el grado de coincidencia entre las máscaras predichas



por el modelo y las máscaras de referencia anotadas manualmente.

2.4.1. Coeficiente Dice

El coeficiente Dice mide el grado de solapamiento entre una máscara predicha y una máscara de referencia. A diferencia de la exactitud por píxel, esta métrica no considera los verdaderos negativos, por lo que evita que el alto número de píxeles de fondo infle artificialmente el desempeño del modelo. Por esta razón, resulta adecuada en problemas con desbalance de clases, como ocurre frecuentemente en segmentación médica [56].

Expresado en términos de verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN) a nivel de píxel, el coeficiente Dice se define como:

$$\text{Dice} = \frac{2 |P \cap G|}{|P| + |G|} = \frac{2 \text{ TP}}{2 \text{ TP} + \text{FP} + \text{FN}} \quad (2.15)$$

donde P representa la máscara predicha y G la máscara de referencia. En este contexto, los verdaderos positivos corresponden a los píxeles de lesión correctamente segmentados, los falsos positivos a los píxeles clasificados erróneamente como lesión y los falsos negativos a los píxeles de lesión que no fueron detectados por el modelo.

2.4.2. Intersección sobre Unión (IoU)

La métrica de Intersección sobre Unión (*Intersection over Union*) (IoU), también denominada índice de Jaccard, cuantifica el grado de solapamiento entre la región predicha y la región de referencia [57]. Es una de las métricas más utilizadas en tareas de segmentación semántica, segmentación de instancias y detección de objetos [58] [20].

Formalmente, el IoU se calcula como:

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad (2.16)$$

El IoU penaliza tanto la sobresegmentación como la infrasegmentación, ya que considera en el denominador los píxeles predichos incorrectamente y los píxeles reales no detectados. Además, mantiene una relación matemática directa con el coeficiente Dice:



$$\text{Dice} = \frac{2 \text{IoU}}{1 + \text{IoU}} \quad (2.17)$$

Esta relación se cumple cuando ambas métricas se calculan sobre una misma máscara binaria individual. En escenarios experimentales donde los resultados se reportan como promedios sobre múltiples imágenes o clases, los valores agregados de Dice e IoU pueden no mantener exactamente esta relación matemática.

El coeficiente Dice y la métrica IoU mantienen una relación matemática directa cuando se calculan sobre una misma máscara binaria. En este contexto, en la evaluación experimental los valores reportados corresponden a promedios sobre el conjunto de prueba, por lo que dicha relación puede no cumplirse exactamente en las métricas finales reportadas.

2.4.3. Precisión y exhaustividad (*Precision / Recall*)

La precisión (*Precisión*) y la exhaustividad (*Recall*) permiten evaluar el comportamiento del modelo respecto a las detecciones correctas, las falsas alarmas y las regiones no detectadas [59].

La precisión expresa la proporción de píxeles predichos como lesión que corresponden efectivamente a regiones de interés:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.18)$$

Por lo tanto, una precisión alta indica que el modelo genera pocas detecciones falsas.

La exhaustividad indica la proporción de píxeles de lesión reales que fueron correctamente detectados:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.19)$$

Un valor alto de exhaustividad indica que el modelo omite pocas regiones afectadas. En aplicaciones clínicas, esta métrica es especialmente relevante, ya que los falsos negativos pueden implicar que una región lesionada no sea detectada. En problemas de segmentación multiclase, estas métricas pueden calcularse tanto de forma global como por clase. Las métricas globales permiten evaluar la capacidad general del modelo para segmentar regiones de interés frente al



fondo, mientras que las métricas por clase permiten analizar la correcta asignación semántica de cada categoría clínica.

Finalmente, el coeficiente Dice también puede interpretarse como una variante del F1-score aplicada a segmentación binaria, ya que combina precisión y exhaustividad en una sola métrica.

En este trabajo se priorizaron métricas de solapamiento y sensibilidad, como Dice, IoU y Recall, debido a su uso extendido en segmentación médica y a la necesidad de evaluar adecuadamente la cobertura espacial de las regiones de quemadura, minimizando la omisión de áreas lesionadas.

2.4.4. Tiempo de inferencia y eficiencia computacional

Además de la calidad de segmentación, la eficiencia computacional es importante para evaluar la viabilidad de los modelos en dispositivos con recursos limitados [60]. En este trabajo se reportan las siguientes métricas de eficiencia:

Número de parámetros. Representa el total de pesos ajustables del modelo. Un mayor número de parámetros suele incrementar la capacidad de representación del modelo, aunque también aumenta los requerimientos de memoria y cómputo.

FLOPs (*Floating Point Operations*). Cuantifica el número total de operaciones en punto flotante requeridas para procesar una imagen de entrada. Permite comparar la complejidad computacional entre arquitecturas de forma independiente del hardware utilizado [60]. Nótese que FLOPs (número total de operaciones) es distinto de FLOPS con mayúscula, que denota la velocidad del hardware en operaciones por segundo.

Latencia (*Latency*). Corresponde al tiempo requerido por el modelo para generar una predicción a partir de una imagen de entrada. La latencia suele reportarse mediante estadísticas descriptivas obtenidas a partir de múltiples inferencias consecutivas, incluyendo media, desviación estándar, valor mínimo, valor máximo y mediana, generalmente expresadas en milisegundos.

FPS (*Frames Per Second*). Representa el número de imágenes que el modelo puede procesar por segundo y se obtiene como:

$$\text{FPS} = \frac{1000}{\mu_{\text{ms}}} \quad (2.20)$$



donde μ_{ms} es la latencia media expresada en milisegundos. Valores altos de FPS reflejan una mayor velocidad de procesamiento.

VRAM (Video Random Access Memory). Indica la cantidad de memoria de la GPU utilizada durante la inferencia. Esta métrica es importante para determinar si un modelo puede ejecutarse en hardware con capacidades de memoria limitadas.

2.5. Análisis estadístico

El análisis estadístico permite evaluar si las diferencias observadas entre los modelos corresponden a variaciones reales en el desempeño o si pueden atribuirse al azar. En tareas de segmentación médica, donde los resultados suelen obtenerse a partir de múltiples ejecuciones experimentales, el uso de pruebas estadísticas resulta importante para realizar comparaciones más robustas entre arquitecturas. En este trabajo se emplearon pruebas no paramétricas para analizar las diferencias en las métricas de segmentación obtenidas por los modelos evaluados.

2.5.1. Pruebas no paramétricas (Kruskal-Wallis, Dunn, Bonferroni)

Para comparar el desempeño entre los distintos modelos se utilizaron pruebas estadísticas no paramétricas, ya que las métricas obtenidas en un número reducido de experimentos pueden no seguir una distribución normal [61].

Prueba de Kruskal-Wallis. Es el equivalente no paramétrico del análisis de varianza de un factor (ANOVA de una vía). Permite determinar si existen diferencias estadísticamente significativas entre múltiples grupos independientes [62]. El estadístico de prueba se define como:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1) \quad (2.21)$$

donde N es el número total de observaciones, k el número de grupos, n_i el tamaño del i -ésimo grupo y R_i la suma de rangos del i -ésimo grupo. Un resultado significativo ($p < 0,05$) indica que al menos un grupo difiere estadísticamente de los demás, sin especificar cuáles.

Prueba post hoc de Dunn. Ante un resultado significativo en la prueba de Kruskal-Wallis, se aplica la prueba de Dunn para identificar qué pares de grupos difieren entre sí [63]. Esta prueba



realiza comparaciones por pares utilizando los mismos rangos calculados en la prueba de Kruskal-Wallis, manteniendo el mismo criterio de comparación basado en rangos. El estadístico de la comparación entre los grupos i y j es:

$$z_{ij} = \frac{\bar{R}_i - \bar{R}_j}{\sqrt{\frac{N(N+1)}{12} \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}} \quad (2.22)$$

donde $\bar{R}_i$ y $\bar{R}_j$ son las medias de los rangos de cada grupo.

Corrección de Bonferroni. Para controlar la tasa de error de tipo I en comparaciones múltiples, se aplica la corrección de Bonferroni, que ajusta el nivel de significancia α dividiéndolo entre el número total de comparaciones m realizadas [64]:

$$\alpha_{\text{ajustado}} = \frac{\alpha}{m} \quad (2.23)$$

Equivalentemente, los p -valores obtenidos se multiplican por m antes de compararse con el nivel α original. Esta corrección disminuye el riesgo de detectar diferencias significativas de manera errónea cuando se realizan múltiples comparaciones.

2.6. Despliegue en dispositivos móviles

El despliegue de modelos de aprendizaje profundo en dispositivos móviles ha cobrado relevancia debido a la necesidad de ejecutar inferencia de manera local, sin depender de servidores externos o conexión a la nube. En aplicaciones clínicas como la evaluación de quemaduras, esto permite utilizar el sistema incluso en entornos con acceso limitado a infraestructura tecnológica [65].

El principal inconveniente es, que el despliegue en dispositivos móviles impone restricciones significativas sobre los modelos. A diferencia de los servidores con GPUs dedicadas de alto rendimiento, los dispositivos móviles cuentan con recursos más limitados en términos de cómputo, memoria y batería [66]. Debido a estas limitaciones, muchos modelos de gran tamaño requieren estrategias de optimización para ejecutarse de manera eficiente en este tipo de hardware.

Restricciones Las principales limitaciones que deben considerarse al desplegar modelos en dispositivos móviles son [67, 68]:



- **Memoria limitada:** los modelos de *deep learning* pueden requerir cientos de megabytes de memoria durante la inferencia. La mayoría de los dispositivos móviles dispone de pocos gigabytes de RAM compartidos entre el sistema operativo y las aplicaciones.
- **Capacidad de cómputo reducida:** los dispositivos móviles carecen de GPUs dedicadas de alto rendimiento. La inferencia suele ejecutarse en CPUs de propósito general o, en dispositivos más recientes, en unidades de procesamiento neuronal (NPU).
- **Consumo energético:** las operaciones de redes neuronales profundas son intensivas en cómputo y generan un consumo elevado de energía, lo que acorta la duración de la batería del dispositivo.
- **Latencia:** en aplicaciones clínicas, se requiere que el tiempo de inferencia sea suficientemente bajo para ofrecer retroalimentación en tiempo real al personal de salud.

Técnicas de optimización. Existen diversas estrategias para optimizar la ejecución de modelos en dispositivos móviles [69, 70].

En particular, la cuantización permite reducir el tamaño del modelo y acelerar la inferencia mediante representaciones numéricas de menor precisión, como INT8 o FP16, siendo una de las técnicas más utilizadas en despliegue móvil.

Frameworks de despliegue. Existen diversas herramientas diseñadas para facilitar la ejecución eficiente de modelos de *deep learning* en dispositivos móviles y embebidos [67, 71]:

- **TensorFlow Lite (TFLite):** es un intérprete ligero desarrollado por Google para ejecutar modelos en dispositivos Android, iOS y sistemas embebidos. Soporta cuantización de 8 y 16 bits, y puede delegar operaciones a aceleradores de hardware como la Android Neural Networks API (NNAPI) [71].

En este trabajo, métricas como FLOPs, latencia, FPS y consumo de VRAM se utilizan para evaluar la viabilidad de los modelos en escenarios de inferencia móvil y hardware con recursos limitados.

Capítulo 3: Metodología

3.1. Diseño general del estudio

El presente trabajo está orientado a la implementación, experimentación y evaluación de modelos de aprendizaje profundo para la segmentación y clasificación automática de quemaduras en la piel de segundo y tercer grado en imágenes clínicas. La investigación se enmarca dentro de un enfoque aplicado, experimental y comparativo.

La naturaleza aplicada del estudio se justifica por el desarrollo de una solución tecnológica funcional, con potencial de utilizarse como herramienta de apoyo en entornos clínicos, mediante la integración de modelos de inteligencia artificial en una aplicación móvil.

El enfoque experimental se adopta debido a que el desempeño de las arquitecturas evaluadas se analiza bajo condiciones controladas de entrenamiento, validación y prueba. Para ello, se realizaron experimentos de optimización del modelo YOLO11-seg, un *benchmark* comparativo *multi-seed* entre YOLO11m-seg y arquitecturas CNN basadas en U-Net, Attention U-Net y U-Net++, así como una evaluación de eficiencia computacional.

El enfoque comparativo permitió analizar el comportamiento de cada arquitectura bajo el mismo protocolo experimental, considerando métricas de segmentación, análisis estadístico y eficiencia computacional. De esta manera, se proporciona evidencia empírica para evaluar la viabilidad del modelo seleccionado en aplicaciones móviles con recursos computacionales limitados orientadas al apoyo en la evaluación clínica de quemaduras.

3.2. Descripción del conjunto de datos

3.2.1. Recolección de los datos

Se utilizó un conjunto de datos de imágenes médicas proporcionado por la Unidad de Quemados del Hospital General de Chihuahua en dos etapas, el cual contiene fotografías clínicas



de lesiones por quemaduras. Las imágenes fueron capturadas en el mismo hospital a pacientes con quemaduras mediante dispositivos móviles bajo condiciones reales de iluminación y adquisición clínica, reflejando escenarios representativos del entorno hospitalario.

Primera etapa: El Hospital General de Chihuahua, proporcionó una base de datos previamente anotada por especialistas médicos en quemaduras por medio de la plataforma de Roboflow. El conjunto descargado contenía 321 imágenes clínicas en formato .jpg junto con sus respectivas anotaciones en formato YOLO (.txt), con estructura:

$$\text{clase } u_1 \ v_1 \ u_2 \ v_2 \ \dots \ u_n \ v_n$$

donde (u_i, v_i) representan las coordenadas normalizadas de los vértices del polígono de segmentación asociados a cada instancia anotada.

Segunda etapa: Posteriormente el mismo Hospital General de Chihuahua, proporcionó otro conjunto de 432 imágenes en formato .jpg sin anotaciones, las cuales fueron delimitadas manualmente en las regiones de interés por medio de polígonos cerrados en la plataforma de LabelMe, como se muestra en la Ilustración 3.1, siguiendo los criterios clínicos establecidos por especialistas. El proceso de anotación de las regiones de interés mediante polígonos fue realizado manualmente por el autor de esta tesis.

Las anotaciones generadas se almacenaron en archivos en formato YOLO para cada imagen, los cuales contienen la información geométrica de los polígonos y la clase asociada a cada región anotada. Estas máscaras son utilizadas como etiquetas de referencia durante los procesos de entrenamiento, validación y prueba de los modelos de segmentación. Las máscaras correspondientes al conjunto de prueba se emplean como etiquetas reales para la comparación directa con las predicciones generadas por los modelos, siendo su calidad determinante en el cálculo de las métricas de segmentación.

Posteriormente, el conjunto de datos fue revisado y validado por un especialista en el tratamiento de quemaduras en la piel, con el objetivo de asegurar la correcta correspondencia entre las regiones anotadas y las lesiones observadas en la imagen original en cuanto a clase y área.

Verificación de calidad de los datos Se realizó una inspección visual de las imágenes en cada una de las etapas y se eliminaron las que contenían sustancias como solución jabonosa, ungüento,



Ilustración 3.1: Ejemplo del proceso de anotación manual de una imagen clínica de quemadura mediante polígonos en la herramienta LabelMe.

obstrucción visual con algún objeto sobre la quemadura o que no pertenecían a una representación de herida por quemadura de una de las clases correspondientes. Más adelante se desarrolló un procedimiento de validación visual mediante la superposición de la máscara en formato PNG sobre la imagen original de la herida para verificar la correcta alineación entre el polígono y la región de quemadura.

Se corrigieron las anotaciones de 30 imágenes correspondientes al conjunto de datos de la etapa 1. El problema identificado consistía en que una misma región de la imagen estaba asociada simultáneamente a dos etiquetas diferentes. Esta situación ocurría principalmente en casos donde una región de un determinado grado de quemadura se encontraba contenida dentro de otra región etiquetada con un grado distinto.

Este error se originó durante el proceso de anotación, ya que la herramienta LabelMe permite la superposición de regiones cuando los polígonos no son delimitados adecuadamente. Por ello, se revisaron y corrigieron manualmente las anotaciones afectadas para garantizar que cada píxel estuviera asociado a una única clase. Una vez corregidas, las imágenes y sus respectivas anotaciones fueron reincorporadas al conjunto de datos correspondiente.



En la Tabla 3.1 se muestran el número de imágenes resultante después de la inspección, las instancias por grado y la resolución en cada etapa.

Tabla 3.1: Distribución de datos por etapas del dataset de quemaduras.

Etapas	Imágenes	Instancias grado 2	Instancias grado 3	Resolución
Etapas 1	309	309	182	640×640
Etapas 2	403	519	459	variable

3.2.2. Estructura y características del dataset

Las dos etapas del conjunto de datos hospitalarios están compuestas por imágenes clínicas de quemaduras clasificadas en tres clases, mostradas en la Tabla 3.2.

Tabla 3.2: Clases del dataset de quemaduras.

Clase	Valor	Descripción
0	Fondo	Área sin lesión
1	Segundo grado	Quemadura de segundo grado
2	Tercer grado	Quemadura de tercer grado

3.3. Preprocesamiento de imágenes

El preprocesamiento de las imágenes constituye una etapa fundamental en el desarrollo del sistema, ya que permite transformar los datos en un formato estándar estructurado, para que sea consistente y adecuado para el entrenamiento e inferencia del modelo de segmentación. El objetivo es garantizar la adecuada uniformidad y compatibilidad con los recursos computacionales, así como la consistencia en el proceso del modelo de aprendizaje profundo.

Redimensionamiento Las imágenes correspondientes a la segunda etapa del conjunto de datos hospitalario presentaban resoluciones variables, con dimensiones que oscilaban aproximadamente



entre 561×561 y 4160×4160 píxeles. Por otra parte, las imágenes provenientes de la primera etapa que fue descargada desde Roboflow ya se encontraban previamente exportadas a una resolución fija de 640×640 píxeles.

Con el objetivo de mantener uniformidad durante el entrenamiento e inferencia, todas las imágenes utilizadas en el estudio fueron procesadas finalmente a una resolución de 640×640 píxeles, correspondiente al tamaño de entrada utilizado por YOLO11m-seg.

El proceso de redimensionamiento se realizó mediante interpolación bilineal para las imágenes RGB y utilizando interpolación nearest-neighbor para las máscaras de segmentación, preservando la correspondencia espacial de las anotaciones. Esta resolución permitió mantener un balance adecuado entre precisión de segmentación, estabilidad del entrenamiento y costo computacional.

Normalización Como parte del preprocesamiento utilizado durante el entrenamiento y evaluación de las arquitecturas implementadas, los valores de intensidad de los píxeles, originalmente codificados en RGB dentro del rango de 0 a 255, fueron escalados al rango de 0 a 1:

$$\text{valor normalizado} = \frac{\text{valor del píxel}}{255} \quad (3.1)$$

En el caso de YOLO11-seg, esta normalización corresponde al esquema de preprocesamiento estándar implementado automáticamente por Ultralytics YOLO durante el entrenamiento e inferencia. Para las arquitecturas basadas en U-Net se utilizó el mismo esquema de escalamiento durante la carga y transformación de imágenes.

No se aplicó una normalización basada en medias y desviaciones estándar de ImageNet, con el objetivo de mantener consistencia entre entrenamiento, evaluación y despliegue del modelo en dispositivos móviles.

3.3.1. División del conjunto de datos

El conjunto total de datos utilizado en este trabajo quedó compuesto por 712 imágenes clínicas correspondientes a quemaduras de segundo y tercer grado. La integración de ambas etapas permitió construir un conjunto de datos más amplio y representativo del problema clínico.

Debido a que el conjunto de la primera etapa incluía aumentaciones *offline* generadas antes de la exportación y no se disponía de trazabilidad completa entre imágenes originales y sus variantes



aumentadas, se utilizó exclusivamente durante el entrenamiento del modelo. En el conjunto de la segunda etapa se realizó una partición reproducible en subconjuntos de entrenamiento, validación y prueba.

La estrategia adoptada permitió evitar que posibles variantes derivadas de una misma imagen estuvieran presentes simultáneamente en entrenamiento y evaluación, reduciendo el riesgo de *data leakage* y proporcionando una evaluación más controlada sobre imágenes clínicas originales.

La división final utilizada en el estudio fue de aproximadamente 77 % para entrenamiento, 10 % para validación y 14 % para prueba.

La Tabla 3.3 presenta la distribución final de imágenes empleadas en cada subconjunto.

Tabla 3.3: Distribución final de imágenes utilizadas en cada subconjunto del dataset.

Subconjunto	Dataset 1	Dataset 2	Total
Entrenamiento (train)	309	307	616
Validación (val)	0	40	40
Prueba (test)	0	56	56
Total	309	403	712

Distribución de clases del conjunto de datos Se realizó un análisis exploratorio de la distribución de clases presentes en el conjunto de datos utilizado para el entrenamiento del modelo. Considerando ambas etapas del dataset empleadas durante el entrenamiento, se observó una distribución relativamente equilibrada entre las clases principales, con aproximadamente 56 % y 43 % para quemaduras de segundo y tercer grado, respectivamente. Esta distribución resulta favorable para el entrenamiento de modelos de aprendizaje profundo, ya que reduce el sesgo asociado a desbalances extremos entre clases.

En la Figura 3.1 se muestra una representación visual de la distribución de instancias presentes en el conjunto de datos.

La Figura 3.2 muestra la distribución del número de objetos anotados por imagen en el conjunto total de datos. Se observa una mayor frecuencia de imágenes que contienen entre una y tres lesiones segmentadas, mientras que un número reducido de imágenes presenta múltiples regiones

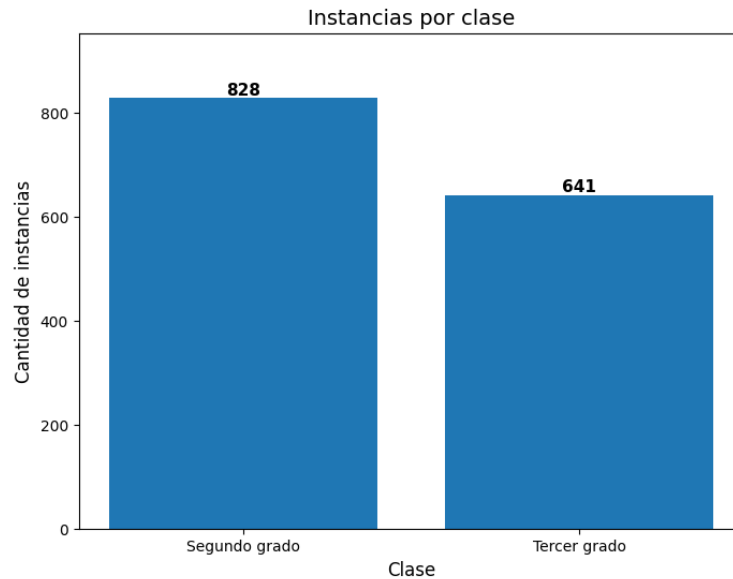


Figura 3.1: Distribución de clases en el conjunto de datos clínico.

de quemadura. Esta distribución refleja la variabilidad clínica presente en el dataset y permite caracterizar la complejidad estructural de las anotaciones utilizadas durante el entrenamiento del modelo.

Aumento de datos Se emplearon técnicas de aumento de datos con el objetivo de mejorar la capacidad de generalización de los modelos y reducir el riesgo de sobreajuste. Estas técnicas consisten en generar variaciones de las imágenes mediante transformaciones geométricas y fotométricas, preservando la información clínica relevante de las lesiones.

En la primera etapa del conjunto de datos, las imágenes descargadas desde Roboflow ya incluían versiones aumentadas generadas automáticamente por la plataforma. De acuerdo con la configuración reportada en el archivo *README* del conjunto, las transformaciones aplicadas incluyeron volteos horizontales y verticales, rotaciones, variaciones de brillo y exposición, desenfoque gaussiano y ruido tipo *salt-and-pepper*. Estas aumentaciones formaban parte del conjunto original recibido y no fueron generadas manualmente durante este trabajo.

Debido a que estas imágenes aumentadas podían corresponder a variantes derivadas de una misma imagen fuente, la partición final fue reconstruida para evitar solapamiento entre entrenamiento, validación y prueba. En consecuencia, las imágenes provenientes del conjunto aumentado de Roboflow (primera etapa) fueron utilizadas únicamente dentro del subconjunto de entrenamiento.

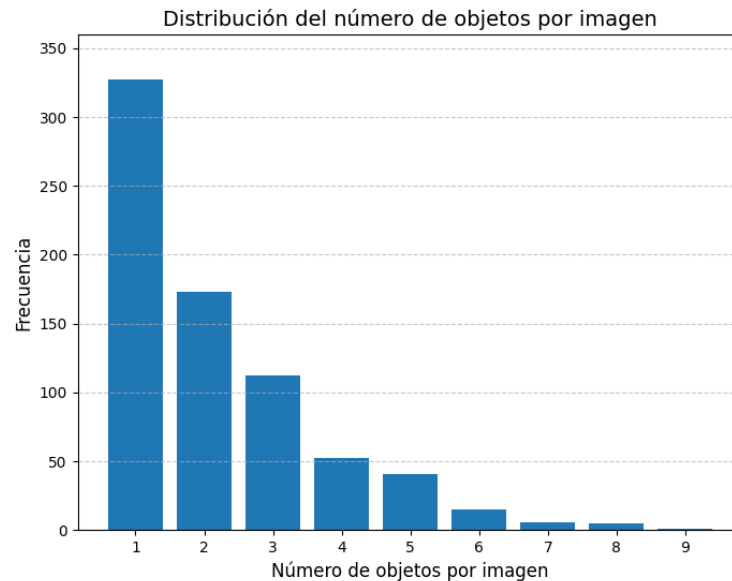


Figura 3.2: Distribución del número de objetos anotados por imagen en el conjunto de datos clínico.

Para la segunda etapa del conjunto de datos no se generaron nuevas imágenes aumentadas de forma física ni se almacenaron copias adicionales de las imágenes originales. Esta decisión permitió mantener trazabilidad directa entre cada imagen clínica y su anotación correspondiente.

En lugar de generar aumentaciones *offline* permanentes, las transformaciones fueron aplicadas dinámicamente durante el entrenamiento mediante *pipelines* de aumento de datos *online*. Estas transformaciones se ejecutaron de manera probabilística en cada iteración de entrenamiento, generando variaciones distintas de una misma imagen únicamente durante el proceso de aprendizaje, sin modificar permanentemente el conjunto de datos original.

Las transformaciones *online* fueron restringidas exclusivamente al subconjunto de entrenamiento. Los subconjuntos de validación y prueba se mantuvieron sin aumentaciones adicionales.

Las transformaciones utilizadas incluyeron rotaciones aleatorias leves, volteo horizontal, variaciones moderadas de brillo y escalado leve de la imagen.

Consideraciones éticas Las imágenes utilizadas están anonimizadas y no contienen ninguna información explícita o metadatos de los pacientes. El manejo de los datos se realizó conforme a principios éticos de confidencialidad y uso responsable de información clínica, garantizando que las imágenes se emplearan exclusivamente con fines de investigación académica.



3.4. Modelo supervisado YOLOv11-seg

Arquitectura general En este trabajo se empleó la arquitectura YOLO11-seg como modelo base para la segmentación y clasificación automática de quemaduras en imágenes clínicas. Inicialmente, se realizaron diferentes experimentaciones con el objetivo de identificar una configuración adecuada de hiperparámetros, incluyendo resolución de entrada, número de épocas y tamaño de *batch*.

Posteriormente, la configuración seleccionada fue utilizada como referencia experimental para realizar el *benchmark* comparativo *multi-seed* frente a las arquitecturas U-Net, Attention U-Net y U-Net++ bajo condiciones homogéneas de entrenamiento y evaluación.

Formulación del problema de segmentación. El problema abordado en este trabajo consiste en la segmentación de lesiones por quemaduras en imágenes clínicas, formulado como una tarea de segmentación de instancias para dos clases clínicas.

El modelo recibe como entrada una imagen clínica y produce simultáneamente:

- Cajas delimitadoras
- Etiquetas de clase
- Máscaras de segmentación

Las clases consideradas en el modelo corresponden a:

- Clase 0: quemadura de segundo grado
- Clase 1: quemadura de tercer grado

Configuración de entrenamiento El modelo YOLO11-seg fue entrenado bajo un esquema de aprendizaje supervisado utilizando imágenes anotadas con máscaras de segmentación de instancias. Durante el entrenamiento, la arquitectura optimiza de manera conjunta las tareas de detección, clasificación y segmentación mediante funciones de pérdida implementadas internamente en el framework Ultralytics YOLO.

Estas funciones incluyen términos asociados a la clasificación de objetos, localización de cajas delimitadoras y segmentación de máscaras, incorporando mecanismos como Distribution Focal Loss (DFL) para mejorar la precisión en la localización espacial de las predicciones [55].



Diseño experimental Se realizaron dos experimentos principales con el objetivo de analizar el impacto de diferentes parámetros de entrenamiento sobre el desempeño del modelo YOLOv11-seg en la tarea de segmentación de quemaduras.

Las configuraciones fueron ajustadas de manera progresiva mediante un enfoque iterativo de experimentación, donde cada nueva configuración fue diseñada a partir del comportamiento observado en los experimentos previos. El objetivo fue encontrar una configuración adecuada para el conjunto de datos, buscando mejorar la capacidad de generalización del modelo sin inducir a sobreajuste.

Como configuración base se utilizó el modelo YOLO11m-seg entrenado con imágenes de entrada de 640×640 píxeles, 200 épocas de entrenamiento, tamaño de *batch* de 16, *patience* de 50 épocas y *workers*=8 para la carga de datos.

Experimento 1

El experimento 1 consistió en evaluar tres configuraciones de entrenamiento derivadas de una configuración inicial de referencia, modificando progresivamente hiperparámetros relacionados con resolución de entrada, número de épocas y variante de arquitectura. Las configuraciones evaluadas se muestran en la Tabla 3.4.

Los resultados obtenidos durante esta fase experimental mostraron comportamientos distintos dependiendo de la configuración evaluada. El incremento de resolución permitió mantener un desempeño competitivo y favoreció cualitativamente algunas regiones pequeñas y bordes complejos; no obstante, no produjo mejoras globales consistentes respecto a la configuración base.

Por otra parte, el incremento a 300 épocas no generó mejoras relevantes en las métricas globales bajo el protocolo experimental utilizado. En contraste, la variante YOLO11s-seg obtuvo el mayor valor de Dice global durante esta etapa exploratoria; en este contexto, la configuración YOLO11m-seg mantuvo un mejor equilibrio general entre IoU, recall, estabilidad y capacidad de generalización.

A partir de estos resultados, las configuraciones basadas en YOLO11m-seg fueron utilizadas como referencia para los experimentos posteriores y para el *benchmark* comparativo *multi-seed*.

Experimento 2

El experimento 2 se enfocó en evaluar configuraciones adicionales utilizando un máximo de 300 épocas de entrenamiento, modificando parámetros relacionados con resolución de entrada,



Tabla 3.4: Configuraciones evaluadas durante el experimento 1 para la selección inicial de hiperparámetros.

Configuración	Modelo	Resolución	Épocas	Objetivo experimental
Configuración base	YOLO11m	640	200	Configuración inicial utilizada como punto de partida experimental.
Configuración 1	YOLO11m	768	200	Evaluar el efecto de una mayor resolución sobre la segmentación de bordes complejos y regiones pequeñas.
Configuración 2	YOLO11m	640	300	Analizar si un mayor número de épocas favorece la convergencia y la generalización del modelo.
Configuración 3	YOLO11s	640	200	Evaluar una variante más ligera orientada a escenarios con restricciones computacionales.

estrategias de augmentación y tamaño de *batch*. Las configuraciones evaluadas se muestran en la Tabla 3.5. Los parámetros no mostrados mantuvieron los valores de la configuración base utilizada en los experimentos previos. Una de las configuraciones tuvo como objetivo evaluar el efecto de augmentaciones *online* conservadoras aplicadas durante el entrenamiento sobre las imágenes clínicas correspondientes a la segunda etapa del conjunto de datos.

Los resultados obtenidos mostraron que la reducción del tamaño de *batch* favoreció la capacidad de generalización del modelo bajo el protocolo experimental utilizado. En particular, la configuración con *batch size* de 8 presentó el mejor equilibrio entre Dice, IoU y recall respecto a las demás configuraciones evaluadas. Por esta razón, dicha configuración fue seleccionada para el *benchmark multi-seed*.



Tabla 3.5: Configuraciones evaluadas durante el experimento 2 utilizando 300 épocas de entrenamiento.

Configuración	Modelo	Resolución	Épocas	Batch	Objetivo experimental
Configuración 1	YOLO11m	768	300	16	Combinar alta resolución y mayor número de épocas para evaluar mejoras simultáneas en segmentación global y regiones pequeñas.
Configuración 2	YOLO11m	640	300	16	Evaluar el efecto de augmentación clínica conservadora durante el entrenamiento.
Configuración 3	YOLO11m	640	300	8	Analizar el impacto de un <i>batch size</i> menor sobre la estabilidad y capacidad de generalización del modelo.

Por otra parte, la combinación de alta resolución y 300 épocas incrementó los requerimientos computacionales asociados al entrenamiento sin producir mejoras consistentes en las métricas globales. Asimismo, la estrategia de augmentación clínica conservadora mantuvo un desempeño competitivo y mostró ligeras mejoras en generalización; no obstante, no superó a la configuración con *batch size* reducido.

A partir de estos resultados, la configuración final seleccionada para el *benchmark multi-seed* correspondió al modelo YOLO11m-seg con resolución de 640×640 píxeles, *batch size* de 8 y un máximo de 300 épocas de entrenamiento.



3.5. Benchmark multi-seed y análisis estadístico

Con el objetivo de realizar una comparación experimental entre el modelo YOLO11m-seg seleccionado y arquitecturas clásicas de segmentación médica, se implementó un *benchmark* supervisado utilizando variantes de U-Net. El propósito de esta comparación fue analizar el desempeño de un modelo de segmentación de instancias frente a arquitecturas especializadas en segmentación semántica para la tarea de segmentación de quemaduras.

3.5.1. Arquitecturas evaluadas

Las arquitecturas incluidas en el *benchmark multi-seed* fueron:

- YOLO11m-seg
- U-Net
- U-Net++
- Attention U-Net

3.5.2. Preparación del conjunto de datos para el *benchmark*

Las máscaras utilizadas por las arquitecturas U-Nets fueron generadas mediante la rasterización de los polígonos contenidos en las etiquetas YOLO de segmentación. Este procedimiento permitió utilizar exactamente el mismo conjunto de datos para todos los modelos evaluados.

3.5.3. Configuración experimental

Todas las arquitecturas fueron entrenadas con imágenes de entrada de 640×640 píxeles, *batch size* de 8 y un máximo de 300 épocas. En el caso de las arquitecturas U-Net, Attention U-Net y U-Net++, se utilizó ResNet34 como encoder preentrenado en ImageNet. El entrenamiento de las CNN fue realizado mediante PyTorch y la librería *segmentation_models_pytorch*, mientras que YOLO11m-seg fue entrenado mediante Ultralytics YOLO.



En todos los casos se seleccionó el mejor *checkpoint* de acuerdo con el desempeño observado en el conjunto de validación.

3.5.4. Estrategias de augmentación durante el *benchmark*

Durante el *benchmark multi-seed* se mantuvo el esquema de augmentación *online* descrito en la etapa de preprocesamiento. Estas transformaciones fueron aplicadas únicamente durante el entrenamiento y exclusivamente sobre el subconjunto de entrenamiento. Los subconjuntos de validación y prueba se mantuvieron sin augmentaciones adicionales, con el fin de garantizar una evaluación consistente sobre imágenes clínicas originales.

3.5.5. Funciones de pérdida

El modelo YOLO11m-seg utiliza las funciones de pérdida implementadas internamente por Ultralytics YOLO, orientadas a optimizar de manera conjunta detección, clasificación y segmentación de instancias.

Por otra parte, las arquitecturas U-Net, Attention U-Net y U-Net++ fueron entrenadas mediante una función de pérdida compuesta basada en CrossEntropy Loss y Dice Loss, con el objetivo de optimizar simultáneamente la clasificación píxel a píxel y el solapamiento espacial entre las máscaras predichas y las máscaras de referencia.

3.5.6. Protocolo multi-seed

Cada arquitectura fue entrenada mediante 10 corridas independientes utilizando diferentes semillas aleatorias, con el objetivo de evaluar la estabilidad y variabilidad estadística del entrenamiento. Las semillas utilizadas fueron:

- 3401
- 3402
- 3403
- 3404



- 3405
- 3406
- 3407
- 3408
- 3409
- 3410

Durante todas las corridas se mantuvieron constantes:

- Conjunto de datos
- Particiones *train/validation/test*
- Resolución de entrada
- Funciones de pérdida
- Hiperparámetros
- Protocolo de evaluación
- Criterio de selección del mejor *checkpoint*

3.5.7. Cálculo de métricas de segmentación

Para evaluar el desempeño de los modelos se utilizaron las métricas Dice, IoU, precisión y recall. Estas métricas fueron calculadas a partir de los valores de verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN), obtenidos al comparar las máscaras predichas contra las máscaras de referencia del conjunto de prueba.

En el caso de YOLO11m-seg, el modelo genera máscaras de instancia asociadas a una clase y a un nivel de confianza. Para hacer comparable su salida con las arquitecturas de segmentación semántica, las máscaras predichas fueron proyectadas sobre una máscara semántica en el mismo espacio de la imagen original. De esta manera, cada píxel quedó asignado a una de tres categorías:



fondo, quemadura de segundo grado o quemadura de tercer grado. Cuando existieron regiones traslapadas entre múltiples instancias, se conservó la clase correspondiente a la predicción con mayor nivel de confianza.

Para las arquitecturas U-Net, Attention U-Net y U-Net++, la salida del modelo corresponde directamente a un mapa de segmentación semántica multiclase.

Las métricas globales fueron calculadas mediante una evaluación binaria de región lesionada frente a fondo. Para ello, las clases de quemadura de segundo y tercer grado fueron fusionadas en una sola región de *foreground*, mientras que el resto de los píxeles fueron considerados como fondo. Bajo esta definición, las métricas globales evalúan la capacidad general del modelo para localizar regiones lesionadas, independientemente del grado clínico asignado a cada píxel.

El coeficiente Dice, IoU, precisión y recall globales se calcularon de forma independiente para cada imagen del conjunto de prueba y posteriormente se reportaron como el promedio aritmético sobre todas las imágenes evaluadas. De esta manera, cada imagen tuvo el mismo peso dentro del resultado final, evitando que imágenes con regiones lesionadas más extensas dominaran completamente el promedio global.

Adicionalmente, se calcularon métricas por clase para las categorías de quemadura de segundo y tercer grado, excluyendo la clase *background*. En este análisis, cada clase fue evaluada mediante un esquema uno-contra-resto, donde los píxeles pertenecientes a la clase evaluada fueron considerados positivos y el resto de los píxeles negativos. Las métricas por clase fueron calculadas acumulando los valores de TP, FP y FN correspondientes a cada categoría sobre todo el conjunto de prueba.

Por tanto, las métricas globales y las métricas por clase representan evaluaciones complementarias. Las métricas globales cuantifican la capacidad general del modelo para localizar regiones lesionadas frente al fondo, mientras que las métricas por clase permiten evaluar la correcta asignación semántica entre quemaduras de segundo y tercer grado.

3.5.8. Matrices de confusión entre clases

Además de las métricas de segmentación, se construyeron matrices de confusión normalizadas para analizar los errores de asignación semántica entre grados de quemadura. Para este análisis se consideraron únicamente los píxeles cuya clase de referencia correspondía a quemadura



de segundo o tercer grado.

A diferencia de una matriz 2×2 entre clases clínicas, se utilizó una representación 2×3 . Las filas corresponden a la clase de referencia, quemadura de segundo grado y quemadura de tercer grado, mientras que las columnas corresponden a la clase predicha: fondo, quemadura de segundo grado y quemadura de tercer grado. Esta representación permite distinguir entre dos tipos de error: la omisión de píxeles lesionados hacia fondo y la confusión semántica entre grados de quemadura.

Las matrices fueron normalizadas por filas, por lo que cada fila representa la distribución porcentual de los píxeles de una clase de referencia entre las clases predichas por el modelo.

3.5.9. Análisis estadístico

Las métricas obtenidas en las 10 corridas independientes de cada arquitectura fueron analizadas mediante pruebas estadísticas no paramétricas. Inicialmente se aplicó la prueba de Kruskal-Wallis para detectar diferencias globales entre modelos. Posteriormente, se realizó un análisis post-hoc de Dunn con corrección de Bonferroni para identificar diferencias estadísticamente significativas entre pares de arquitecturas. Además, se calcularon medidas de tendencia central y dispersión, incluyendo media y desviación estándar, con el objetivo de evaluar tanto el rendimiento promedio como la estabilidad de cada modelo.

3.5.10. Benchmark de eficiencia computacional

Además de las métricas de segmentación, se realizó una evaluación de eficiencia computacional para analizar el costo de procesamiento de cada arquitectura.

Las métricas evaluadas incluyeron:

- Número de parámetros
- FLOPs
- Latencia de inferencia
- FPS
- Consumo máximo de memoria GPU



Para el análisis de latencia se utilizó el *checkpoint* seleccionado de cada modelo bajo el mismo criterio de validación empleado en la evaluación cuantitativa. Dado que métricas como número de parámetros, FLOPs, consumo de memoria y latencia dependen principalmente de la arquitectura, el tamaño de entrada y el *pipeline* de inferencia, estas mediciones no fueron recalculadas para cada semilla del análisis *multi-seed*.

3.6. Recursos computacionales

Los experimentos de entrenamiento y evaluación fueron ejecutados en una estación de trabajo con aceleración por GPU. La configuración del entorno computacional utilizado se presenta en la Tabla 3.6.

El entrenamiento de los modelos y las evaluaciones experimentales se realizaron utilizando el *framework* PyTorch 2.10 con soporte CUDA y la implementación Ultralytics YOLO versión 8.4.21.

Tabla 3.6: Entorno computacional utilizado para entrenamiento y evaluación experimental.

Componente	Especificación
CPU	Intel Core Ultra 7 265KF
Núcleos	20
GPU	NVIDIA GeForce RTX 5080
VRAM	16 GB
RAM	62 GB
Sistema operativo	Ubuntu 24.04.3 LTS
CUDA Toolkit	12.6
PyTorch CUDA	cu128
Python	3.11.15
Framework	PyTorch 2.10.0
Implementación YOLO	Ultralytics 8.4.21
Tipo de cómputo	Aceleración por GPU



3.7. Implementación en aplicación móvil

En esta sección se describe la última parte de la metodología, el modelo propuesto de inteligencia artificial (YOLOv11-seg) se integra en una aplicación móvil funcional. El enfoque desarrollado puede aplicarse en entornos clínicos reales como herramienta de apoyo visual en la evaluación inicial de lesiones por quemaduras. La estrategia propuesta permite analizar el desempeño del modelo en dispositivos móviles, considerando las restricciones propias de estos sistemas, como la capacidad de memoria, procesamiento y latencia durante la inferencia.

Objetivo de la implementación. El objetivo de la implementación fue el desarrollo de una aplicación móvil automatizada, capaz de generar resultados confiables y visualmente interpretables para usuarios clínicos en dispositivos móviles comunes. El despliegue de la aplicación permite evaluar técnicamente el sistema para su uso en entornos reales, analizando el tiempo de respuesta y la capacidad de procesamiento en dispositivos móviles.

Usuarios objetivo. El sistema puede ser utilizado por diferentes usuarios clínicos con el objetivo de apoyar la determinación preliminar de la gravedad de una quemadura en la piel, incluyendo:

- Médicos generales
- Personal de enfermería
- Personal de atención de urgencias
- Personal clínico en formación

Justificación del uso de la aplicación móvil. El despliegue de una aplicación móvil para clasificar y segmentar heridas por quemadura en imágenes se justifica por la necesidad de contar con herramientas clínicas portátiles, accesibles y de bajo costo. El método propuesto se implementó como un sistema *offline*. Esta característica permite:

- Reducir la dependencia de infraestructura computacional externa
- Facilitar la evaluación en tiempo real



- Permitir el uso del sistema en entornos clínicos con recursos limitados
- Evaluar el desempeño del modelo bajo condiciones operativas reales

Tipo de sistema desarrollado. El sistema proporciona información generada por el modelo de inteligencia artificial, cuyo desempeño fue previamente validado mediante métricas de segmentación. Además, integra un sistema basado en reglas clínicas para la generación de recomendaciones preliminares. La información desplegada en la aplicación incluye:

- Segmentación de áreas afectadas por quemaduras
- Clasificación del grado de severidad de la lesión
- Estimación del área corporal afectada
- Generación de recomendaciones clínicas preliminares

Tecnologías y plataforma de desarrollo. El modelo seleccionado para el despliegue móvil fue YOLO11m-seg, previamente entrenado y evaluado en un entorno de escritorio Linux. Posteriormente, el modelo fue convertido a TensorFlow Lite para permitir su ejecución local en dispositivos Android.

El sistema fue desarrollado utilizando el entorno de desarrollo Android Studio, el cual permite la creación de aplicaciones para dispositivos con sistema operativo Android. Este entorno proporciona una amplia disponibilidad de bibliotecas y herramientas para la integración de modelos de inteligencia artificial. El lenguaje de programación utilizado para el desarrollo fue Kotlin. El framework TensorFlow Lite se utilizó para ejecutar el modelo en el dispositivo móvil, permitiendo la optimización del modelo y la reducción del tiempo de inferencia durante su operación.

3.7.1. Enfoque metodológico

El desarrollo del sistema se llevó a cabo mediante un enfoque incremental [72], lo que permite que se desarrolle mediante fases. Esta metodología permite validar cada paso de forma independiente e ir incrementando cada componente del sistema. El método facilita la reducción de la complejidad, además de identificar errores de manera más fluida.



El sistema se dividió en fases llamadas *sprints*, cada una de estas fases está dividida en objetivos funcionales de la aplicación y tiene un objetivo funcional completo en el sistema. Cada *sprint* contempla historias de usuario específicas que a su vez contienen criterios de aceptación que validan que se ha completado satisfactoriamente. Estas subdivisiones permiten reducir la complejidad del sistema en pasos concretos y bien definidos. Cada *sprint* puede ser resumido de la siguiente manera:

1. Adquisición de imágenes
2. Validación del flujo de interacción
3. Integración del modelo de inteligencia artificial
4. Visualización y almacenamiento de resultados
5. Evaluación del rendimiento computacional

La planificación de las fases de desarrollo del sistema se presenta en la Tabla C.1, ubicada en el Anexo Anexo C. El detalle completo de las historias de usuario, incluyendo su descripción y criterios de aceptación, se presenta en la Tabla C.2, ubicada en el Anexo Anexo C.

Se realizaron pruebas funcionales y de rendimiento computacional en diferentes dispositivos para verificar el sistema. Las pruebas de latencia se implementaron con la toma de tiempos en secciones específicas del *pipeline* (preprocesamiento, inferencia, postprocesamiento) con el fin de medir cada fase con precisión, así como el *pipeline* completo.

3.7.2. Arquitectura general del sistema

La arquitectura general de la aplicación móvil está organizada en módulos; adquisición de imágenes, procesamiento e inferencia con el modelo YOLOv11-seg, resultados clínicos y el almacenamiento de información. Este sistema permite el control y la gestión de la información procesada para el análisis automatizado de imágenes clínicas de quemaduras mediante inteligencia artificial.

La arquitectura del sistema se diseñó de forma modular y en capas. Esta organización permite separar la interfaz de usuario, el proceso de inferencia, la lógica clínica y el almacenamiento de



datos. De esta manera, cada componente puede modificarse o depurarse de forma independiente, facilitando el mantenimiento y la escalabilidad del sistema.

El sistema está implementado como una aplicación móvil Android y se compone de los siguientes módulos principales:

Módulo 1 — Interfaz de usuario UI: Permite la interacción del usuario con el sistema, carga de imágenes y la visualización de resultados.

Módulo 2 — Inferencia: Es el encargado del preprocesamiento, inferencia con el modelo YOLOv11-seg y postprocesamiento necesario para generar áreas de segmentación y objetos detectados en las imágenes clínicas.

Módulo 3 — Sistema de lógica clínica: Este módulo interpreta los resultados generados y los datos proporcionados por el usuario para generar recomendaciones clínicas orientativas.

Módulo 4 — Almacenamiento de datos: Se encarga de almacenar datos a lo largo del sistema como imágenes, resultados de inferencia, datos del paciente y recomendaciones orientativas para su posterior gestión.

Para una descripción estructural detallada de los módulos del sistema, se presenta la arquitectura en capas del sistema en el Anexo Anexo D.

El flujo general del sistema inicia con la adquisición de una imagen clínica, posteriormente, la imagen es procesada por el módulo de inferencia, donde se ejecuta el modelo de inteligencia artificial YOLOv11-seg para detectar y segmentar las lesiones. Los resultados generados son interpretados por el sistema de reglas clínicas, el cual generará recomendaciones orientativas. Finalmente, la información es visualizada en la interfaz de usuario y almacenada en la base de datos del sistema.

La Figura 3.3 muestra la arquitectura general de la aplicación móvil, que describe los módulos principales de la aplicación móvil y el flujo de procesamiento de la información.

3.7.3. Caso de uso del sistema

El sistema puede ser representado mediante la descripción de casos de uso, que es un conjunto de acciones del usuario y del sistema para cada una de las fases descritas anteriormente.

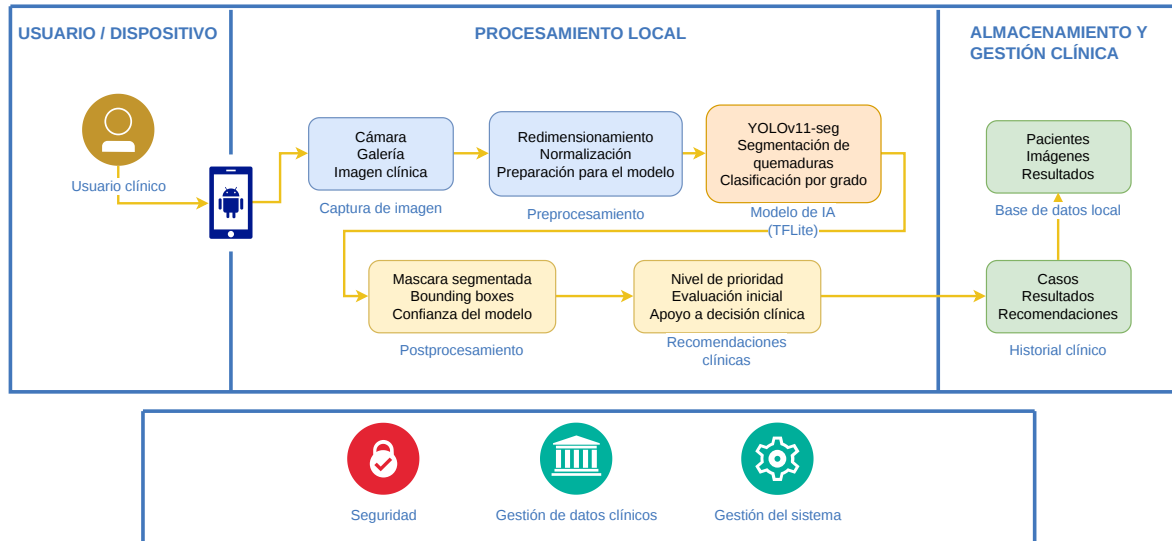


Figura 3.3: Arquitectura general de la aplicación móvil. Elaboración propia

La interacción entre el usuario y el sistema permite la carga de una imagen, inferencia y obtener resultados para el apoyo en la evaluación visual de quemaduras. El proceso general del sistema se desarrolla mediante las siguientes acciones:

1. El usuario inicia la aplicación móvil
2. El usuario almacena datos de un nuevo paciente
3. El usuario captura una imagen clínica o selecciona una imagen de la galería
4. El sistema realiza el preprocesamiento de la imagen
5. El sistema ejecuta el modelo de inteligencia artificial para la segmentación y clasificación de la quemadura
6. El sistema genera resultados visuales sobre la imagen procesada
7. El sistema calcula parámetros clínicos relevantes, como el porcentaje estimado de superficie corporal afectada
8. El sistema genera recomendaciones clínicas orientativas



9. El sistema permite almacenar los resultados obtenidos

El comportamiento del sistema se representa mediante un diagrama de casos de uso; este diagrama describe la interacción entre el usuario y el sistema. El diagrama permite ver las funcionalidades del sistema y las acciones del usuario con el sistema. El diagrama correspondiente se presenta en la figura 3.4.

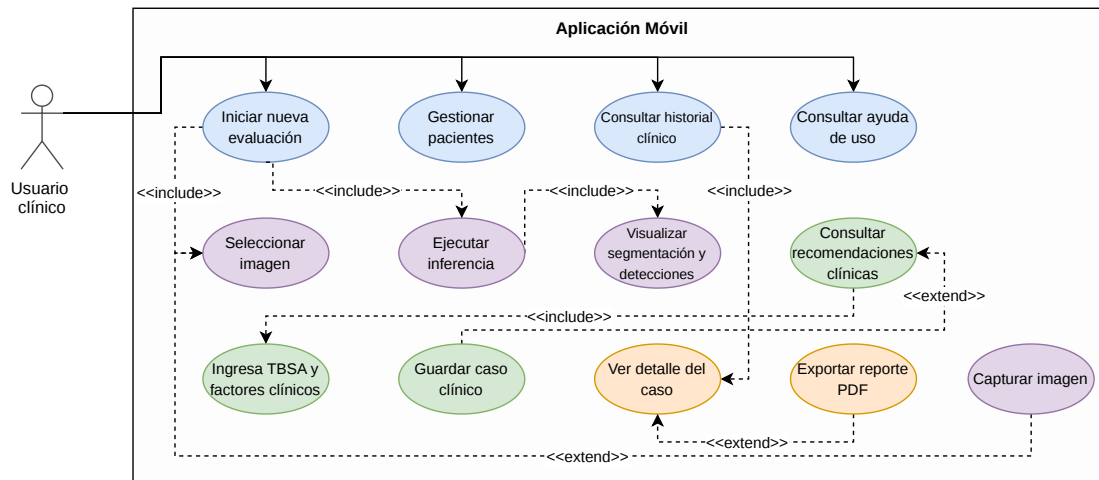


Figura 3.4: Caso de uso de la aplicación móvil: Elaborado por el autor

Conversión y optimización del modelo para despliegue en dispositivos móviles El modelo de inteligencia artificial fue entrenado previamente en un entorno de escritorio utilizando el framework PyTorch y almacenado en formato `best.pt`. Posteriormente, se realizó su conversión a TensorFlow Lite (TFLite) para habilitar su ejecución en dispositivos móviles Android, obteniendo un archivo `.tflite` optimizado para inferencia local.

Durante este proceso se aplicó cuantización de punto flotante de 16 bits (FP16), generando el archivo `best_float16.tflite`. Esta optimización permitió reducir el tamaño del modelo y mejorar la eficiencia computacional durante la inferencia móvil, manteniendo un desempeño competitivo respecto a la versión original en precisión completa.

El modelo convertido conserva las capacidades de detección y segmentación de YOLO11m-seg, permitiendo ejecutar inferencia local directamente en el dispositivo móvil mediante TensorFlow Lite.



3.7.4. Pipeline de procesamiento e inferencia

El sistema desarrollado se implementó con un *pipeline* de procesamiento e inferencia diseñado para procesar imágenes clínicas de quemaduras en la piel mediante técnicas de visión por computadora. La implementación permite procesar imágenes clínicas en dispositivos móviles sin la necesidad de conexiones o servidores externos. Una vez procesada la inferencia, el sistema muestra resultados visiblemente interpretables para el usuario. El **pipeline** de procesamiento e inferencia del sistema se compone de las siguientes etapas principales:

1. Adquisición de imagen
2. Preprocesamiento de imagen
3. Conversión a tensor de entrada
4. Inferencia mediante modelo de inteligencia artificial
5. Postprocesamiento de resultados
6. Generación de resultados visuales
7. Cálculo de parámetros clínicos

Cada etapa del sistema se ejecuta de manera secuencial y controlada, permitiendo la transformación progresiva de la imagen clínica desde su adquisición hasta la generación de resultados clínicamente interpretables.

Etapas del *pipeline* de procesamiento.

Adquisición de imagen. Es la primera etapa del *pipeline*, la imagen se puede cargar desde la galería del dispositivo o desde la cámara. La imagen se guarda en formato **Bitmap**, este formato es la estructura de datos utilizada para almacenamiento en dispositivos Android como una matriz bidimensional de píxeles en memoria y es la entrada inicial del sistema.

Preprocesamiento de imagen. Una vez que la imagen es cargada, se prepara para tener el formato adecuado para la inferencia. Las operaciones realizadas en esta etapa incluyen el redimensionamiento de la imagen (640 x 640) compatible con la entrada del modelo, la normalización de



los valores de intensidad (0 - 1), la conversión de la imagen a tensores compatibles con TensorFlow Lite y la conversión del formato de color (RGB).

Durante el preprocesamiento, la imagen original se adapta al tamaño requerido por el modelo mediante una técnica conocida como redimensionamiento con conservación de proporciones (letterbox). Este procedimiento permite ajustar la imagen al tamaño de entrada del modelo (640×640 píxeles) sin distorsionar su contenido.

Para lograrlo, se calcula un factor de escala y un desplazamiento de relleno (*padding*) que permiten centrar la imagen dentro del nuevo tamaño.

El factor de escala se calcula como:

$$\text{scale} = \min \left(\frac{640}{\text{ancho_original}}, \frac{640}{\text{alto_original}} \right) \quad (3.2)$$

Posteriormente, se calculan los valores de padding horizontal y vertical:

$$\text{padX} = \frac{640 - \text{ancho_redimensionado}}{2} \quad (3.3)$$

$$\text{padY} = \frac{640 - \text{alto_redimensionado}}{2} \quad (3.4)$$

Estos parámetros se almacenan durante el preprocesamiento y se utilizan posteriormente en la etapa de postprocesamiento para transformar las coordenadas generadas por el modelo al tamaño original de la imagen.

Una vez preparada la imagen los datos de entrada se convierten en estructura matricial, llamado tensor de entrada que es compatible con el modelo de inteligencia artificial, y presenta las siguientes características:

- Dimensión: [1, 640, 640, 3]
- Tipo de dato: float32
- Número de canales: 3 (RGB)

Inferencia mediante el modelo de inteligencia artificial. Durante la inferencia el modelo YOLOv11-seg convertido con TFLite, analiza la imagen de entrada para hacer predicciones e iden-



tificar las lesiones por quemaduras. El modelo, recibe la imagen normalizada en formato numérico, YOLOv11-seg genera dos tensores principales como salida durante el proceso de inferencia:

- Tensor de detecciones:
 - Dimensión: [1, 39, 8400]
 - Contenido: coordenadas de cajas delimitadoras, probabilidades de clase y coeficientes de máscara
- Tensor de prototipos de segmentación:
 - Dimensión: [1, 160, 160, 32]
 - Contenido: mapas de características utilizados para la generación de máscaras de segmentación

El tensor de detecciones contiene información estructurada sobre las posibles regiones de interés identificadas en la imagen, incluyendo las coordenadas de las cajas delimitadoras, la probabilidad asociada a cada clase clínica y los coeficientes necesarios para la reconstrucción de las máscaras de segmentación.

El tensor de prototipos de segmentación corresponde a un conjunto de mapas de características generados por la red neuronal, los cuales se combinan posteriormente con los coeficientes de máscara para reconstruir las regiones segmentadas a nivel de píxel.

Parámetros operativos del modelo. Durante la ejecución del modelo en el dispositivo móvil, se utilizaron los siguientes parámetros de configuración:

- Tamaño de entrada: 640 x 640 píxeles
- Umbral de confianza: 0.70
- Umbral de supresión de no máximos (IoU): 0.45
- Número máximo de detecciones: 60

En la figura 3.5 se muestra el sistema de procesamiento e inferencia completo de la aplicación.

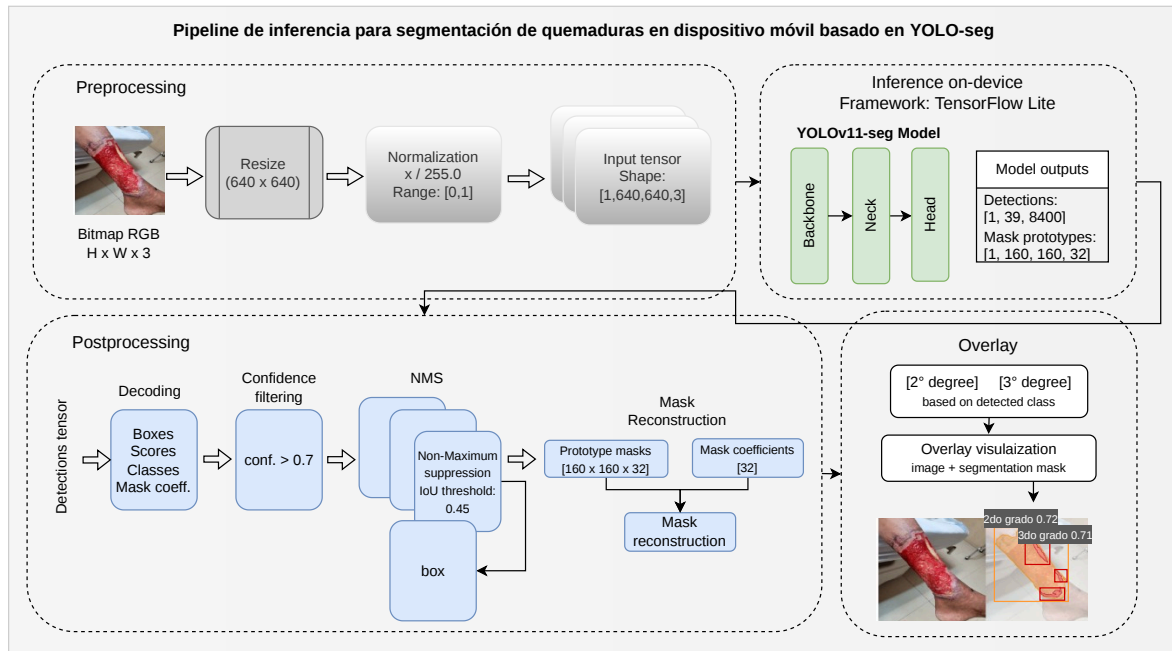


Figura 3.5: Pipeline de inferencia de la aplicación móvil: Elaborado por el autor

Estos parámetros permiten controlar el comportamiento del *pipeline* durante la inferencia, optimizando el equilibrio entre precisión, estabilidad visual de las detecciones y eficiencia computacional en dispositivos móviles. Los valores de los parámetros fueron definidos experimentalmente durante las pruebas de validación del sistema, buscando reducir la aparición de detecciones redundantes y mejorar la confiabilidad de los resultados visuales generados por la aplicación.

Postprocesamiento de resultados. Una vez que el modelo genera predicciones, el sistema ejecuta operaciones correspondientes para transformar los resultados en información utilizable. El postprocesamiento incluye:

- Filtrado de detecciones por nivel de confianza (permite eliminar predicciones con baja probabilidad de pertenencia a una clase válida)
- Aplicación de supresión de no máximos (Non-Maximum Suppression, NMS), que elimina detecciones redundantes que se superponen en la imagen
- Construcción de máscaras de segmentación
- Cálculo de áreas segmentadas



Las máscaras de segmentación se generan mediante la combinación de coeficientes de máscara y prototipos de segmentación para permitir la reconstrucción de las regiones correspondientes de cada clase detectada por el modelo de inteligencia artificial.

Las coordenadas generadas por el modelo durante la inferencia se encuentran definidas en el espacio de la imagen redimensionada a 640×640 píxeles. Para representar correctamente las detecciones en la imagen original, el sistema realiza una transformación geométrica utilizando los parámetros de escala y padding calculados durante el preprocesamiento.

La transformación de coordenadas se realiza mediante las siguientes relaciones geométricas:

$$u_{\text{original}} = \frac{u_{\text{modelo}} - \text{padX}}{\text{scale}} \quad (3.5)$$

$$v_{\text{original}} = \frac{v_{\text{modelo}} - \text{padY}}{\text{scale}} \quad (3.6)$$

donde u_{modelo} y v_{modelo} corresponden a las coordenadas obtenidas en el espacio de entrada del modelo, mientras que u_{original} y v_{original} representan las coordenadas reescaladas a la resolución original de la imagen.

Este procedimiento permite ubicar correctamente las detecciones y las máscaras de segmentación en la imagen original, manteniendo la coherencia espacial entre la salida del modelo y la representación visual generada por el sistema.

Cálculo del valor de confianza. Para cada detección generada por el modelo, el sistema obtiene las probabilidades asociadas a las clases de quemadura mediante la aplicación de una función sigmoide sobre los valores de salida del modelo. Después, se selecciona la clase con mayor probabilidad y dicho valor se utiliza como nivel de confianza de la detección.

En este trabajo, únicamente se conservaron aquellas detecciones cuyo valor de confianza fue superior al umbral definido experimentalmente. Este filtrado permite descartar predicciones con baja probabilidad y reducir la presencia de falsos positivos en la visualización final.

$$\text{confianza} = \text{máx}(\sigma(l_c)) \quad (3.7)$$

donde l_c representa el valor de salida asociado a cada clase y σ corresponde a la función sigmoide.



Reconstrucción de máscaras de segmentación. Después del filtrado por confianza y de la aplicación de NMS, el sistema reconstruye la máscara correspondiente a cada detección válida. Primero, se combinan los coeficientes de máscara generados por la salida de detección con los prototipos de segmentación producidos por el modelo. Esta combinación permite obtener una máscara individual asociada a cada objeto detectado.

Después, la máscara reconstruida se ajusta al área delimitada por la caja de detección y se transforma al espacio de la imagen original utilizando los parámetros de escala y padding calculados durante el preprocesamiento. De esta manera, la región segmentada mantiene correspondencia espacial con el área de lesión observada en la imagen clínica.

Generación del *overlay* de visualización. En esta etapa, el sistema produce la representación visual de los resultados obtenidos mediante el modelo de inteligencia artificial. Las operaciones realizadas incluyen:

- Superposición de máscaras de segmentación
- Dibujo de cajas delimitadoras
- Visualización de etiquetas de clase
- Visualización del nivel de confianza

Las máscaras se representan mediante colores que permiten identificar visualmente las áreas afectadas por la lesión. Este enfoque facilita la interpretación de los resultados por parte del usuario clínico. La generación de resultados visuales se realiza sobre la imagen original procesada, permitiendo la representación gráfica de las regiones detectadas y segmentadas mediante la superposición de elementos visuales generados por el sistema sobre la imagen clínica original.

Cálculo de parámetros clínicos. La última etapa del *pipeline* consiste en la generación de parámetros clínicos derivados del análisis de la imagen. El sistema calcula automáticamente indicadores relevantes para la evaluación de la lesión, incluyendo:

- Porcentaje de superficie corporal afectada (TBSA) estimado por el personal clínico mediante criterios convencionales de evaluación
- Clasificación de severidad dominante detectada por el sistema



- Prioridad de atención clínica basado en criterios clínicos internacionales

Estas recomendaciones no sustituyen la valoración médica especializada, sino que funcionan como apoyo orientativo para la evaluación inicial de la lesión.

Los criterios implementados en el sistema se fundamentan en recomendaciones internacionales para la referencia y manejo inicial de pacientes con quemaduras, particularmente en las guías clínicas propuestas por la American Burn Association (ABA). Por ejemplo, establecen que las quemaduras de espesor total, las lesiones extensas y aquellas que afectan zonas críticas requieren evaluación médica urgente.

El sistema utiliza estos criterios como base para clasificar automáticamente la prioridad de atención clínica en tres niveles:

- Atención urgente
- Atención el mismo día
- Vigilancia ambulatoria

Las recomendaciones proporcionadas por el sistema no determinan un diagnóstico definitivo, sirven como apoyo al usuario clínico, basado en estándares utilizados frecuentemente en unidades de quemaduras.

Estos parámetros son calculados automáticamente a partir de las áreas segmentadas generadas durante el postprocesamiento, permitiendo la generación de indicadores clínicos relevantes para la evaluación inicial de la lesión.

El sistema de recomendaciones clínicas se basa en un conjunto de reglas definidas mediante un árbol de decisión, el cual permite determinar la prioridad de atención médica en función del grado de la quemadura, el porcentaje de superficie corporal afectada (TBSA) y la presencia de factores clínicos de riesgo, como se muestra en la Figura 3.6.

Generación de recomendaciones de primeros auxilios. Dentro del funcionamiento del sistema, la aplicación móvil proporciona recomendaciones automáticas básicas de primeros auxilios una vez que se han mostrado los resultados visuales y activado la generación de recomendaciones.



Sistema de decisión clínica basado en reglas para priorización de atención

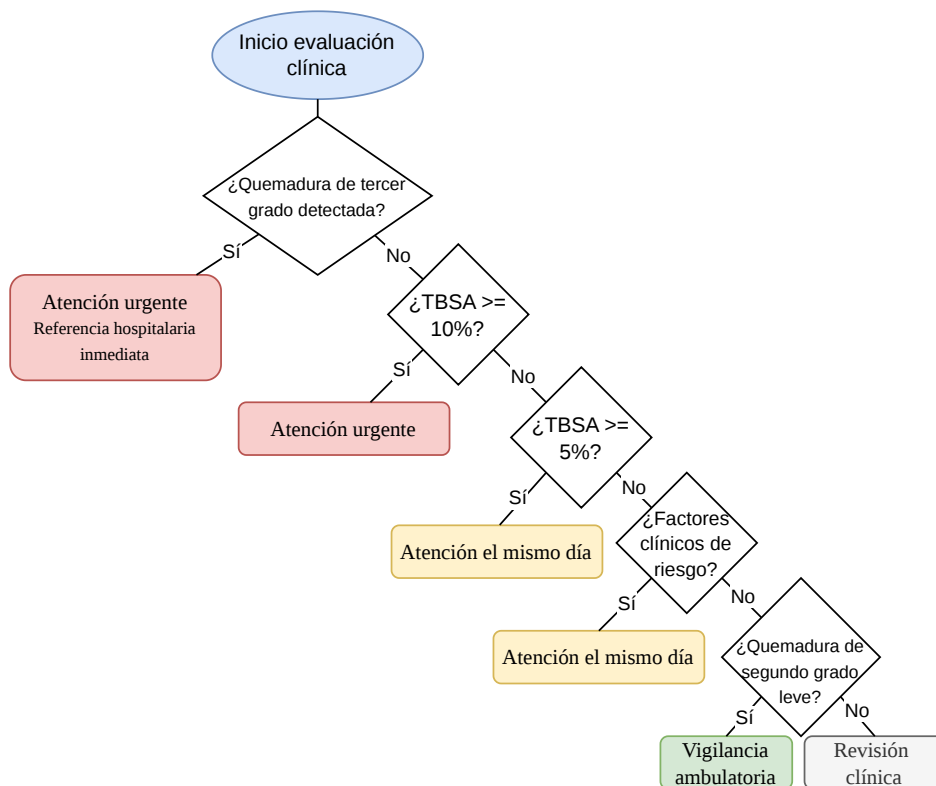


Figura 3.6: Árbol de decisión clínica utilizado para la generación de recomendaciones en el sistema.

Elaboración propia



Las recomendaciones tienen como objetivo orientar al usuario sobre las acciones iniciales que pueden realizarse inmediatamente después de una lesión por quemadura, mientras se busca atención médica especializada.

Las principales indicaciones mostradas por el sistema incluyen:

- Enfriar la zona afectada con agua corriente fresca o tibia durante aproximadamente 20 minutos
- No aplicar hielo, cremas, grasas u otras sustancias sobre la lesión
- Retirar ropa, accesorios o joyería cercana a la zona afectada, siempre que no se encuentren adheridos a la piel
- Cubrir la lesión con material limpio y no adherente
- Mantener al paciente abrigado, especialmente cuando la superficie afectada es extensa

Estas recomendaciones se basan en lineamientos internacionales de atención inicial a pacientes con quemaduras y se muestran únicamente con fines informativos [73].

Capítulo 4: Resultados

4.1. Resultados de la experimentación con YOLOv11-seg

En esta sección se presentan los resultados obtenidos durante el proceso de experimentación realizado con YOLO11-seg para la segmentación de quemaduras de segundo y tercer grado. Los experimentos fueron diseñados con el objetivo de analizar el efecto de diferentes configuraciones de entrenamiento sobre el desempeño del modelo. La evaluación se realizó sobre el conjunto de prueba, compuesto por 56 imágenes clínicas, utilizando las métricas Dice, IoU, precisión y recall.

Las métricas globales reportadas en esta sección fueron calculadas a nivel de imagen fusionando las clases de quemadura de segundo y tercer grado en una sola máscara binaria de región lesionada. De esta manera, la evaluación cuantificó el solapamiento espacial entre la región de quemadura predicha y la anotación de referencia, considerando la tarea como una segmentación binaria de quemadura frente al fondo.

4.1.1. Resultados del experimento 1

En el primer experimento se evaluó el efecto de la resolución de entrada, el número de épocas y el tamaño del modelo sobre el desempeño de segmentación. Los resultados obtenidos se presentan en la Tabla 4.1.

Tabla 4.1: Resultados obtenidos en el experimento 1.

Experimento	Modelo	ImgSz	Épocas	Dice	IoU	Precisión	Recall
Baseline	YOLO11m-seg	640	200	0.8196	0.7369	0.8395	0.8452
Configuración 1	YOLO11m-seg	768	200	0.8147	0.7323	0.8711	0.8307
Configuración 2	YOLO11m-seg	640	300	0.7984	0.7111	0.8619	0.7918
Configuración 3	YOLO11s-seg	640	200	0.8240	0.7333	0.8616	0.8320

Los resultados muestran que el incremento de resolución a 768×768 píxeles mantuvo un



desempeño competitivo, aunque no produjo mejoras globales consistentes respecto a la configuración base. El aumento a 300 épocas tampoco mejoró el desempeño global bajo este protocolo. Aunque YOLO11s-seg obtuvo el mayor Dice promedio en esta etapa, YOLO11m-seg mantuvo un mejor equilibrio entre IoU, recall y comportamiento en las curvas de entrenamiento. A partir de este comportamiento, el segundo experimento se enfocó en continuar la optimización de la variante YOLO11m-seg, manteniéndola como arquitectura base debido a su mejor equilibrio general entre desempeño y capacidad del modelo.

4.1.2. Resultados del experimento 2 y selección del modelo final

En esta fase se compararon tres estrategias derivadas de los resultados del experimento anterior: combinación de mayor resolución con 300 épocas, augmentación clínica conservadora y reducción del tamaño de *batch* a 8. Los resultados se presentan en la Tabla 4.2.

Tabla 4.2: Resultados obtenidos en el experimento 2 bajo el protocolo experimental definitivo.

Experimento	Resolución	<i>Batch</i>	Dice	IoU	Precisión	Recall
y11m_768_e300	768	16	0.8139	0.7332	0.8236	0.8319
y11m_640_e300_augclin	640	16	0.7986	0.7069	0.8605	0.8051
y11m_640_e300_b8	640	8	0.8194	0.7392	0.8204	0.8644

La configuración YOLO11m-seg con resolución de 640×640 píxeles, 300 épocas y *Batch* de 8 obtuvo el mejor equilibrio entre Dice, IoU y recall, alcanzando valores de 0.8194, 0.7392 y 0.8644, respectivamente. Aunque la configuración con augmentación clínica conservadora presentó la mayor precisión promedio, obtuvo menor Dice, IoU y recall. Por otra parte, el incremento de resolución mantuvo un desempeño competitivo, pero no superó a la configuración con *batch size* reducido.

A partir de estos resultados, la configuración YOLO11m-seg con resolución de 640×640 píxeles, 300 épocas y *batch size* de 8 fue seleccionada como configuración final para el sistema propuesto y como base para el *benchmark* comparativo *multi-seed*. Las curvas completas de entrenamiento se presentan en el Anexo Anexo A.



Los resultados cualitativos, incluyendo visualizaciones tipo *trptych*, *overlays* y mapas de error, se presentan en el Anexo Anexo B.

4.2. Resultados del benchmark comparativo multi-seed

En esta sección se presentan los resultados del *benchmark* comparativo *multi-seed* realizado con 10 corridas independientes por arquitectura. Los resultados se reportan como media y desviación estándar entre corridas, utilizando las métricas globales Dice, IoU, precisión y recall sobre el conjunto de prueba.

Las métricas globales fueron calculadas fusionando las clases de quemadura de segundo y tercer grado en una sola máscara de región lesionada, con el objetivo de evaluar la capacidad general de segmentación de cada modelo frente al fondo.

4.2.1. Resultados globales promedio

La Tabla 4.3 presenta los resultados globales promedio obtenidos durante el *benchmark multi-seed*, calculados mediante una evaluación binaria de región lesionada frente a fondo. Bajo este protocolo, Attention U-Net obtuvo los valores promedio más altos de Dice, IoU y precisión global, mientras que YOLO11m-seg presentó el mayor valor promedio de recall.

Las arquitecturas basadas en U-Net mostraron valores globales cercanos entre sí y una menor variabilidad entre corridas. Por su parte, YOLO11m-seg presentó una mayor dispersión entre semillas, aunque mantuvo el mayor valor promedio de recall global.

Tabla 4.3: Resultados globales promedio (10 corridas) del *benchmark multi-seed* utilizando métricas binarias de quemadura frente a fondo.

Modelo	Dice	IoU	Precisión	Recall
YOLO11m-seg	0.8252 ± 0.0264	0.7393 ± 0.0338	0.8360 ± 0.0342	0.8643 ± 0.0232
U-Net	0.8518 ± 0.0045	0.7645 ± 0.0061	0.8918 ± 0.0167	0.8461 ± 0.0122
Attention U-Net	0.8552 ± 0.0113	0.7712 ± 0.0140	0.8938 ± 0.0177	0.8508 ± 0.0161
U-Net++	0.8515 ± 0.0102	0.7630 ± 0.0116	0.8922 ± 0.0134	0.8451 ± 0.0140

4.2.2. Distribución de métricas entre corridas

La Figura 4.1 muestra la distribución de las métricas globales obtenidas durante las 10 corridas independientes por arquitectura. Los boxplots permiten visualizar la estabilidad, dispersión y variabilidad del desempeño entre modelos para Dice global, IoU global, precisión y recall.

U-Net presentó la menor dispersión entre corridas en las métricas Dice e IoU, reflejando un comportamiento de entrenamiento relativamente estable. Por otra parte, YOLO11m-seg mostró los valores más altos en Recall, aunque presentó una mayor dispersión entre corridas, observable en los boxplots, especialmente en Dice e IoU. Esto indica una mayor sensibilidad a la inicialización aleatoria y una menor estabilidad del entrenamiento respecto a las arquitecturas basadas en U-Net.

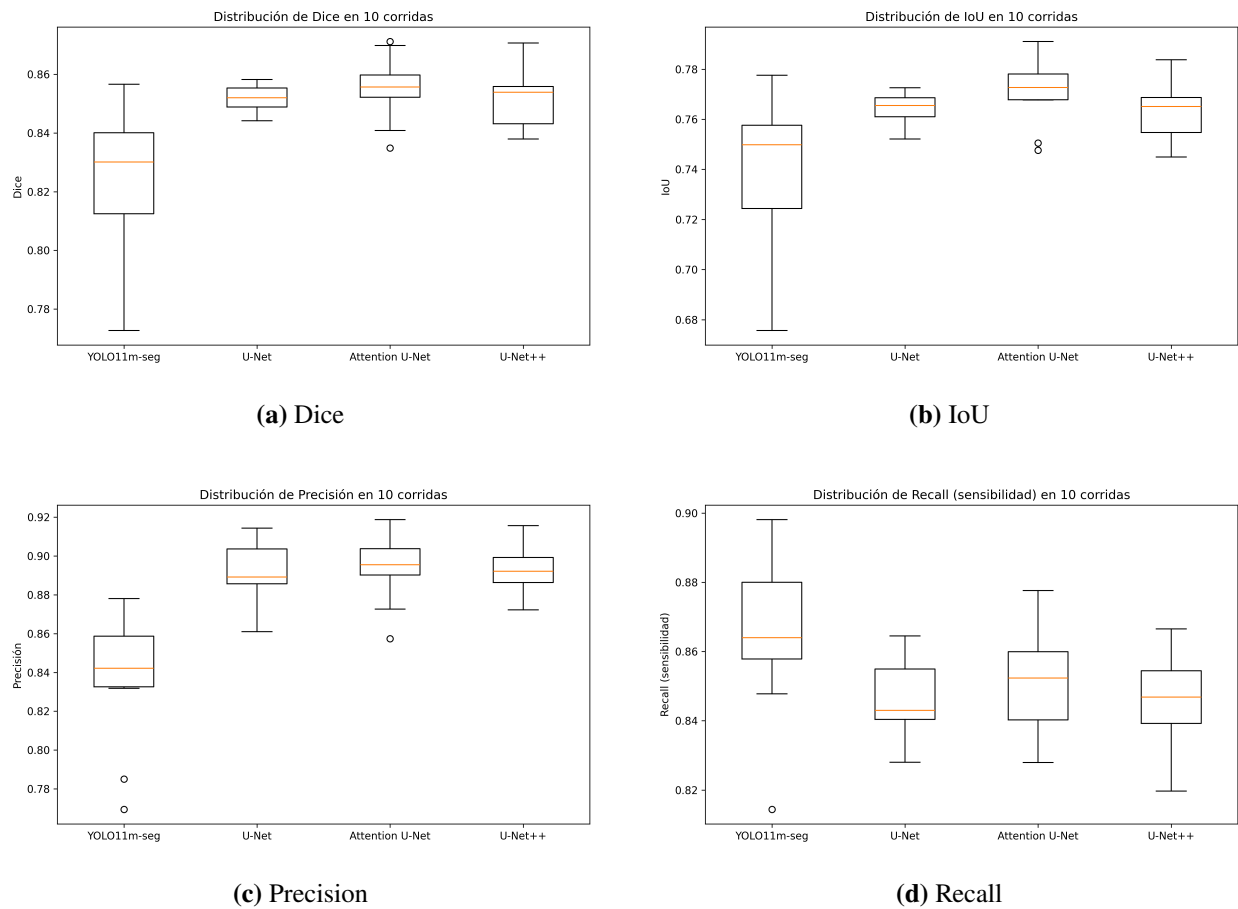


Figura 4.1: Distribución de métricas obtenidas durante las 10 corridas independientes para cada arquitectura evaluada.



4.2.3. Análisis estadístico

La prueba de Kruskal-Wallis detectó diferencias estadísticamente significativas entre las arquitecturas evaluadas para Dice ($p = 0,0101$), IoU ($p = 0,0235$) y precisión ($p = 0,000185$). En contraste, no se observaron diferencias estadísticamente significativas para recall ($p = 0,0522$).

Tabla 4.4: Resultados de la prueba de Kruskal-Wallis para el *benchmark multi-seed* con métricas globales binarias.

Métrica	Estadístico H	p-value
Dice	11.32	0.010132
IoU	9.48	0.023498
Precisión	19.82	0.000185
Recall	7.72	0.052247

Posteriormente, se realizaron comparaciones post-hoc mediante la prueba de Dunn con corrección de Bonferroni. Los resultados mostraron diferencias significativas en Dice entre YOLO11m-seg y Attention U-Net, así como entre YOLO11m-seg y U-Net. Para precisión, YOLO11m-seg presentó diferencias significativas respecto a las tres arquitecturas basadas en U-Net. En IoU y recall no se observaron diferencias significativas entre pares de modelos después de la corrección de Bonferroni.

Tabla 4.5: Comparaciones post-hoc significativas mediante Dunn con corrección de Bonferroni.

Comparación	Métrica	p-value ajustado
Attention U-Net vs YOLO11m-seg	Dice	0.0390
U-Net vs YOLO11m-seg	Dice	0.0489
Attention U-Net vs YOLO11m-seg	Precisión	0.0030
U-Net vs YOLO11m-seg	Precisión	0.0017
U-Net++ vs YOLO11m-seg	Precisión	0.0017



4.2.4. Resultados por clase

La Tabla 4.6 presenta los resultados promedio por clase obtenidos durante el *benchmark multi-seed*, excluyendo la clase *background*. Este análisis permite evaluar de manera independiente el desempeño de cada arquitectura en las clases *2° grado* y *3° grado*.

Tabla 4.6: Resultados promedio por clase obtenidos durante el *benchmark multi-seed*. La clase *background* fue excluida del análisis.

Modelo	Clase	Dice	IoU	Precision	Recall
U-Net	burn_2nd	0.7088 ± 0.0151	0.5491 ± 0.0179	0.7682 ± 0.0225	0.6591 ± 0.0308
U-Net	burn_3rd	0.8223 ± 0.0113	0.6984 ± 0.0162	0.8090 ± 0.0291	0.8374 ± 0.0206
Attention U-Net	burn_2nd	0.7143 ± 0.0189	0.5559 ± 0.0225	0.7590 ± 0.0148	0.6753 ± 0.0317
Attention U-Net	burn_3rd	0.8253 ± 0.0082	0.7026 ± 0.0118	0.8235 ± 0.0186	0.8277 ± 0.0200
U-Net++	burn_2nd	0.7034 ± 0.0212	0.5428 ± 0.0251	0.7583 ± 0.0180	0.6569 ± 0.0357
U-Net++	burn_3rd	0.8198 ± 0.0166	0.6949 ± 0.0239	0.8109 ± 0.0355	0.8303 ± 0.0226
YOLO11m-seg	burn_2nd	0.7062 ± 0.0593	0.5489 ± 0.0682	0.7220 ± 0.0739	0.6946 ± 0.0622
YOLO11m-seg	burn_3rd	0.8223 ± 0.0248	0.6989 ± 0.0357	0.7672 ± 0.0410	0.8876 ± 0.0254

Los resultados muestran que las arquitecturas basadas en U-Net obtuvieron un desempeño alto y estable en las métricas por clase. Attention U-Net presentó los mejores valores de Dice e IoU para ambas clases, alcanzando un Dice de 0.7143 para *2° grado* y 0.8253 para *3° grado*. U-Net, U-Net++ y YOLO11m-seg mostraron valores cercanos, especialmente para la clase *3° grado*. Por otro lado el modelo YOLO11m-seg obtuvo los mayores valores de Recall para ambas clases.

4.2.5. Análisis de matrices de confusión entre clases

Con el objetivo de analizar el comportamiento de los modelos a nivel de clasificación semántica a nivel píxel entre grados de quemadura, se generaron matrices de confusión normalizadas considerando únicamente los píxeles cuya clase de referencia correspondía a quemadura de segundo o tercer grado. A diferencia de una matriz 2×2 , esta representación 2×3 permite distinguir entre píxeles omitidos hacia fondo y píxeles confundidos entre clases clínicas.

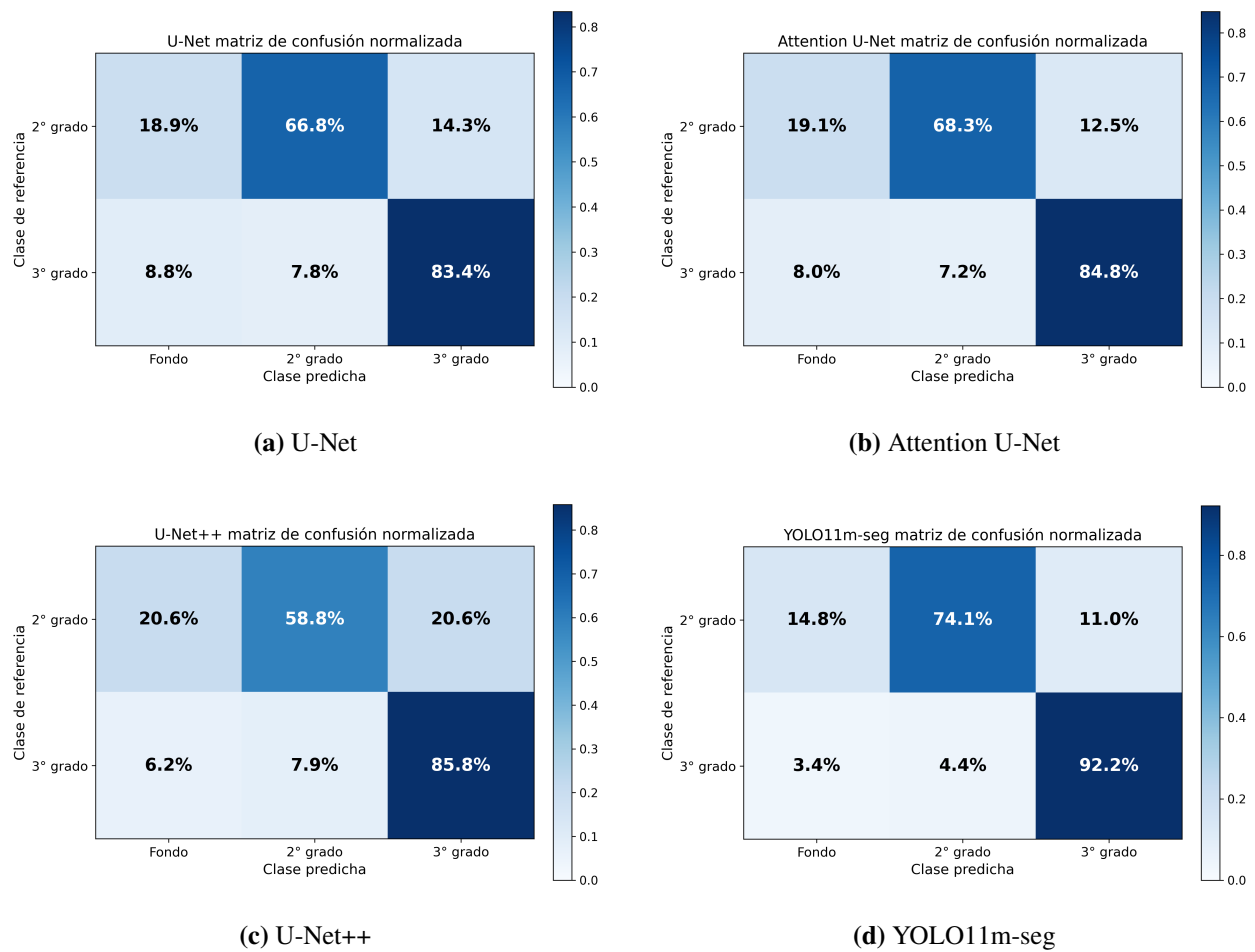


Figura 4.2: Matrices de confusión normalizadas para las arquitecturas evaluadas. Las filas representan la clase de referencia y las columnas representan la clase predicha: fondo, quemadura de segundo grado y quemadura de tercer grado.



Como se observa en la Figura 4.2, las arquitecturas evaluadas presentaron un comportamiento distinto según la clase clínica. En general, todos los modelos clasificaron con mayor consistencia las regiones de quemadura de tercer grado. Para la clase de segundo grado se observaron mayores errores, tanto por omisión hacia fondo como por confusión con quemaduras de tercer grado.

YOLO11m-seg presentó una alta proporción de píxeles de segundo grado clasificados correctamente como segundo grado, así como el mayor porcentaje de píxeles de tercer grado clasificados correctamente. Estos resultados muestran confusión entre segundo y tercer grado en una parte de las regiones evaluadas. Este comportamiento complementa los resultados por clase y permite distinguir entre errores de omisión y errores de asignación semántica.

4.2.6. Resultados de eficiencia computacional

La Tabla 4.7 presenta la comparación de eficiencia computacional entre las arquitecturas evaluadas, considerando número de parámetros, FLOPs, latencia promedio de inferencia, FPS y consumo máximo de memoria GPU. Estas métricas permiten analizar el costo computacional de cada modelo bajo la misma resolución de entrada.

Tabla 4.7: Comparación de eficiencia computacional entre YOLO11m-seg y arquitecturas CNN.

Modelo	Parámetros	FLOPs	Latencia	FPS	VRAM
YOLO11m-seg	22.36M	56.76G	7.02 ms	142.4	261 MB
U-Net	24.44M	49.18G	5.86 ms	170.5	269 MB
U-Net++	26.08M	115.43G	13.05 ms	76.66	622 MB
Attention U-Net	30.12M	62.56G	6.95 ms	143.94	362 MB

U-Net presentó la menor latencia de inferencia y el menor número de FLOPs, mientras que U-Net++ mostró el mayor costo computacional y el mayor consumo de memoria GPU. YOLO11m-seg presentó el menor número de parámetros y el menor consumo de memoria GPU, con una latencia cercana a la observada en Attention U-Net. Estas diferencias muestran que los FLOPs no determinan por sí solos la latencia de inferencia, ya que esta también depende del tipo de operaciones, el uso de memoria, la implementación del modelo y las etapas de postprocesamiento.

La Figura 4.3 muestra la relación entre el desempeño global de segmentación, medido



mediante Dice global, y la complejidad computacional expresada en GFLOPs. Por otra parte, la Figura 4.4 presenta la relación entre la latencia promedio de inferencia y el consumo máximo de memoria GPU (VRAM) de cada arquitectura evaluada.

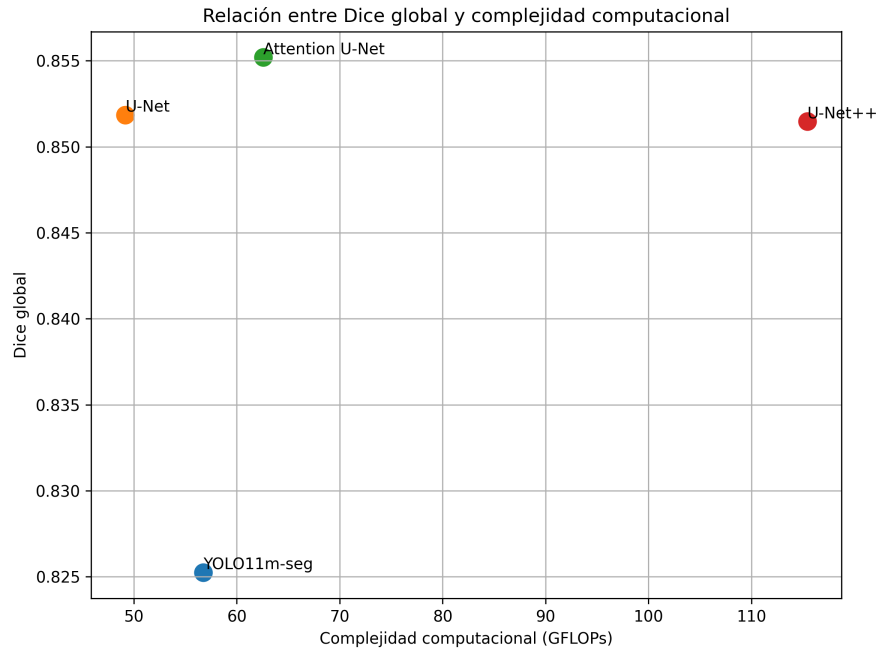


Figura 4.3: Relación entre desempeño global de segmentación, medido mediante Dice global, y complejidad computacional (GFLOPs) de las arquitecturas evaluadas.

En conjunto, los resultados muestran diferencias importantes entre desempeño global, latencia y consumo de memoria. YOLO11m-seg mantuvo un desempeño global relativamente menor con bajo consumo de VRAM, mientras que U-Net presentó la menor latencia de inferencia. Estos resultados permiten analizar el compromiso entre desempeño de segmentación y costo computacional de cada arquitectura.

4.2.7. Resultados cualitativos

Con el objetivo de complementar los resultados cuantitativos obtenidos durante el *benchmark multi-seed*, se realizó una evaluación cualitativa de las segmentaciones generadas por las arquitecturas evaluadas. Las figuras presentadas permiten visualizar el comportamiento de los modelos sobre imágenes clínicas reales del conjunto de prueba.

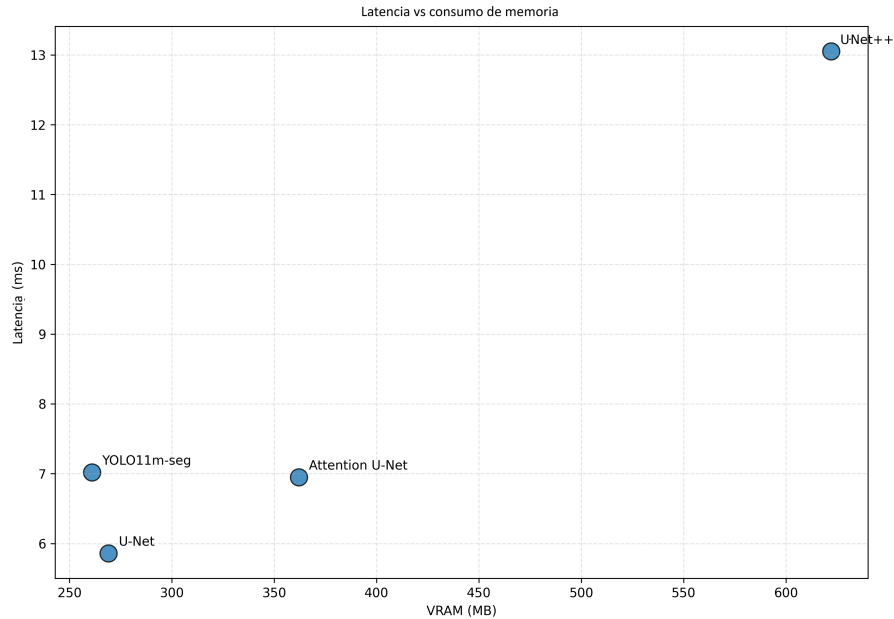


Figura 4.4: Relación entre latencia promedio de inferencia y consumo máximo de memoria GPU (VRAM) de las arquitecturas evaluadas.

La Figura 4.5 muestra ejemplos representativos de segmentación, donde se presenta la imagen original, la máscara de referencia y la predicción generada por cada modelo evaluado.

Por otra parte, la Figura 4.6 presenta mapas de error entre las máscaras de referencia y las predicciones generadas. Estas visualizaciones permiten identificar regiones con falsos positivos, falsos negativos y errores de clasificación entre grados de quemadura.

En las regiones correspondientes a quemaduras extensas y bien delimitadas, las arquitecturas evaluadas mostraron un comportamiento consistente.

4.2.8. Capacidad funcional de las arquitecturas

Además de las métricas cuantitativas, se compararon las capacidades funcionales de las arquitecturas evaluadas en relación con su posible integración en un sistema móvil. Esta comparación tiene un carácter cualitativo y se basa en las funcionalidades proporcionadas de forma nativa por cada arquitectura, así como en la experiencia obtenida durante su implementación y evaluación en este trabajo. La Tabla 4.8 resume las capacidades principales de cada modelo.

Como se observa en la Tabla 4.8, YOLO11m-seg fue la única arquitectura evaluada que

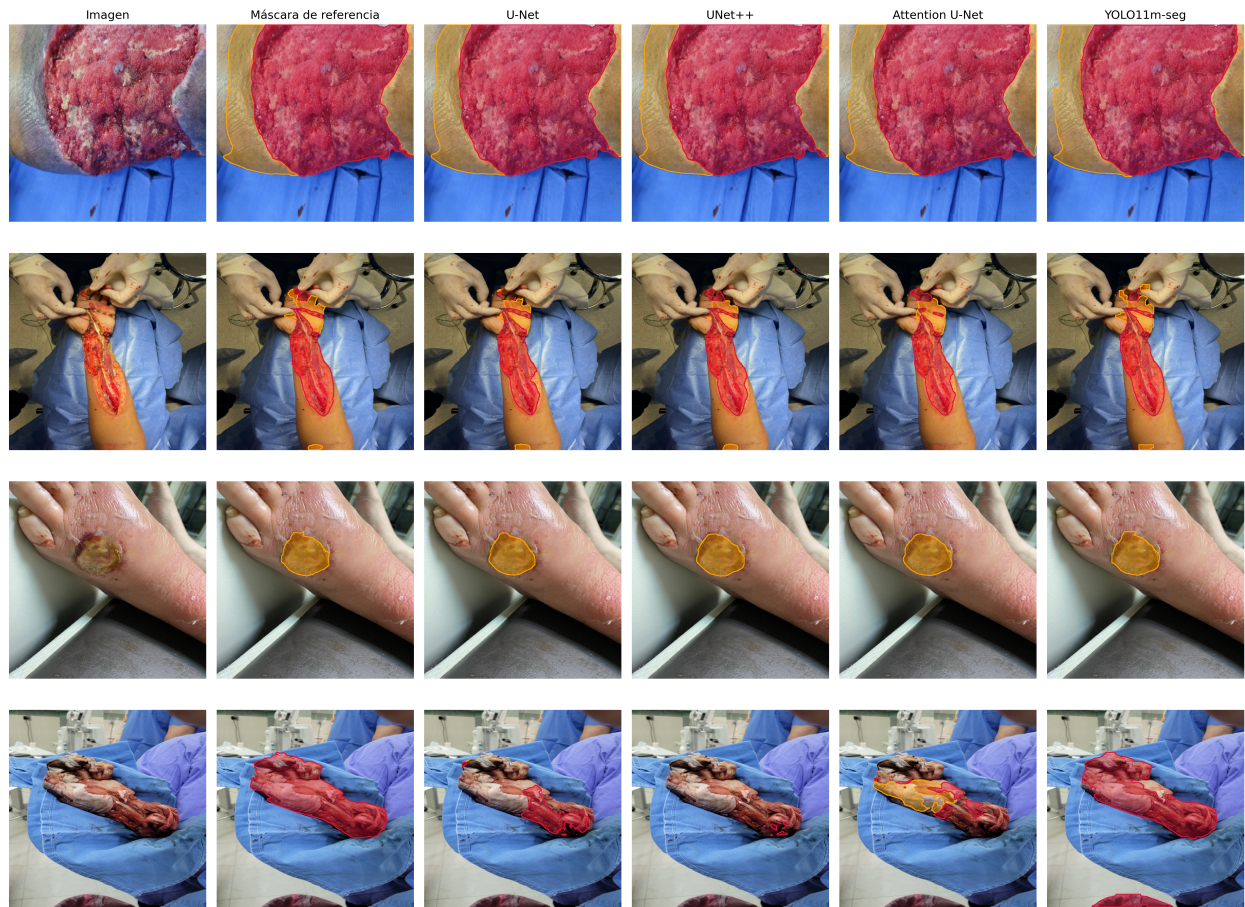


Figura 4.5: Comparación cualitativa entre arquitecturas utilizando imágenes del conjunto de prueba. Las regiones de quemadura de segundo grado se muestran en naranja y las de tercer grado en rojo.

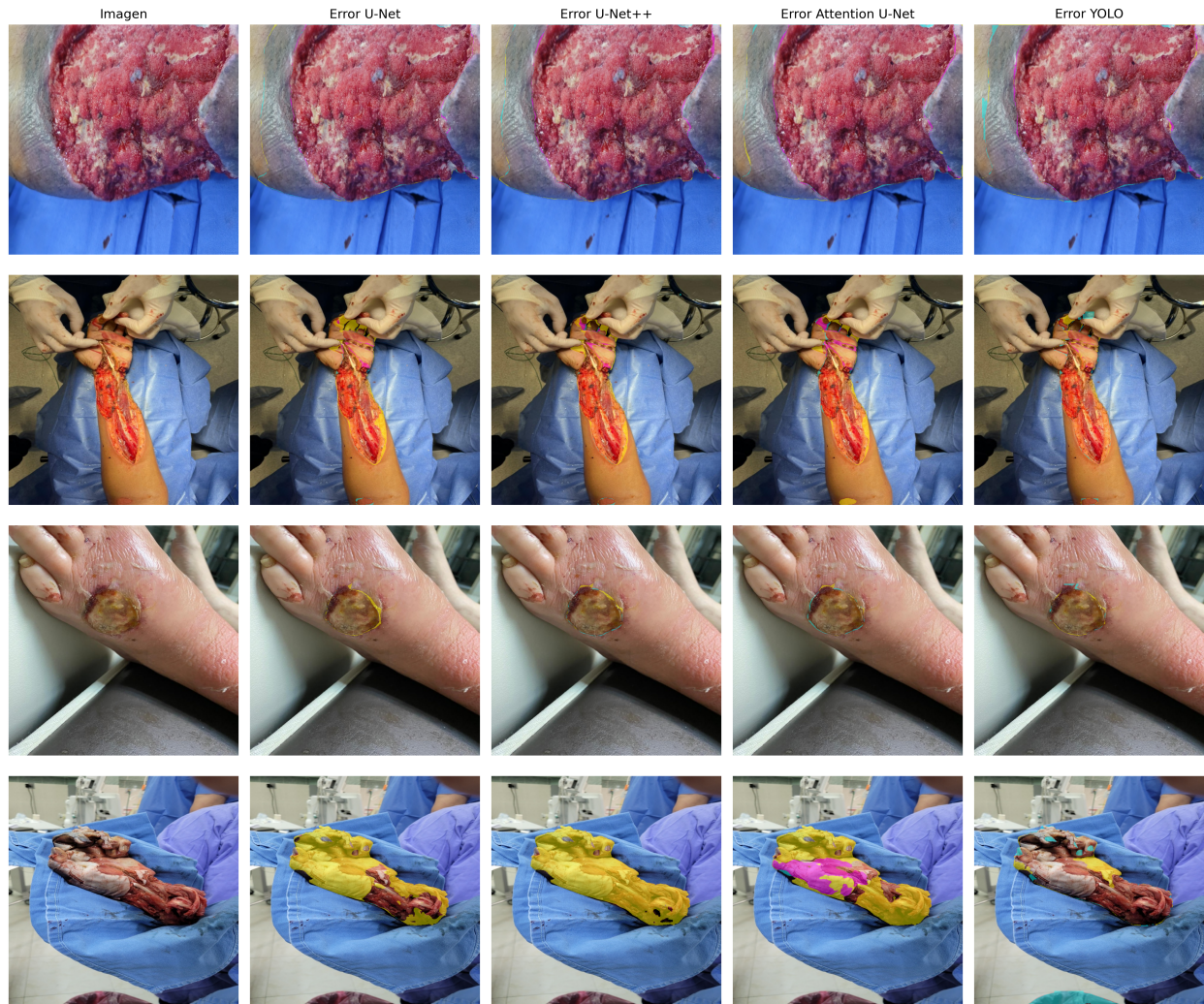


Figura 4.6: Mapas de error generados a partir de las predicciones de los modelos sobre imágenes clínicas del conjunto de prueba; falsos positivos (cian), falsos negativos (amarillo) y errores de clasificación entre grados de quemadura (magenta).



Tabla 4.8: Capacidad funcional de los modelos evaluados en el contexto de segmentación de quemaduras e implementación en dispositivos móviles.

Capacidad	YOLO11m-seg	U-Net	Attention U-Net	U-Net++
Detección de objetos	Sí	No	No	No
Bounding box	Sí	No	No	No
Confianza por predicción	Sí	No	No	No
Segmentación	Sí	Sí	Sí	Sí
Exportación a TFLite	Integrada en el flujo YOLO	Requiere adaptación	Requiere adaptación	Requiere adaptación
Integración móvil	Directa en este trabajo	No implementada	No implementada	No implementada

generó de forma nativa detecciones, cajas delimitadoras, máscaras de instancia, clases y niveles de confianza dentro de una sola inferencia. Estas características resultan útiles para aplicaciones móviles orientadas al apoyo clínico, ya que permiten mostrar información visual e interpretativa directamente al usuario sin requerir etapas adicionales de procesamiento.

Por el contrario, las arquitecturas U-Net, Attention U-Net y U-Net++ fueron diseñadas principalmente para tareas de segmentación semántica. Aunque producen máscaras con una delimitación precisa de las regiones lesionadas, no generan de forma directa información asociada a detección de objetos, cajas delimitadoras ni niveles de confianza por instancia. La obtención de estas salidas requeriría módulos adicionales de postprocesamiento o modelos complementarios.

Asimismo, durante el desarrollo de este trabajo se observó que la exportación e integración de YOLO11m-seg hacia TensorFlow Lite se encuentra soportada dentro del flujo de trabajo de Ultralytics, facilitando su despliegue en Android. En contraste, la implementación móvil de las arquitecturas basadas en U-Net no fue abordada en este estudio y requeriría etapas adicionales de adaptación, conversión y desarrollo de componentes específicos para la generación de resultados equivalentes.

Por lo anterior, la selección de YOLO11m-seg como modelo final no se fundamentó únicamente en su desempeño cuantitativo, sino también en las capacidades funcionales necesarias para construir un sistema móvil capaz de realizar detección, clasificación y segmentación de quemaduras dentro de un único proceso de inferencia.



4.3. Resultados del despliegue en aplicación móvil

Validación funcional del sistema

El sistema se implementó y ejecutó en dispositivos móviles reales. Se verificó la ejecución y el flujo del *pipeline* del sistema, validando la carga de imágenes desde la galería o cámara, así como la generación de resultados visuales y recomendaciones clínicas según los datos del sistema y el usuario. El sistema opero de manera local sin la necesidad de conexiones externas. En la Ilustración 4.1, se muestra una de las secuencias funcionales del sistema que incluye:

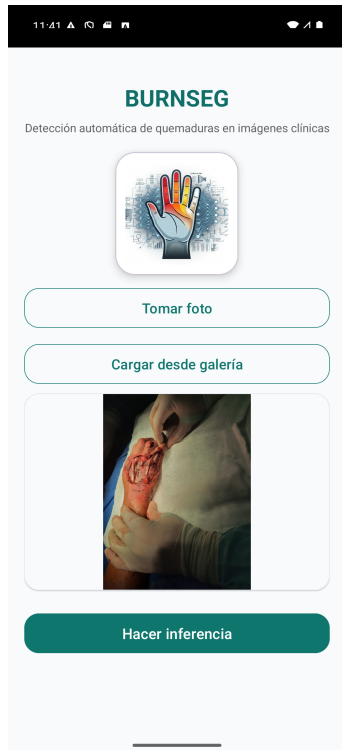
- Carga de imagen en (a)
- Inferencia con resultados visuales en (b)
- Recomendaciones clínicas en (c)
- Visualización del reporte para envío o impresión en (d)

En la Ilustración 4.1, se muestra (a), el historial de inferencias ejecutadas asociadas a cada paciente previamente guardado, el historial está coloreado de acuerdo a la urgencia de recomendación (verde:vigilancia ambulatoria, amarillo:atención el mismo día y rojo:Atención urgente); cada historial contiene datos del paciente, resultados de la inferencia y fecha. En (b) se muestran los pacientes guardados en la memoria, cada paciente contiene además de los datos asociados, el historial de inferencias y la opción de realizar una nueva.

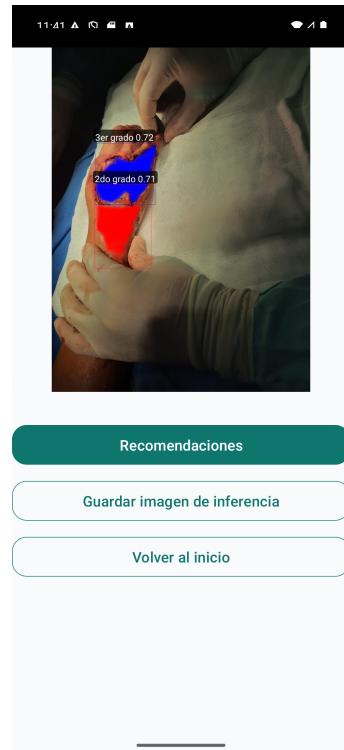
4.3.1. Resultados visuales de segmentación

Detección y segmentación básica La Ilustración 4.3 muestra la imagen original (a) y un ejemplo de resultado generado por el sistema durante la ejecución en un dispositivo móvil en (b). En la imagen de inferencia se observa la detección automática de la región afectada por la quemadura, representada mediante una máscara de segmentación superpuesta sobre la imagen clínica original.

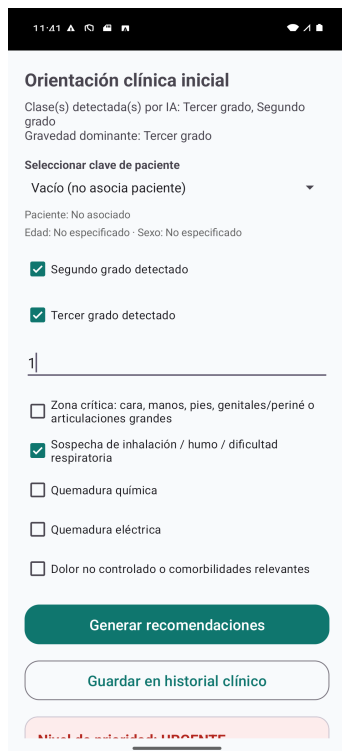
Además, se visualiza una caja delimitadora alrededor de la lesión, la etiqueta de clase correspondiente y el nivel de confianza asociado a la predicción del modelo. Este resultado confirma el correcto funcionamiento del proceso de inferencia y la generación de resultados visuales en el dispositivo móvil.



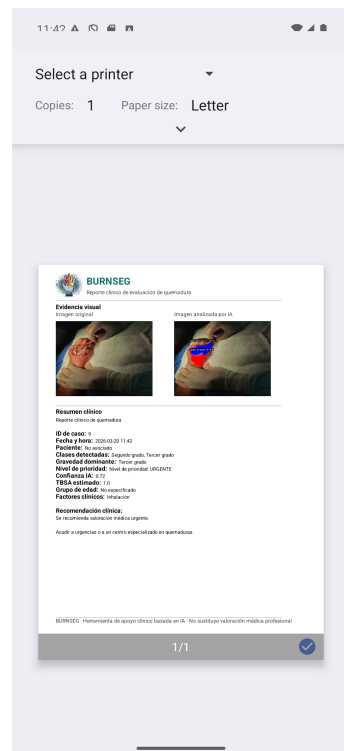
(a)



(b)

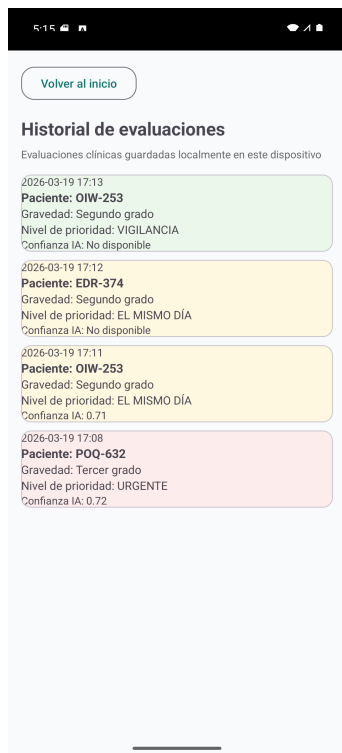


(c)

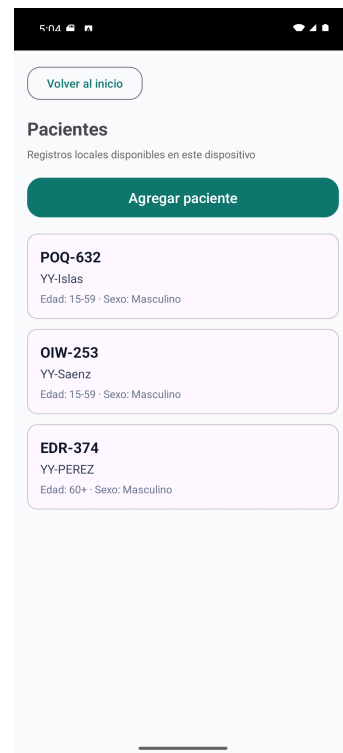


(d)

Ilustración 4.1: Capturas de pantalla del flujo del sistema principal en dispositivo móvil.



(a)

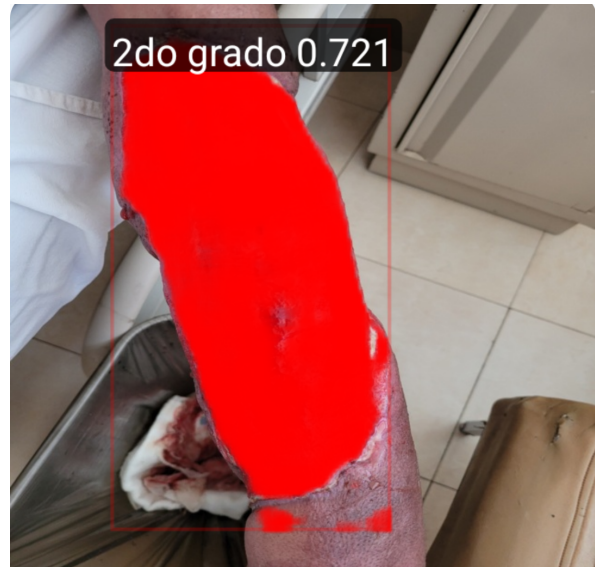


(b)

Ilustración 4.2: Capturas de pantalla del del sistema de almacenamiento de datos en dispositivo móvil.



(a) Imagen original.



(b) Resultado de inferencia.

Ilustración 4.3: Ejemplo de resultado de inferencia del sistema en el dispositivo móvil. A la izquierda se muestra la imagen original y a la derecha el resultado generado por el modelo, incluyendo la segmentación y la detección de la región afectada.

Múltiples regiones detectadas La Ilustración 4.4 presenta la imagen original (a) y un ejemplo en el que el sistema identifica múltiples regiones afectadas dentro de una misma imagen clínica en (b). Cada región detectada se representa mediante una caja delimitadora independiente y una máscara de segmentación correspondiente.

Este resultado evidencia la capacidad del sistema para procesar imágenes con múltiples áreas lesionadas y generar resultados consistentes en escenarios clínicos complejos.

Recomendaciones de primeros auxilios La Ilustración 4.5 muestra la visualización de las recomendaciones de primeros auxilios generadas por el sistema después de la detección y clasificación de la lesión. Estas recomendaciones se presentan de forma automática en la interfaz de usuario con el objetivo de orientar al usuario sobre las acciones iniciales que pueden realizarse inmediatamente después de una quemadura.

La inclusión de esta funcionalidad demuestra la integración del modelo de inteligencia artificial con mecanismos de apoyo a la toma de decisiones iniciales en el contexto de atención a pacientes con lesiones por quemaduras.



(a) Imagen original.



(b) Resultado de inferencia.

Ilustración 4.4: Ejemplo de resultado de inferencia del sistema en el dispositivo móvil. A la izquierda se muestra la imagen original y a la derecha el resultado generado por el modelo, incluyendo la segmentación y la detección de la región afectada.

Primeros auxilios

- Enfriar con agua corriente fresca durante 20 minutos si la lesión es reciente
- No usar hielo
- Retirar ropa o accesorios no adheridos
- Cubrir con material limpio o estéril
- No aplicar pomadas caseras, pasta dental, mantequilla ni remedios irritantes

Ilustración 4.5: Recomendaciones de primeros auxilios mostrados por el sistema



La Ilustración 4.6 muestra un ejemplo del reporte generado automáticamente por el sistema después de la ejecución del proceso de inferencia en el dispositivo móvil. El reporte se genera en formato PDF e incluye la imagen analizada, el resultado de la clasificación del tipo de quemadura, así como información básica del análisis realizado por el sistema.

Esta funcionalidad permite registrar y documentar los resultados obtenidos durante la evaluación, facilitando su almacenamiento y consulta posterior en entornos clínicos o de seguimiento del paciente.

El ejemplo de una de las funcionalidades del sistema (video demostrativo de inferencia sobre la imagen de un paciente) se presenta en el enlace del Anexo Anexo E.

4.3.2. Evaluación del rendimiento computacional

Condiciones de evaluación Se evaluó el rendimiento computacional del sistema mediante la medición del tiempo de ejecución del *pipeline* completo de inferencia en entorno de escritorio. La simulación del flujo de procesamiento incluyó las siguientes etapas:

- Carga de la imagen desde almacenamiento local
- Redimensionamiento de la imagen
- Normalización de los datos de entrada
- Ejecución del modelo YOLO11m-seg
- Decodificación de predicciones
- Aplicación del algoritmo Non-Maximum Suppression (NMS)
- Generación de máscaras de segmentación
- Renderizado de resultados anotados

El dataset utilizado para la evaluación estuvo compuesto por:

- Conjunto de prueba: 110 imágenes
- Resolución de entrada: 640×640 píxeles



BURNSEG

Reporte clínico de evaluación de quemadura

Evidencia visual

Imagen original



Imagen analizada por IA



Resumen clínico

Reporte clínico de quemadura

ID de caso: 8

Fecha y hora: 2026-03-20 11:38

Paciente: POQ-632

Clases detectadas: Segundo grado, Tercer grado

Gravedad dominante: Tercer grado

Nivel de prioridad: Nivel de prioridad: URGENTE

Confianza IA: 0.72

TBSA estimado: 1.0

Grupo de edad: 15-59

Factores clínicos: Inhalación

Recomendación clínica:

Se recomienda valoración médica urgente.

Acudir a urgencias o a un centro especializado en quemaduras.

BURNSEG · Herramienta de apoyo clínico basada en IA · No sustituye valoración médica profesional

Ilustración 4.6: Ejemplo de reporte en formato PDF generado por el sistema



- *Batch size*: 1
- Modelo evaluado: YOLO11m-seg convertido a TensorFlow Lite

Se evaluaron las siguientes métricas de rendimiento:

- Tiempo de preprocesamiento (Preprocess time)
- Tiempo de inferencia del modelo (Inference time)
- Tiempo de postprocesamiento (Postprocess time)
- Tiempo total del *pipeline* (*Total pipeline time*)
- Comparación del tiempo de inferencia entre el modelo YOLO11m-seg en formato PyTorch (best.pt) y el modelo convertido al formato TensorFlow Lite (best.tflite).

Las mediciones se realizaron utilizando registros de tiempo del sistema mediante timestamps de alta precisión, y los resultados se reportaron como media y desviación estándar sobre el conjunto de prueba.

Resultados de latencia del sistema Con el objetivo de evaluar el desempeño temporal del sistema, se realizó la medición del tiempo de ejecución de cada etapa del *pipeline* de inferencia sobre el conjunto de prueba. Los tiempos registrados corresponden a las etapas de preprocesamiento, inferencia y postprocesamiento, así como al tiempo total del *pipeline*.

Los resultados estadísticos obtenidos se presentan en la Tabla 4.9, donde se reportan los valores de media, desviación estándar, mínimo, máximo y mediana del tiempo de procesamiento por imagen, expresados en milisegundos.

Los resultados promedio del tiempo de procesamiento por etapa del *pipeline* se presentan en la Figura 4.7. Como se observa, el tiempo de inferencia representa la mayor proporción del tiempo total de procesamiento, de este modo, resulta consistente en términos del comportamiento esperado en sistemas basados en modelos de *deep learning*.

La variabilidad del tiempo de procesamiento por etapa se muestra en la Figura 4.8. Esta representación permite analizar la dispersión de los tiempos de ejecución y la presencia de valores atípicos durante las ejecuciones del sistema.



Tabla 4.9: Tiempo promedio de procesamiento por etapa del *pipeline* de inferencia.

Métrica	mean_ms	std_ms	min_ms	max_ms	median_ms
Preprocess time	0.766	0.309	0.429	1.696	0.683
Inference time	6.277	2.388	4.372	17.360	5.645
Postprocess time	0.869	0.422	0.163	2.448	0.747
Total pipeline time	17.061	14.917	7.504	67.960	11.649

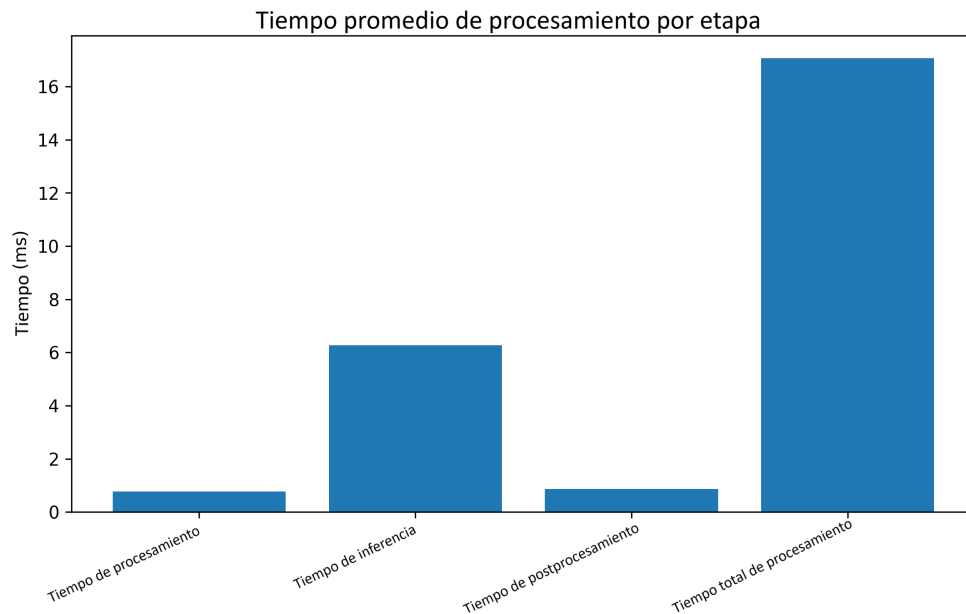


Figura 4.7: Tiempo promedio de procesamiento por etapa del *pipeline* de inferencia.

Se realizó una comparación entre el tiempo de inferencia del modelo evaluado en un entorno de *benchmark* y el tiempo total de procesamiento del sistema desplegado en la aplicación móvil (*pipeline* real, incluyendo las etapas de preprocesamiento, inferencia y postprocesamiento de la imagen). Esta comparación permite identificar la latencia de las etapas adicionales del *pipeline* sobre la latencia total observada en condiciones reales de operación. Los resultados se presentan en la Tabla 4.10.

Clasificación estándar de latencia en sistemas interactivos De acuerdo con la percepción de un usuario en la respuesta de rapidez interactiva de sistemas informáticos, se utilizó una clasificación

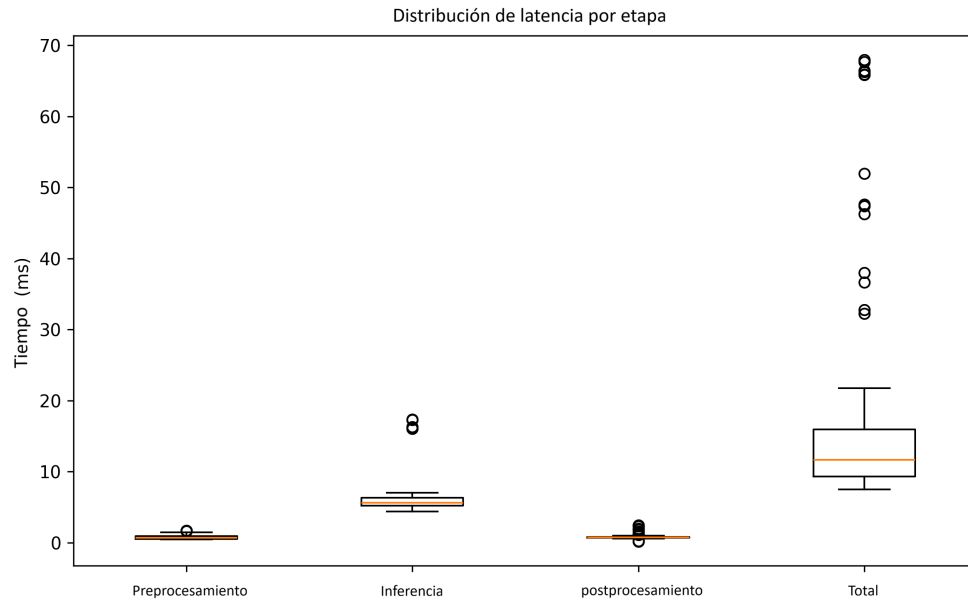


Figura 4.8: Distribución de latencia por etapa del *pipeline* de inferencia.

estándar de latencia mostrada en la Tabla 4.11, para contextualizar los tiempos de respuesta observados durante la evaluación del sistema, de acuerdo con los criterios establecidos en la literatura de interacción humano-computadora [1, 2].

Es importante mencionar que el sistema evaluado no mide video continuo sino latencia en imágenes individuales (tiempo de procesamiento). Para fines comparativos con la literatura de visión por computadora se reportó la métrica FPS (frames per second), que en ese contexto se calculó a partir del tiempo promedio de procesamiento por imagen.

Comparación del desempeño entre dispositivos

Con el objetivo de evaluar el rendimiento del sistema en diferentes condiciones de hardware, se realizaron pruebas de ejecución en dos dispositivos móviles con distintas capacidades computacionales: un dispositivo de gama básica y un dispositivo de gama media-alta, como se muestra en la Tabla 4.12.

La evaluación se llevó a cabo mediante la medición del tiempo de ejecución del *pipeline* completo del sistema, el cual incluye las etapas de preprocesamiento, inferencia del modelo de inteligencia artificial y postprocesamiento de resultados. Las mediciones se realizaron utilizando imágenes clínicas reales y registrando los tiempos de ejecución mediante herramientas de monitoreo del sistema en la plataforma Android Studio. La Tabla 4.12 presenta el tiempo promedio de



Tabla 4.10: Comparación entre la latencia del modelo en entorno de *benchmark* y la latencia del sistema completo en la aplicación móvil.

Métrica	Modelo (<i>benchmark</i>)	Sistema (<i>pipeline</i> real)
Preprocess time	–	0.766 ms
Inference time	7.02 ms	6.277 ms
Postprocess time	–	0.869 ms
Total latency	7.02 ms	17.061 ms
FPS	142 FPS	58.6 FPS

Tabla 4.11: Clasificación estándar de latencia en sistemas interactivos (adaptado de [1, 2]).

Clasificación de latencia	Tiempo de respuesta
Instantáneo	0.1 s
Interactivo	1 s
Atención del usuario	10 s
Procesamiento prolongado	>10 s

procesamiento por imagen y la velocidad de procesamiento del sistema en términos de imágenes por segundo para cada dispositivo móvil.

La relación de velocidad de procesamiento del sistema entre los dos dispositivos indica que el sistema es aproximadamente 7.4 veces más rápido en el dispositivo Samsung Galaxy S10e (gama media-alta) con respecto al dispositivo Samsung Galaxy A02s (gama básica).

Los resultados obtenidos permiten analizar el impacto del hardware en el desempeño del sistema y determinar la viabilidad de su uso en dispositivos móviles con diferentes niveles de capacidad computacional.

Estos resultados indican que el sistema puede ejecutarse en ambos tipos de dispositivos móviles evaluados, aunque el tiempo de procesamiento varía en función de las características del hardware disponible.

En el dispositivo de gama básica se observaron tiempos de respuesta mayores, mientras que



Tabla 4.12: Comparación del desempeño del sistema entre dispositivos móviles con diferentes capacidades computacionales.

Característica	Samsung Galaxy A02s	Samsung Galaxy S10e
Clasificación del dispositivo	Gama básica	Gama media-alta
Tiempo promedio por imagen (s)	27.1	3.68
Velocidad de procesamiento (FPS)	0.04	0.27

en el dispositivo de gama media-alta el procesamiento se realizó en menor tiempo, lo que evidencia el impacto directo de la capacidad computacional en el rendimiento del sistema.

4.3.3. Estabilidad del sistema

La estabilidad del sistema se evaluó mediante la ejecución repetida del *pipeline* de inferencia utilizando múltiples imágenes clínicas en dispositivos móviles reales. Durante las pruebas realizadas, el sistema mantuvo un comportamiento consistente, sin interrupciones ni fallos durante el procesamiento.

Los tiempos de ejecución registrados presentaron variaciones dentro de rangos esperados, lo que indica un funcionamiento estable del sistema bajo condiciones normales de operación.

Para la evaluación del desempeño del sistema se realizaron 15 ejecuciones independientes del *pipeline* de inferencia en cada dispositivo móvil evaluado. Se utilizaron 15 imágenes de prueba en ambos dispositivos, empleando las mismas imágenes en cada caso con el fin de mantener condiciones de evaluación equivalentes.

Los tiempos de procesamiento reportados corresponden al promedio obtenido a partir de estas mediciones, las cuales fueron registradas mediante herramientas de monitoreo del sistema durante la ejecución de la aplicación.

Los tiempos de ejecución utilizados en esta evaluación fueron obtenidos a partir de registros del sistema (*logcat*), los cuales se presentan en el Anexo Anexo F.



4.4. Discusión

Interpretación del *benchmark multi-seed* y análisis estadístico Los modelos basados en la familia U-Net obtuvieron resultados consistentes en las métricas de segmentación semántica, especialmente en Dice, IoU y precisión por clase. Attention U-Net presentó el mejor desempeño general en las métricas semánticas, mostrando una mejor correspondencia entre las regiones segmentadas y las anotaciones de referencia.

Por otra parte, YOLO11m-seg obtuvo los valores más altos de Recall tanto en las métricas globales como en las métricas por clase. En el contexto global, esto indica una mayor capacidad para detectar regiones lesionadas frente al fondo, reduciendo la cantidad de falsos negativos. Este comportamiento resulta importante en aplicaciones clínicas, donde omitir regiones de quemadura puede afectar la evaluación de la lesión.

Las matrices de confusión muestran un comportamiento consistente con las métricas obtenidas. En particular, YOLO11m-seg presentó menor confusión hacia fondo en comparación con las arquitecturas basadas en U-Net, especialmente para la clase de tercer grado, donde alcanzó la mayor proporción de píxeles correctamente clasificados. Esto explica los valores elevados de Recall observados durante la evaluación, indicando una mayor cobertura de las regiones lesionadas.

Aun así, aunque YOLO11m-seg logró detectar una mayor proporción de regiones lesionadas, también presentó menor precisión semántica en algunas zonas complejas, particularmente en bordes irregulares y regiones pequeñas. Esto se refleja en valores ligeramente menores de Dice e IoU respecto a Attention U-Net y U-Net. En contraste, las arquitecturas U-Net produjeron segmentaciones más precisas a nivel píxel, aunque con una tendencia ligeramente más conservadora al clasificar ciertas regiones como fondo.

Las pruebas estadísticas mediante Kruskal-Wallis detectaron diferencias significativas entre arquitecturas para Dice, IoU y precisión. En contraste, no se observaron diferencias estadísticamente significativas para Recall, lo que sugiere que todos los modelos mantienen una capacidad relativamente similar para recuperar regiones lesionadas.

Adicionalmente, las comparaciones post-hoc mediante la prueba de Dunn mostraron que las diferencias significativas se concentran principalmente en Dice y precisión entre YOLO11m-seg y las arquitecturas basadas en U-Net. Esto indica que las principales diferencias entre modelos se



relacionan con la calidad de la segmentación semántica fina y la precisión espacial de las regiones segmentadas, más que con la capacidad general de detectar áreas lesionadas.

Los resultados obtenidos son consistentes con trabajos previos basados en arquitecturas U-Net, donde las conexiones de salto y los mecanismos de atención permiten preservar información espacial y mejorar la precisión semántica en tareas de segmentación médica [22, 24]. En contraste, el comportamiento observado en YOLO11m-seg resulta consistente con trabajos previos basados en la familia YOLO, donde las arquitecturas orientadas a detección mostraron una alta capacidad para localizar regiones lesionadas y mantener inferencia eficiente en tiempo real [29–31].

Eficiencia computacional y despliegue móvil En términos de eficiencia computacional, las arquitecturas evaluadas presentaron diferencias importantes tanto en complejidad como en desempeño práctico durante la inferencia. U-Net obtuvo la menor latencia y el mayor FPS, lo que indica una ejecución más rápida bajo las condiciones evaluadas. Además, presentó el menor número de FLOPs entre los modelos comparados, manteniendo al mismo tiempo métricas competitivas de segmentación.

Por otra parte, U-Net++ presentó el mayor costo computacional, reflejado en el número de FLOPs, el consumo de memoria GPU y la latencia de inferencia. Aunque este modelo mantiene un desempeño competitivo en segmentación, el incremento en complejidad computacional resulta considerable respecto a las demás arquitecturas evaluadas.

Attention U-Net mostró el mejor desempeño global en métricas semánticas, aunque con un incremento en número de parámetros y consumo de memoria GPU respecto a U-Net convencional. Esto sugiere que los mecanismos de atención permiten mejorar la calidad de segmentación, particularmente en regiones complejas, a costa de un mayor costo computacional.

En el caso de YOLO11m-seg, el modelo presentó el menor número de parámetros y el menor consumo de VRAM entre las arquitecturas evaluadas, manteniendo una latencia cercana a la observada en Attention U-Net. Aunque sus métricas semánticas fueron ligeramente menores en comparación con las arquitecturas basadas en U-Net, el modelo mantuvo un desempeño competitivo considerando que integra simultáneamente detección, clasificación y segmentación de instancias en una sola arquitectura.

Las Figuras 4.3 y 4.4 muestran que no existe una relación lineal directa entre FLOPs,



latencia y desempeño de segmentación. Por ejemplo, YOLO11m-seg presentó menos parámetros y menor consumo de memoria que varias arquitecturas CNN, pero con una latencia mayor que U-Net. Esto puede relacionarse con diferencias en la estructura interna del modelo, operaciones utilizadas, manejo de memoria y etapas adicionales de postprocesamiento requeridas durante la segmentación por instancias.

Desde el punto de vista del despliegue móvil, YOLO11m-seg representa una alternativa favorable debido a que combina buen desempeño global, bajo consumo de memoria GPU y capacidades funcionales adicionales como detección, segmentación de instancias, cajas delimitadoras y niveles de confianza. Aunque U-Net presentó menor latencia en el entorno de escritorio, YOLO11m-seg ofreció un equilibrio más adecuado para el sistema móvil propuesto. Además, las pruebas de latencia realizadas en dispositivos móviles con diferentes capacidades computacionales demuestran que el sistema puede ejecutarse localmente en dispositivos Android, manteniendo la inferencia funcional sin necesidad de procesamiento en servidores externos.

A diferencia de varios trabajos previos orientados únicamente a segmentación o detección, el presente trabajo incorpora además evaluación de eficiencia computacional y despliegue móvil en Android, aspectos poco reportados en la literatura actual [12, 13].

Limitaciones del estudio Este estudio se basó en el entrenamiento y evaluación de modelos de aprendizaje profundo utilizando un conjunto de datos clínico propio. El principal inconveniente es que el tamaño del conjunto de datos continúa siendo relativamente reducido en comparación con grandes conjuntos utilizados en otras áreas de visión por computadora. Además, aunque las imágenes fueron obtenidas en un entorno clínico real, el conjunto de datos no representa completamente la variabilidad presente en distintos hospitales, dispositivos de captura, condiciones de iluminación, tonalidades de piel y tipos de lesión que pueden encontrarse en escenarios clínicos reales.

Otra limitación corresponde al proceso de anotación de las máscaras de referencia, ya que la delimitación de regiones de quemadura y la asignación de profundidad pueden presentar variabilidad entre especialistas, especialmente en zonas con transiciones visuales poco definidas entre distintos grados de severidad.

Asimismo, el estudio se enfocó únicamente en quemaduras de segundo y tercer grado, por



lo que no se evaluó el desempeño del sistema sobre otros tipos de lesiones cutáneas o patologías dermatológicas que podrían generar patrones visuales similares.

Por otra parte, aunque el sistema fue desplegado y evaluado en dispositivos móviles Android, las pruebas de latencia se realizaron sobre un número limitado de dispositivos y bajo condiciones controladas. Por ello, el desempeño puede variar en teléfonos con diferentes capacidades de hardware, versiones del sistema operativo o condiciones de ejecución distintas.

Finalmente, aunque las arquitecturas evaluadas mostraron resultados competitivos, el estudio se centró principalmente en modelos supervisados basados en CNN y YOLO. En trabajos futuros podrían explorarse enfoques más recientes basados en transformers, modelos fundacionales o estrategias de aprendizaje con pocos datos, con el objetivo de mejorar la capacidad de generalización en conjuntos clínicos reducidos.

Síntesis de hallazgos Los resultados obtenidos en este estudio son consistentes con la naturaleza de las arquitecturas evaluadas. Las arquitecturas de la familia U-Net corresponden a modelos de segmentación semántica orientados a la clasificación precisa de cada píxel, mientras que YOLO11m-seg pertenece a un enfoque de segmentación de instancias que integra detección, clasificación y segmentación dentro de una misma arquitectura. Por tanto, las diferencias observadas entre métricas no deben interpretarse como resultados contradictorios, sino como evaluaciones complementarias del comportamiento de cada modelo.

Las métricas globales permitieron evaluar la capacidad general para localizar regiones lesionadas frente al fondo, mientras que las métricas por clase analizaron la asignación semántica específica entre quemaduras de segundo y tercer grado. Por otra parte, las matrices de confusión permitieron identificar si los errores se relacionaban principalmente con omisión de regiones hacia fondo o con confusión entre distintos grados de severidad.

Los resultados mostraron que las arquitecturas basadas en U-Net obtuvieron una mejor precisión semántica y mayor consistencia en las métricas de segmentación a nivel píxel. En contraste, YOLO11m-seg presentó los valores más altos de Recall global y por clase, así como menor confusión hacia fondo, indicando una mayor capacidad para cubrir regiones lesionadas y reducir falsos negativos.

En el contexto clínico de las quemaduras, omitir regiones lesionadas puede representar una



limitación más importante que pequeñas pérdidas de precisión semántica en bordes o regiones complejas. Bajo esta perspectiva, el comportamiento observado en YOLO11m-seg puede resultar útil como herramienta de apoyo inicial para triaje clínico y evaluación visual asistida en dispositivos móviles, especialmente en escenarios con recursos computacionales limitados.

Capítulo 5: Conclusiones

En este trabajo se desarrolló e implementó un sistema basado en aprendizaje profundo para la segmentación y clasificación automática de quemaduras de segundo y tercer grado en imágenes clínicas, considerando además su integración en dispositivos móviles Android mediante TensorFlow Lite.

Los resultados obtenidos mostraron que la arquitectura Attention U-Net permitió obtener el mejor desempeño global promedio durante el *benchmark multi-seed*, alcanzando los mayores valores de Dice, IoU y Precisión bajo el protocolo experimental utilizado. Por otro lado el modelo YOLO11m-seg mostró el mejor resultado en la métrica Recall. Asimismo, el modelo presentó un menor consumo de memoria GPU y tiempos de inferencia competitivos, lo cual favorece su utilización en escenarios de despliegue móvil.

El análisis estadístico mediante la prueba de Kruskal-Wallis mostró diferencias estadísticamente significativas entre las arquitecturas evaluadas para las métricas Dice, IoU y precisión global. Posteriormente, las comparaciones post-hoc mediante la prueba de Dunn con corrección de Bonferroni identificaron diferencias significativas principalmente entre YOLO11m-seg y las arquitecturas basadas en U-Net en las métricas Dice y precisión. En contraste, no se observaron diferencias estadísticamente significativas para Recall. Estos resultados sugieren que las principales diferencias entre modelos se relacionan con la calidad de la segmentación semántica y la precisión espacial de las regiones segmentadas, más que con la capacidad general de detectar regiones lesionadas.

Por otra parte, las arquitecturas evaluadas mostraron similitudes en las métricas de segmentación por clase, con una ligera superioridad en la evaluación del modelo Attention U-Net. Además todas las arquitecturas mostraron más dificultades en la clase de segundo grado. Por otro lado, el modelo YOLO11m-seg presentó los mayores valores en Recall en la dos clases evaluadas.

Los resultados obtenidos evidencian que las métricas globales y las métricas por clase representan evaluaciones complementarias y no equivalentes. Mientras las métricas globales reflejan la capacidad de cobertura espacial y detección general de regiones lesionadas, las métricas por clase



permiten evaluar con mayor sensibilidad la preservación de la asignación semántica específica entre tipos de quemadura.

En términos funcionales, YOLO11m-seg fue la única arquitectura capaz de integrar simultáneamente segmentación, clasificación, generación de bounding boxes y niveles de confianza dentro de una sola inferencia, facilitando su integración en sistemas móviles clínicos. La conversión e implementación del modelo mediante TensorFlow Lite permitió realizar inferencia local sobre dispositivos Android reales sin necesidad de procesamiento remoto.

Los resultados de latencia en dispositivos móviles mostraron diferencias importantes entre las capacidades computacionales. Sin embargo, el sistema mantuvo la capacidad de realizar inferencia local en ambos escenarios evaluados, demostrando la viabilidad del despliegue móvil de modelos basados en segmentación de instancias para apoyo clínico en el análisis automático de quemaduras.

Finalmente, este trabajo demuestra que los modelos basados en segmentación de instancias pueden representar una alternativa eficiente para aplicaciones móviles de apoyo clínico, al ofrecer un equilibrio favorable entre desempeño global, eficiencia computacional y capacidades funcionales de integración.

Capítulo 6: Trabajo futuro

Como trabajo futuro, se propone ampliar el conjunto de datos clínicos utilizado en este estudio, incorporando imágenes obtenidas en distintos entornos hospitalarios y bajo diferentes condiciones de adquisición. Esto permitiría incrementar la diversidad visual de las lesiones y mejorar la capacidad de generalización del sistema ante variaciones de iluminación, tono de piel, calidad de captura y dispositivos móviles utilizados durante la adquisición de las imágenes clínicas.

Además, este trabajo puede extenderse hacia enfoques basados en *few-shot learning* y *meta-learning* orientados al dominio médico, donde los modelos puedan adaptarse a nuevos dominios o tareas utilizando únicamente unos pocos ejemplos anotados [74]. Este tipo de estrategias resulta especialmente relevante en aplicaciones clínicas, donde la obtención de grandes conjuntos de datos anotados suele ser limitada.

Por otra parte, futuras líneas de investigación podrían incluir el uso de modelos fundacionales y técnicas de adaptación de dominio para mejorar la robustez del sistema en escenarios clínicos reales y diferentes ambientes hospitalarios.

Finalmente, se propone realizar estudios clínicos con profesionales de la salud para evaluar el impacto del sistema como herramienta de apoyo visual en escenarios reales de diagnóstico y telemedicina, así como analizar su usabilidad y desempeño en dispositivos móviles utilizados en práctica clínica.

ANEXOS

Anexo A. Curvas de entrenamiento del modelo YOLOm11-seg

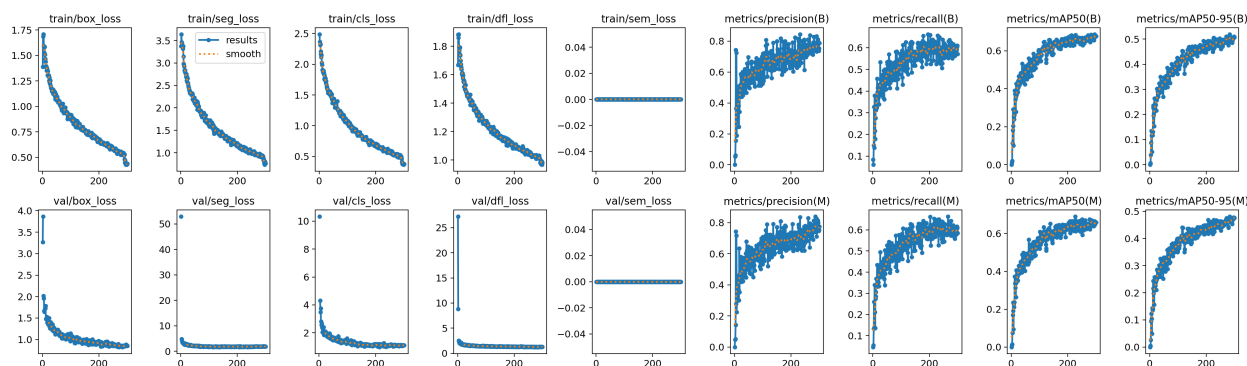


Figura A.1: Curvas de entrenamiento y validación obtenidas durante el entrenamiento del modelo YOLOv11m-seg seleccionado como configuración final en la experimentación de distintas configuraciones.

Anexo B. Resultados cualitativos adicionales de YOLO11m-seg

Además de la evaluación cuantitativa, se realizó un análisis visual de las predicciones generadas por el modelo sobre imágenes clínicas no utilizadas durante el entrenamiento.

Se generaron visualizaciones tipo *triptych*, *overlays* y mapas de error, donde se muestran la imagen original, la máscara *ground truth* y la predicción obtenida por el sistema.

Las Figuras B.1 y B.2 presentan ejemplos cualitativos de las segmentaciones generadas por el modelo YOLO11m-seg sobre imágenes clínicas reales del conjunto de prueba.

En la mayoría de los casos evaluados, el modelo logra segmentar adecuadamente regiones extensas de quemaduras, manteniendo una buena correspondencia espacial con las anotaciones de referencia. Sin embargo, también se observaron algunos casos complejos donde existen discrepancias localizadas principalmente en bordes irregulares, regiones pequeñas y zonas con diferencias de textura poco definidas, particularmente en quemaduras de segundo grado.

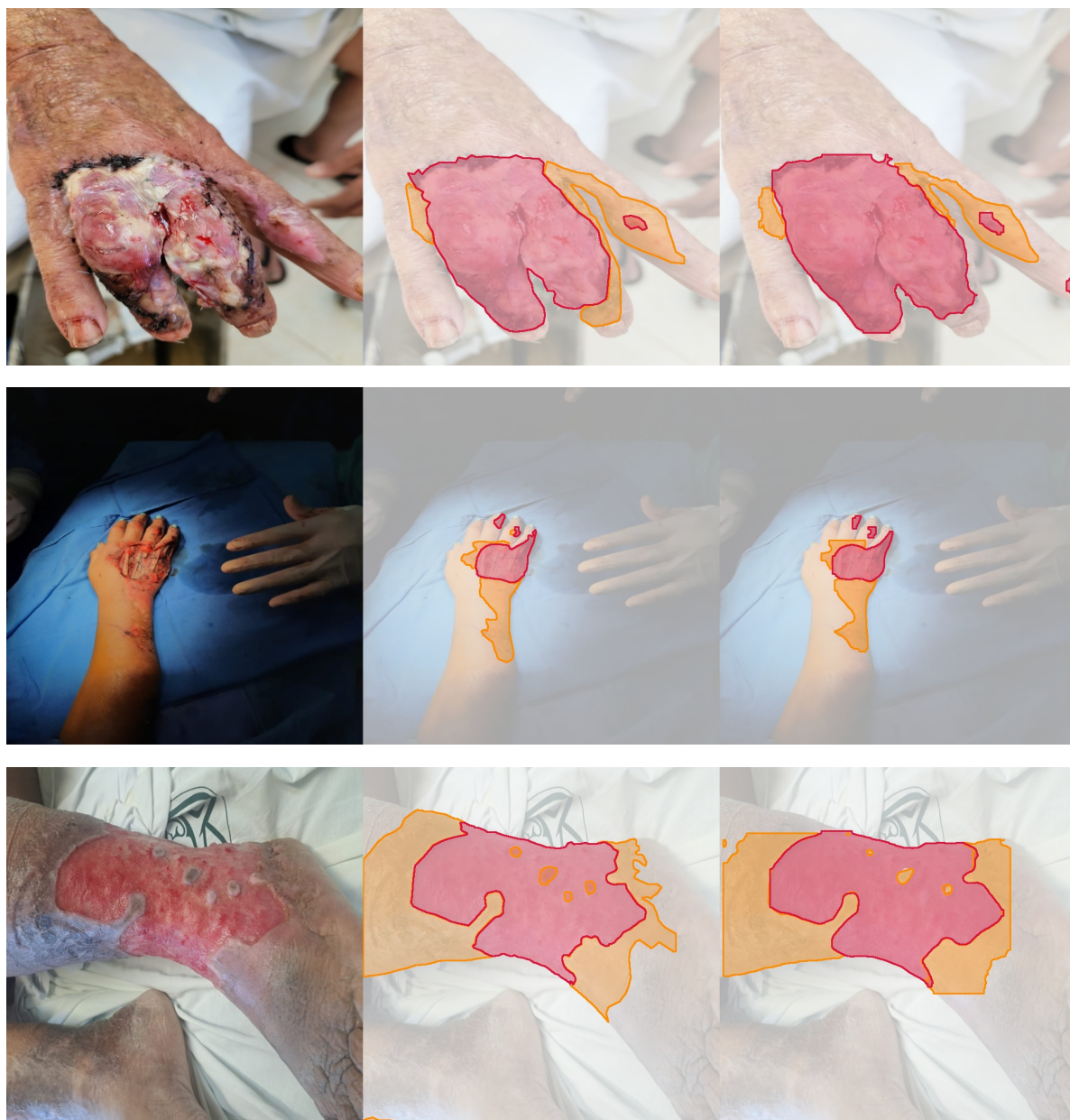


Figura B.1: Resultados cualitativos obtenidos por el modelo YOLO11m-seg sobre imágenes clínicas del conjunto de prueba. Cada ejemplo muestra la imagen original, la anotación *ground truth* y la predicción generada por el sistema. Las regiones segmentadas corresponden a quemaduras de segundo grado (naranja) y tercer grado (rojo).



Figura B.2: Continuación de los resultados cualitativos obtenidos por el modelo YOLO11m-seg sobre imágenes clínicas del conjunto de prueba.



La Figura B.3 muestra ejemplos representativos de mapas de error obtenidos durante la evaluación del modelo. Se incluyen casos con errores mínimos, discrepancias moderadas y ejemplos más complejos donde existen diferencias visibles entre la predicción y las anotaciones de referencia.

Los principales errores observados corresponden a regiones pequeñas, bordes complejos y zonas con transiciones de textura poco definidas, particularmente en quemaduras de segundo grado. Aun así, el modelo mantiene una adecuada capacidad para detectar regiones extensas de lesión sobre imágenes clínicas reales.

Anexo C. Backlog de la aplicación móvil

C.1. Sprints del sistema

Tabla C.1: Distribución de historias de usuario por sprints durante el desarrollo de la aplicación móvil

Sprint	Historias de usuario	Objetivo del sprint
Sprint 1	HU-01 – HU-09	Establecer la base funcional de la aplicación móvil, incluyendo la estructura inicial del proyecto, la interfaz principal, la navegación entre pantallas y la captura de imágenes clínicas.
Sprint 2	HU-10 – HU-14, HU-19 – HU-22	Implementar la gestión de pacientes y el almacenamiento persistente de la información clínica mediante una base de datos local, permitiendo registrar, consultar y administrar casos clínicos.
Sprint 3	HU-15 – HU-18, HU-25	Desarrollar la lógica de evaluación clínica mediante variables médicas y reglas de decisión para generar recomendaciones clínicas automáticas basadas en la severidad de la quemadura.



Sprint	Historias de usuario	Objetivo del sprint
Sprint 4	HU-26 – HU-29	Integrar el modelo de inteligencia artificial en la aplicación móvil, incluyendo la carga del modelo, el preprocesamiento de la imagen y la ejecución de la inferencia automática.
Sprint 5	HU-30 – HU-31, HU-23 – HU-24	Visualizar los resultados generados por el sistema, integrar las recomendaciones clínicas y permitir la generación y compartición de reportes clínicos.

C.2. Backlog de historias de usuario de la aplicación móvil

La Tabla C.2 presenta el conjunto completo de historias de usuario definidas para el desarrollo de la aplicación móvil.

Tabla C.2: Backlog de historias de usuario de la aplicación móvil

ID	Descripción	Criterios de aceptación
HU-01	Configurar la estructura inicial del proyecto en Android Studio, incluyendo la creación del proyecto, la definición de la arquitectura de la aplicación y la organización de los componentes principales.	■ El proyecto se crea correctamente en Android Studio ■ La estructura de carpetas del proyecto está definida ■ Las dependencias necesarias están configuradas ■ El entorno de desarrollo está listo para implementar funcionalidades



ID	Descripción	Criterios de aceptación
HU-02	Diseñar la interfaz de usuario para la pantalla de recomendaciones clínicas, permitiendo capturar variables clínicas relevantes y mostrar resultados de evaluación.	■ La pantalla muestra los campos necesarios para la evaluación clínica ■ Los elementos de la interfaz son visibles y funcionales ■ El usuario puede ingresar información clínica manualmente ■ La interfaz permite continuar con el proceso de evaluación
HU-03	Definir la entidad PatientCase en la base de datos para almacenar la información clínica generada durante la evaluación.	■ La entidad PatientCase se crea correctamente en la base de datos ■ La estructura incluye los campos definidos para el caso clínico ■ Los datos pueden almacenarse y recuperarse sin errores ■ La entidad es accesible desde el repositorio de datos



ID	Descripción	Criterios de aceptación
HU-04	Configurar la base de datos local utilizando Room Database para permitir el almacenamiento persistente de información de pacientes.	■ La base de datos se crea correctamente al iniciar la aplicación ■ La aplicación puede conectarse a la base de datos sin errores ■ La estructura de la base de datos se mantiene entre sesiones ■ No se pierde información al reiniciar la aplicación
HU-05	Diseñar la interfaz principal de la aplicación, incluyendo elementos visuales como logotipo, botones y mensajes informativos.	■ La pantalla principal se muestra correctamente ■ Los elementos visuales son visibles y funcionales ■ La interfaz mantiene consistencia visual ■ La aplicación inicia sin errores



ID	Descripción	Criterios de aceptación
HU-06	Implementar la navegación entre las pantallas principales de la aplicación mediante botones y menús.	■ El usuario puede navegar entre las pantallas principales ■ La navegación se realiza sin errores ■ La aplicación mantiene el estado de la interfaz ■ Las pantallas se cargan correctamente
HU-07	Mostrar un mensaje informativo indicando que el sistema no sustituye la evaluación médica profesional.	■ El mensaje se muestra en la pantalla de resultados ■ El texto es visible y legible ■ El mensaje no bloquea la navegación del usuario
HU-08	Permitir al usuario capturar una imagen clínica utilizando la cámara del dispositivo o seleccionar una imagen desde la galería.	■ El usuario puede abrir la cámara desde la aplicación ■ El usuario puede seleccionar una imagen desde la galería ■ La imagen seleccionada se carga correctamente en la aplicación ■ La aplicación no presenta errores durante la captura o selección



ID	Descripción	Criterios de aceptación
HU-09	Mostrar la imagen seleccionada o capturada en la pantalla principal para permitir su visualización previa antes del procesamiento.	■ La imagen se muestra correctamente en la pantalla ■ La imagen mantiene su proporción original ■ El usuario puede continuar con el flujo de evaluación ■ No se presentan errores durante la visualización
HU-10	Definir la entidad Patient en la base de datos para representar la información básica de cada paciente.	■ La entidad Patient contiene los campos definidos (código, nombre, edad, sexo, notas) ■ La estructura de la tabla se crea correctamente en la base de datos ■ Los datos pueden almacenarse en la entidad sin errores ■ La entidad es accesible desde el repositorio de datos



ID	Descripción	Criterios de aceptación
HU-11	Permitir al usuario registrar nuevos pacientes mediante un formulario dentro de la aplicación.	■ El usuario puede ingresar los datos del paciente ■ El sistema valida los campos obligatorios ■ El paciente se guarda correctamente en la base de datos ■ Se muestra un mensaje de confirmación al guardar
HU-12	Mostrar la lista de pacientes registrados en la base de datos mediante una interfaz visual.	■ Los pacientes registrados se muestran en una lista ■ La información se carga correctamente desde la base de datos ■ La lista se actualiza después de agregar un paciente ■ No se presentan errores durante la visualización



ID	Descripción	Criterios de aceptación
HU-13	Permitir eliminar un paciente registrado en la base de datos desde la lista de pacientes.	■ El usuario puede seleccionar un paciente para eliminar ■ El sistema solicita confirmación antes de eliminar ■ El paciente se elimina correctamente de la base de datos ■ La lista se actualiza después de la eliminación
HU-14	Permitir seleccionar un paciente existente dentro del flujo de recomendaciones clínicas para asociar la evaluación con un registro específico.	■ El usuario puede seleccionar un paciente desde un listado ■ La información del paciente se carga automáticamente ■ La evaluación se asocia correctamente al paciente seleccionado ■ La selección de paciente es opcional



ID	Descripción	Criterios de aceptación
HU-15	Permitir al usuario ingresar el porcentaje de superficie corporal afectada (TBSA) como variable clínica para la evaluación de la severidad de la quemadura	■ El usuario puede ingresar un valor numérico de TBSA ■ El sistema valida que el valor esté dentro de un rango permitido ■ El valor se almacena correctamente en la evaluación ■ El campo no permite valores inválidos
HU-16	Permitir seleccionar factores clínicos que influyen en la gravedad de la quemadura, como inhalación, exposición química, lesión eléctrica o zonas críticas.	■ El usuario puede seleccionar uno o más factores clínicos ■ Las opciones se muestran correctamente en la interfaz ■ Los factores seleccionados se registran en la evaluación ■ El sistema permite continuar sin errores



ID	Descripción	Criterios de aceptación
HU-17	Desarrollar la lógica de evaluación clínica que clasifica la severidad de la quemadura en diferentes niveles de prioridad según las reglas definidas.	■ El sistema evalúa los datos clínicos ingresados ■ Se determina un nivel de prioridad clínica ■ La clasificación se realiza de acuerdo con las reglas definidas ■ El resultado se genera sin errores
HU-18	Generar una recomendación clínica basada en el nivel de prioridad determinado por el sistema de evaluación.	■ El sistema muestra el nivel de prioridad clínica ■ Se genera un mensaje de recomendación ■ El mensaje es visible y comprensible para el usuario ■ La recomendación se actualiza cuando cambian los datos



ID	Descripción	Criterios de aceptación
HU-19	Permitir almacenar la información clínica generada durante la evaluación en la base de datos local.	■ El sistema guarda correctamente los datos del caso clínico ■ La información se almacena sin pérdida de datos ■ El registro se mantiene después de reiniciar la aplicación ■ Se muestra confirmación al guardar el caso
HU-20	Mostrar el historial completo de evaluaciones clínicas almacenadas en la base de datos.	■ El sistema muestra una lista de casos clínicos registrados ■ La información se carga correctamente desde la base de datos ■ La lista se actualiza cuando se guarda un nuevo caso ■ No se presentan errores durante la visualización



ID	Descripción	Criterios de aceptación
HU-21	Mostrar los detalles completos de un caso clínico seleccionado desde el historial.	■ El sistema muestra la información completa del caso ■ Se visualizan correctamente los datos clínicos ■ El usuario puede acceder al detalle desde el historial ■ No se presentan errores durante la visualización
HU-22	Guardar la imagen original y la imagen procesada como parte del registro clínico.	■ El sistema guarda correctamente la imagen original ■ El sistema guarda correctamente la imagen procesada ■ Las rutas de las imágenes se almacenan en la base de datos ■ Las imágenes pueden recuperarse posteriormente



ID	Descripción	Criterios de aceptación
HU-23	Crear un documento en formato PDF que contenga la información clínica y las imágenes del caso evaluado.	■ El sistema genera un archivo PDF correctamente ■ El documento contiene los datos clínicos del caso ■ El documento incluye las imágenes correspondientes ■ El archivo se guarda en el dispositivo
HU-24	Permitir compartir el reporte clínico generado mediante aplicaciones disponibles en el dispositivo.	■ El usuario puede seleccionar una aplicación para compartir ■ El archivo se envía correctamente ■ El sistema no presenta errores durante el envío
HU-25	Mostrar información de ayuda y orientación sobre el uso del sistema y sus limitaciones.	■ El usuario puede acceder al módulo de ayuda ■ La información se muestra correctamente ■ El contenido es legible y comprensible



ID	Descripción	Criterios de aceptación
HU-26	Cargar el modelo de inteligencia artificial en la aplicación móvil utilizando TensorFlow Lite para permitir su ejecución local en el dispositivo.	■ El modelo se carga correctamente al iniciar la aplicación ■ El archivo del modelo es accesible desde la aplicación ■ No se presentan errores durante la carga del modelo
HU-27	Transformar la imagen capturada en el formato requerido por el modelo de inteligencia artificial mediante redimensionamiento, normalización y conversión a tensor.	■ La imagen se redimensiona al tamaño requerido por el modelo ■ Los valores de la imagen se normalizan correctamente ■ La imagen se convierte en un tensor válido
HU-28	Ejecutar el modelo de inteligencia artificial para analizar la imagen clínica y generar resultados de detección y segmentación.	■ El modelo se ejecuta correctamente ■ Se generan resultados de inferencia ■ El sistema no presenta fallos durante la ejecución ■ El tiempo de respuesta se registra correctamente



ID	Descripción	Criterios de aceptación
HU-29	Procesar los resultados generados por el modelo para interpretar las detecciones y preparar la información para su visualización.	■ El sistema interpreta correctamente las salidas del modelo ■ Se identifican las clases detectadas ■ Los resultados se preparan para su visualización ■ El proceso se ejecuta sin errores
HU-30	Mostrar las detecciones y segmentaciones generadas por el modelo sobre la imagen clínica para facilitar la interpretación visual de los resultados.	■ Las detecciones se muestran sobre la imagen ■ Las segmentaciones se renderizan correctamente ■ La imagen procesada es visible en pantalla ■ No se presentan errores durante la visualización
HU-31	Enviar los resultados generados por el modelo de inteligencia artificial al sistema de recomendaciones clínicas para su evaluación y almacenamiento.	■ Los resultados del modelo se envían correctamente al sistema clínico ■ La información se integra con las recomendaciones ■ El sistema permite continuar con el flujo clínico



Figura B.3: Mapas de error generados por el modelo YOLO11m-seg sobre imágenes del conjunto de prueba. Las regiones coloreadas representan discrepancias entre la segmentación predicha y la anotación *ground truth*, incluyendo falsos positivos (rojo), falsos negativos (amarillo) y errores de clasificación



Anexo D. Arquitectura en capas de la aplicación móvil

La Figura D.1 presenta la arquitectura en capas de la aplicación móvil.

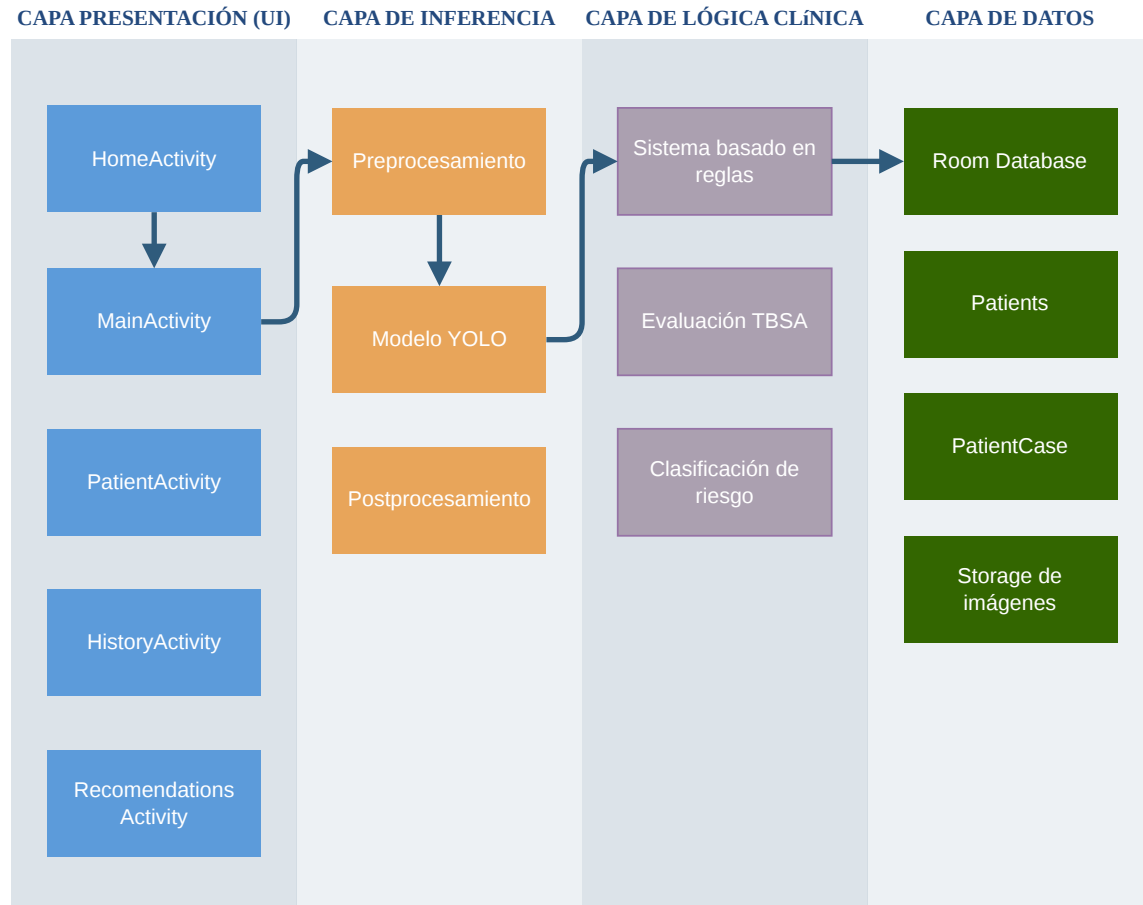


Figura D.1: Arquitectura en capas de la aplicación móvil

Anexo E. Visualización en video de la aplicación móvil

Enlace demo: <https://drive.google.com/file/d/1D-XuuopoCSqKyJdm3jXhrOFFPoTMyxhOm/view>



Anexo F. Registros de ejecución del sistema de evaluación de dispositivos móviles (Logcat)

Los siguientes registros corresponden a los tiempos de ejecución del *pipeline* de inferencia del sistema durante la evaluación experimental en dispositivos móviles reales.

Cada registro incluye el tiempo de procesamiento de las etapas de preprocesamiento, inferencia del modelo y postprocesamiento, así como el tiempo total de ejecución por imagen. Se presentan registros representativos obtenidos durante múltiples ejecuciones del sistema.

Dispositivo Samsung Galaxy A02s (gama básica)

03-25 11:49:14.270 11696 12685 D TIMING : image=1000003382, preprocess=1308.8675 ms, inference=24153.315095 ms, postprocess=535.00349 ms, total=25997.20197 ms 03-25 11:50:46.792 11696 12685 D TIMING : image=1000003374, preprocess=1119.545364 ms, inference=24140.528689 ms, postprocess=637.388749 ms, total=25897.479157 ms 03-25 11:52:59.504 11696 13113 D TIMING : image=1000003373, preprocess=1111.122187 ms, inference=23892.837856 ms, postprocess=437.119115 ms, total=25441.091448 ms 03-25 11:54:24.054 11696 13196 D TIMING : image=1000003376, preprocess=1132.973437 ms, inference=24206.917127 ms, postprocess=196.927812 ms, total=25536.832178 ms 03-25 11:55:32.183 11696 13313 D TIMING : image=1000003387, preprocess=1149.039531 ms, inference=23707.67723 ms, postprocess=1115.107135 ms, total=25971.840875 ms 03-25 12:40:28.761 15627 15894 D TIMING : image=1000003375, preprocess=1301.681978 ms, inference=23215.08572 ms, postprocess=1142.71 ms, total=25659.502126 ms 03-25 12:44:01.010 15627 16074 D TIMING : image=1000003375, preprocess=1109.805104 ms, inference=22668.650252 ms, postprocess=493.694999 ms, total=24272.164678 ms 03-25 13:07:23.646 19030 19160 D TIMING : image=1000003373, preprocess=1334.083437 ms, inference=23044.65244 ms, postprocess=1549.475937 ms, total=25928.231396 ms 03-25 13:31:08.438 20693 20904 D TIMING : image=1000003373, preprocess=1340.33026 ms, inference=23028.464314 ms, postprocess=13494.126193 ms, total=37862.93686 ms 03-25 13:33:01.750 20693 21041 D TIMING : image=1000003375, preprocess=1102.06401 ms, inference=22573.096918 ms, post-



process=11885.441193 ms, total=35560.616705 ms 03-25 13:36:19.780 21116 21204 D TIMING : image=1000003375, preprocess=1350.173957 ms, inference=22713.788533 ms, postprocess=1529.692343 ms, total=25593.671188 ms 03-25 13:53:41.960 22864 23041 D TIMING : image=1000003388, preprocess=1339.130832 ms, inference=22952.459523 ms, postprocess=1343.073646 ms, total=25634.682177 ms 03-26 08:24:26.477 30257 30464 D TIMING : image=1000003388, preprocess=1344.884948 ms, inference=23771.976033 ms, postprocess=1355.976771 ms, total=26472.852906 ms 03-26 10:40:27.786 7403 7602 D TIMING : image=1000003377, preprocess=1340.498072 ms, inference=22885.241606 ms, postprocess=3034.054582 ms, total=27259.811343 ms 03-26 10:41:37.467 7403 7675 D TIMING : image=1000003381, preprocess=1109.12526 ms, inference=22200.596189 ms, postprocess=833.085312 ms, total=24142.818689 ms

Dispositivo Samsung Galaxy S10e (gama media-alta)

2026-03-26 08:58:13.950 30333-32279 TIMING com.example.quemadurashg.dev D image=image:782491, preprocess=235.297916 ms, inference=3206.896613 ms, postprocess=58.901093 ms, total=3501.09953 ms 2026-03-26 08:58:35.880 30333-32361 TIMING com.example.quemadurashg.dev D image=image:782490, preprocess=192.893646 ms, inference=3196.862707 ms, postprocess=96.077865 ms, total=3485.83802 ms 2026-03-26 08:58:53.473 30333-32427 TIMING com.example.quemadurashg.dev D image=image:782495, preprocess=196.552395 ms, inference=3241.871561 ms, postprocess=307.24224 ms, total=3745.670103 ms 2026-03-26 08:59:18.057 30333-32513 TIMING com.example.quemadurashg.dev D image=image:782494, preprocess=193.544114 ms, inference=3231.959582 ms, postprocess=442.215469 ms, total=3867.723124 ms 2026-03-26 08:59:39.826 30333-32593 TIMING com.example.quemadurashg.dev D image=image:782493, preprocess=192.482188 ms, inference=3291.169843 ms, postprocess=33.140521 ms, total=3516.796509 ms 2026-03-26 08:59:59.606 30333-32668 TIMING com.example.quemadurashg.dev D image=image:782492, preprocess=192.316719 ms, inference=3281.705988 ms, postprocess=91.971875 ms, total=3565.998436 ms 2026-03-26 09:00:14.684 30333-32741 TIMING com.example.quemadurashg.dev D image=image:782499, preprocess=191.365833 ms, inference=3370.680051 ms, postprocess=207.130886 ms, total=3769.180571 ms 2026-03-26 09:00:36.182 30333-1009 TI-



MING com.example.quemadurashg.dev D image=image:782498, preprocess=191.115833 ms, inference=3306.211301 ms, postprocess=19.85401 ms, total=3517.185415 ms 2026-03-26 09:00:57.965 30333-1120 TIMING com.example.quemadurashg.dev D image=image:782497, preprocess=196.233229 ms, inference=3302.772238 ms, postprocess=160.432031 ms, total=3659.442864 ms 2026-03-26 09:01:15.517 30333-1361 TIMING com.example.quemadurashg.dev D image=image:782496, preprocess=194.323907 ms, inference=3370.841665 ms, postprocess=302.294479 ms, total=3867.464269 ms 2026-03-26 09:01:34.198 30333-1621 TIMING com.example.quemadurashg.dev D image=image:782502, preprocess=193.39375 ms, inference=3383.805884 ms, postprocess=207.50125 ms, total=3784.704634 ms 2026-03-26 09:01:58.152 30333-2024 TIMING com.example.quemadurashg.dev D image=image:782500, preprocess=193.151407 ms, inference=3472.686457 ms, postprocess=207.855 ms, total=3873.697186 ms 2026-03-26 09:02:19.589 30333-2141 TIMING com.example.quemadurashg.dev D image=image:782504, preprocess=190.918333 ms, inference=3403.756614 ms, postprocess=4.189948 ms, total=3598.868749 ms 2026-03-26 09:02:44.074 30333-2442 TIMING com.example.quemadurashg.dev D image=image:782503, preprocess=192.669948 ms, inference=3460.11828 ms, postprocess=229.481875 ms, total=3882.274582 ms 2026-03-26 08:57:43.383 30333-32187 TIMING com.example.quemadurashg.dev D image=image:782503, preprocess=215.206771 ms, inference=3183.57604 ms, postprocess=224.623802 ms, total=3623.411145 ms

Bibliografía

- [1] J. Nielsen, *Usability Engineering*. Academic Press, 1993.
- [2] B. Shneiderman, C. Plaisant, M. Cohen, and S. Jacobs, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th ed. Pearson, 2016.
- [3] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015, pp. 234–241.
- [4] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A nested U-Net architecture for medical image segmentation,” in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA)*. Springer, 2018, pp. 3–11.
- [5] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, “Attention U-Net: Learning where to look for the pancreas,” *arXiv preprint arXiv:1804.03999*, 2018.
- [6] R. Khanam and M. Hussain, “Yolov11: An overview of the key architectural enhancements,” 2024, arXiv preprint.
- [7] S. A. Suha and T. F. Sanam, “A deep convolutional neural network-based approach for detecting burn severity from skin burn images,” *Machine Learning with Applications*, vol. 9, p. 100371, Sep. 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666827022000639>
- [8] World Health Organization, “Burns,” Fact sheet, Oct. 2023, accessed: Feb. 10, 2026. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/burns>
- [9] N. A. Rendón-Mejía, L. F. Cuervo-Ollervides, N. Flores-González, and C. A. Hernández-Terrazas, “Asociación de supervivencia y mortalidad con aspectos demográficos en grandes quemados en un centro de referencia de México 2022-2023,” *Cirugía Plástica*



- Ibero-Latinoamericana*, vol. 50, no. 3, pp. 333–340, Sep. 2024, publisher: Sociedad Española de Cirugía Plástica, Reparadora y Estética (SECPRE). [Online]. Available: https://scielo.isciii.es/scielo.php?script=sci_abstract&pid=S0376-78922024000300013&lng=es&nrm=iso&tlng=es
- [10] Y. Chang, Y. Wang, H. Wang, and Y. Xie, “Deep learning-based burn depth diagnosis: A review,” *Burns & Trauma*, vol. 9, p. tkab028, 2021.
- [11] H. Liu, K. Yue, S. Cheng, W. Li, and Z. Fu, “A framework for automatic burn image segmentation and burn depth diagnosis using deep learning,” *Computational and Mathematical Methods in Medicine*, vol. 2021, no. 1, p. 5514224, 2021. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/5514224>
- [12] L. Díaz-Granados-Zúñiga *et al.*, “Burnaid: a mobile application to support initial burn management in low-resource settings,” *International Journal of Burns and Trauma*, 2024, aplicación móvil para guía del manejo inicial de quemaduras.
- [13] C. W. Chang, H. Wang, F. Lai, M. Christian, S. Chen Huang, and H. Yi Tsai, “Comparison of 3d and 2d area measurement of acute burn wounds with lidar technique and deep learning model,” *Frontiers in Artificial Intelligence*, vol. Volume 8 - 2025, 2025. [Online]. Available: <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1510905>
- [14] M. G. Jeschke, M. E. van Baar, M. A. Choudhry, K. K. Chung, N. S. Gibran, and S. Logsetty, “Burn injury,” *Nature Reviews Disease Primers*, vol. 6, no. 1, p. 11, Feb. 2020. [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/32054846/>
- [15] M. D. Wilson *et al.*, “Review of machine learning for optical imaging of burn wound severity assessment,” *Journal of Biomedical Optics*, vol. 29, no. 2, p. 021203, 2024. [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10869118/>
- [16] H. Liu, K. Yue, S. Cheng, W. Li, and Z. Fu, “A framework for automatic burn image segmentation and burn depth diagnosis using deep learning,” *Computational and Mathematical Methods in Medicine*, vol. 2021, no. 1, p. 5514224, 2021. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/5514224>



- [17] P. K. Rangaiah, B. P. kumar, and R. Augustine, “Improving burn diagnosis in medical image retrieval from grafting burn samples using b-coefficients and the clahe algorithm,” *Biomedical Signal Processing and Control*, vol. 99, p. 106814, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1746809424008723>
- [18] G. Litjens, T. Kooi, B. Ehteshami Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, “A survey on deep learning in medical image analysis,” *Medical Image Analysis*, vol. 42, pp. 60–88, 2017.
- [19] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, “Transfusion: Understanding transfer learning for medical imaging,” *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: <https://arxiv.org/abs/1902.07208>
- [20] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.
- [21] 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.
- [22] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015, pp. 234–241.
- [23] Z. Cai *et al.*, “GL-FusionNet: Fusing global and local features to classify deep and superficial partial thickness burn,” *Mathematical Biosciences and Engineering*, vol. 20, no. 6, pp. 10 153–10 173, 2023.
- [24] O. Oktay, J. Schlemper, L. Le Folgoc *et al.*, “Attention u-net: Learning where to look for the pancreas,” in *Medical Imaging with Deep Learning (MIDL)*, 2018.
- [25] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A nested U-Net architecture for medical image segmentation,” in *Deep Learning in Medical Image Analysis*



and *Multimodal Learning for Clinical Decision Support*, ser. Lecture Notes in Computer Science, vol. 11045. Springer, 2018, pp. 3–11.

- [26] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, “TransUNet: Transformers make strong encoders for medical image segmentation,” *arXiv preprint arXiv:2102.04306*, 2021. [Online]. Available: <https://arxiv.org/abs/2102.04306>
- [27] C. W. Chang, F. Lai, M. Christian, Y. C. Chen, C. Hsu, Y. S. Chen, D. H. Chang, T. L. Roan, and Y. C. Yu, “Deep learning–assisted burn wound diagnosis: Diagnostic model development study,” *JMIR Medical Informatics*, vol. 9, no. 12, p. e22798, December 2021. [Online]. Available: <https://doi.org/10.2196/22798>
- [28] X. Li, Z. Liu, and L. Liu, “Pediatric burnnet: Robust multi-class segmentation and severity recognition under real-world imaging conditions,” *Sage Open Medicine*, vol. 13, p. 20503121251360090, 2025. [Online]. Available: <https://doi.org/10.1177/20503121251360090>
- [29] Ferdinand, Julius, Viriya Chow, Davy, and Yuda Prasetyo, Simeon, “Automated skin burn detection and severity classification using yolo convolutional neural network pretrained model,” *E3S Web Conf.*, vol. 426, p. 01076, 2023. [Online]. Available: <https://doi.org/10.1051/e3sconf/202342601076>
- [30] M. Yıldız, Y. Sarpdağı, M. Okuyar, N. Çiftci, A. Elkoca, M. S. Yildirim, M. A. Aydin, M. Parlak, and B. Bingöl, “Segmentation and classification of skin burn images with artificial intelligence: Development of a mobile application,” *Burns*, vol. 50, no. 4, pp. 966–979, May 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0305417924000135>
- [31] Y. Luo, K. Zhang, R. Wang, L. Yi, and Z. Zhou, “Yolov7-psa, an intelligent detection model, is designed for assessing the severity of skin burns,” *Regenesis Repair Rehabilitation*, vol. 1, no. 2, pp. 80–86, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S295057552500022X>



- [32] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLOv8.” [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [33] Ultralytics, “Yolo11 documentation,” <https://docs.ultralytics.com/models/yolo11/>, 2024, accessed: 2026-05-14.
- [34] A. B. Frederick, Y. Gavrilova, N. Koren, C. Tresslar, M. Mueller, L. McDuffie, S. A. Kahn, R. Mittal, and A. P. Leshner, “A randomized feasibility study of virtual and face-to-face care using a novel mobile health solution for outpatient pediatric burns,” *Journal of Burn Care & Research*, vol. 46, no. 6, pp. 1392–1401, nov 2025. [Online]. Available: <https://doi.org/10.1093/jbcr/iraf148>
- [35] National Institute for Health and Care Excellence, “Mersey burns technology overview,” <https://www.nice.org.uk/advice/mib58/chapter/technology-overview>, 2025, accessed: 2026-05-14.
- [36] M. Abadi *et al.*, “TensorFlow Lite: Machine learning for mobile and edge devices,” 2020, google Inc. Accedido: 2024. [Online]. Available: <https://www.tensorflow.org/lite>
- [37] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.
- [38] X. Xu, Q. Bu, J. Xie, H. Li, F. Xu, and J. Li, “On-site burn severity assessment using smartphone-captured color burn wound images,” *Computers in Biology and Medicine*, vol. 182, p. 109171, Nov. 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482524012563>
- [39] D. G. Greenhalgh, “Management of burns.” *The New England journal of medicine*, vol. 380, no. 24, pp. 2349–2359, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:189813391>
- [40] A. B. Wallace, “The exposure treatment of burns,” *The Lancet*, vol. 257, no. 6653, pp. 501–504, 1951.



- [41] C. C. Lund and N. C. Browder, “The estimation of areas of burns,” *Surgery, Gynecology & Obstetrics*, vol. 79, pp. 352–358, 1944.
- [42] K. S. Romanowski *et al.*, “Evaluation of a smartphone application as a method for calculating total body surface area burned,” *Journal of Burn Care & Research*, 2022, pMC8946009.
- [43] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 2015.
- [44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [45] F. G. Fernández, “Procesamiento de imágenes con técnicas de aprendizaje profundo para la detección de hojas en plantas,” Trabajo Fin de Grado, Universidad de Sevilla, Sevilla, España, 2022, dirigido por Rubén Martín Clemente.
- [46] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.
[Online]. Available: <https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf>
- [47] T. Dissanayake, Y. George, D. Mahapatra, S. Sridharan, C. Fookes, and Z. Ge, “Few-shot learning for medical image segmentation: A review and comparative study,” *ACM Comput. Surv.*, vol. 58, no. 1, Sep. 2025. [Online]. Available: <https://10.1145/3746224>
- [48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [49] H. Alanazi, “Optimizing medical image analysis: A performance evaluation of yolo-based segmentation models,” *Journal of Medical Imaging and Health Informatics*, 2025, yOLO-based segmentation models for medical image analysis.
- [50] Z. Cai, K. Zhou, and Z. Liao, “A systematic review of yolo-based object detection in medical imaging: Advances, challenges, and future directions,” *Computers, Materials and Continua*, vol. 85, no. 2, pp. 2255–2303, 2025.
- [51] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, “Yolact: Real-time instance segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.



- [52] L. He, Y. Zhou, L. Liu *et al.*, “Application of the yolov11-seg algorithm for ai-based landslide detection and recognition,” *Scientific Reports*, vol. 15, p. 12421, 2025. [Online]. Available: <https://doi.org/10.1038/s41598-025-95959-y>
- [53] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-net: Fully convolutional neural networks for volumetric medical image segmentation,” in *Proceedings of the 3D Vision (3DV)*, 2016.
- [54] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations,” in *MICCAI*, 2017.
- [55] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, and J. Tang, “Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [56] C. H. Sudre *et al.*, “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations,” in *Deep Learning in Medical Image Analysis*. Springer, 2017, pp. 240–248.
- [57] P. Jaccard, “The distribution of the flora in the alpine zone,” *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.
- [58] M. Everingham *et al.*, “The PASCAL visual object classes (VOC) challenge,” vol. 88, 2010, pp. 303–338.
- [59] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [60] A. Canziani, A. Paszke, and E. Culurciello, “An analysis of deep neural network models for practical applications,” *arXiv preprint arXiv:1605.07678*, 2016.
- [61] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [62] W. H. Kruskal and W. A. Wallis, “Use of ranks in one-criterion variance analysis,” *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.



- [63] O. J. Dunn, “Multiple comparisons using rank sums,” *Technometrics*, vol. 6, no. 3, pp. 241–252, 1964.
- [64] C. E. Bonferroni, “Teoria statistica delle classi e calcolo delle probabilità,” *Pubblicazioni del R. Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, vol. 8, pp. 3–62, 1936.
- [65] C. M. Nguyen and T.-S. Hy, “Efficient deep learning for medical imaging: Bridging the gap between high-performance AI and clinical deployment,” *arXiv preprint arXiv:2602.00910*, 2025.
- [66] M. Xu *et al.*, “Enable deep learning on mobile devices: Methods, systems, and applications,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 27, no. 3, pp. 1–50, 2022.
- [67] T. Choudhary *et al.*, “Challenges and obstacles towards deploying deep learning models on mobile devices,” *arXiv preprint arXiv:2105.02613*, 2021.
- [68] Y. Qin *et al.*, “On accelerating edge AI: Optimizing resource-constrained environments,” *arXiv preprint arXiv:2501.15014*, 2025.
- [69] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2018.
- [70] T. Liang *et al.*, “Pruning and quantization for deep neural network acceleration: A survey,” *Neurocomputing*, vol. 461, pp. 370–391, 2021.
- [71] S. Shi *et al.*, “A comparative survey of PyTorch vs TensorFlow for deep learning,” *arXiv preprint arXiv:2508.04035*, 2025.
- [72] I. Sommerville, *Software Engineering*, 10th ed. Boston: Pearson, 2016.
- [73] American Burn Association, “First aid for burns,” 2022, accessed: 2026-04-28. [Online]. Available: <https://ameriburn.org/public-resources/burn-prevention/first-aid/>
- [74] T. Dissanayake, Y. George, D. Mahapatra, S. Sridharan, C. Fookes, and Z. Ge, “Few-shot learning for medical image segmentation: A review and comparative study,” *ACM Comput. Surv.*, vol. 58, no. 1, Sep. 2025. [Online]. Available: <https://doi.org/10.1145/3746224>

Curriculum vitae

Ivan Avena Caro es egresado de Ingeniería Física de la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua. Es candidato a Maestro en Ingeniería en Computación por la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua. Su trabajo de tesis se tituló: *Aplicación móvil para la clasificación de quemaduras en pacientes basada en modelos de aprendizaje profundo*.

Durante sus estudios de posgrado desarrolló experiencia en visión por computadora, segmentación de imágenes médicas, aprendizaje profundo, entrenamiento de modelos supervisados, evaluación mediante métricas de segmentación y despliegue de modelos en dispositivos móviles mediante TensorFlow Lite. Su línea de interés se orienta al desarrollo de sistemas inteligentes aplicados al análisis de imágenes médicas, con énfasis en modelos de segmentación, aprendizaje con pocos datos y aplicaciones móviles de apoyo clínico.

Además, participó en el 47° Congreso Internacional de Ingeniería Electrónica del Tecnológico Nacional de México, “ELECTRO 2025”, en el cual también se desarrolló el artículo *YOLOv11 aplicado a la segmentación y clasificación de quemaduras en la piel*, publicado en la revista *Electro 2025*, ISSN electrónico: 3061-774X.

También impartió el curso *Machine Learning y Redes Neuronales* en el III Verano de Investigación Científica, UACH 2025. Asimismo, obtuvo el tercer lugar nacional en el programa *Acelerando México con Inteligencia Artificial*, Intel México 2026, con el proyecto *Inteligencia artificial para el diagnóstico de quemaduras, herramienta de apoyo para especialistas de la Unidad de Quemados del Hospital General Dr. Salvador Zubirán Anchondo, Chihuahua, Chih.*

Ivan Avena Caro

ivanavena789@gmail.com