

UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA DEL SOFTWARE

Estudio y Mejora de Técnicas de Segmentación en Imágenes Laparoscópicas a través del Aprendizaje Auto Supervisado

Desarrollado por: Ismael Tse Perdomo Rodríguez

Codirigido por: Félix José Fuentes Hurtado y Guillermo Iglesias Hernández

Madrid, 25 de julio de 2024

Estudio y Mejora de Técnicas de Segmentación en Imágenes Laparoscópicas a través del Aprendizaje Auto Supervisado

Desarrollado por: Ismael Tse Perdomo Rodríguez

Dirigido por: Félix José Fuentes Hurtado y Guillermo Iglesias Hernández

Proyecto Fin de Grado, 25 de julio de 2024

E.T.S. de Ingeniería de Sistemas Informáticos

Campus Sur UPM, Carretera de Valencia (A-3), km. 7

28031, Madrid, España

Si deseas citar este trabajo, la entrada completa en BibTeX es la siguiente:

```
@mastersthesis{citekey,  
  title = {Estudio y Mejora de Técnicas de Segmentación en Imágenes Laparoscópicas  
  a través del Aprendizaje Auto Supervisado},  
  author = {Ismael Tse Perdomo y Felix Fuentes y Guillermo Iglesias},  
  school = {E.T.S. de Ingeniería de Sistemas Informáticos},  
  year = {2024},  
  month = {7},  
  type = {Proyecto Fin de Grado}  
}
```

Esta obra está bajo una licencia Creative Commons «Atribución-NoComercial-CompartirIgual 4.0 Internacional». Obra derivada de <https://github.com/blazaid/UPM-Report-Template>.



Todo cambio respecto a la obra original es responsabilidad exclusiva del presente autor.

Resumen

Pese a los excelentes resultados obtenidos en el campo del procesamiento del lenguaje natural y la clasificación de imágenes, el paradigma Auto Supervisado no ha tenido especial relevancia respecto a la tarea de segmentación semántica. El presente Proyecto de Fin de Grado (PFG) se centra en el estudio y mejora de las técnicas actuales de segmentación de imágenes laparoscópicas, utilizando el aprendizaje auto supervisado, con el objetivo de mejorar los resultados clínicos y la seguridad de los pacientes.

Para ello, se ha implementado un *pipeline* de pre-entrenamiento auto supervisado, donde se combinan métodos discriminativos y generativos para extraer información relevante de la naturaleza de las imágenes, sin la necesidad de extraer etiquetas manualmente. Este enfoque ha sido validado mediante experimentos exhaustivos, comparando su rendimiento con dos *baselines* supervisados, que sirven a modo de representantes de las principales arquitecturas tradicionales.

Los resultados obtenidos demuestran una mejora significativa en la segmentación de imágenes laparoscópicas, con una reducción drástica de hasta el 40% en el tiempo de procesamiento comparado con las técnicas supervisadas. Estas mejoras son especialmente notables en escenarios clínicos complejos donde la variabilidad de las imágenes es alta y el margen de error es mínimo, alcanzando un *Dice Coefficient DC* de 0,95. De este modo, se abre la posibilidad de aplicar esta metodología en entornos clínicos reales, para continuar avanzando en la precisión y eficiencia de los procedimientos quirúrgicos asistidos por imagen.

Palabras clave: Deep Learning, Laparoscopia, Redes Neuronales, Transformers, Segmentación, Aprendizaje Auto Supervisado.

Abstract

Despite the remarkable success in natural language processing and image classification, the self-supervised paradigm has not been particularly impactful in the domain of semantic segmentation. This Final Degree Project aims to study and enhance current techniques for laparoscopic image segmentation using self-supervised learning, with the objective of improving clinical outcomes and patient safety.

A self-supervised pre-training pipeline was developed, integrating both discriminative and generative approaches to extract essential information from images without the need for manual labeling. This method was rigorously validated through extensive experiments, and its performance was compared to two supervised baselines representing the main traditional architectures.

The results demonstrate a significant enhancement in laparoscopic image segmentation, with a substantial reduction in processing time by up to 40% compared to supervised techniques. These improvements are especially notable in complex clinical scenarios where image variability is high and the margin for error is minimal, achieving a *Dice Coefficient* of 0,95.

This methodology shows great promise for application in real clinical settings, paving the way for continued advancements in the precision and efficiency of image-assisted surgical procedures.

Keywords: Deep Learning, Laparoscopy, Neural Networks, Transformers, Segmentation, Self Supervised Learning.

Índice general

1	Introducción	1
1.1	Contexto y Justificación	1
1.2	Motivación	2
1.3	Objetivos	3
1.4	Estructura de la memoria	4
2	Marco Teórico	6
2.1	Cirujía Laparoscópica	6
2.1.1	Definición y Origen	6
2.1.2	La técnica Laparoscópica en la Era Moderna	10
2.2	Deep Learning	12
2.2.1	Fundamentos del Deep Learning	12
2.2.2	Redes Neuronales Convolucionales	25
2.2.3	Transformers	31
2.2.4	Segmentación de Imágenes	41
2.3	Aprendizaje Auto Supervisado	46
2.3.1	Pre-Entrenamiento, <i>Transfer Learning</i> y <i>Fine Tuning</i>	46
2.3.2	Paradigmas de Aprendizaje	49
2.3.3	Aplicación en Texto e Imágenes	52
2.3.4	Discriminativo vs Generativo	54
2.3.5	Arquitectura DSSL	57
3	Trabajos Relacionados	59
4	Metodología	62
4.1	Consideraciones previas	62
4.1.1	Entorno de Trabajo	62
4.1.2	Estudio del <i>Dataset</i>	67
4.1.3	Estructura de la investigación	70
4.2	Desarrollo de los <i>baselines</i>	71

4.2.1	Acondicionamiento del <i>Dataset</i>	73
4.2.2	Pérdidas	76
4.2.3	<i>No New UNet</i>	78
4.2.4	Uformer	81
4.3	<i>Pipeline</i> de Pre-entrenamiento mediante el Aprendizaje Auto Supervisado	84
4.3.1	Preparación de los Datos	84
4.3.2	Estructura General	87
4.3.3	Fase Discriminativa	89
4.3.4	Fase Generativa	92
4.3.5	Ajuste Fino	95
5	Resultados	98
6	Impacto del Proyecto	109
6.1	Impacto social, responsabilidad ética y medioambiental	109
6.2	Impacto económico y empresarial	111
7	Conclusiones y Proyección a futuro	112
A	Ejemplos de Segmentaciones	115
B	Repositorio	119

Índice de figuras

2.1 Ejemplo de apendicectomía laparoscópica. [8].	7
2.2 Primeras herramientas para la laparoscopia.	8
2.3 Apendicectomía laparoscópica, pasado y presente.	11
2.4 Correlación entre Inteligencia Artificial, <i>Machine Learning</i> y <i>Deep Learning</i> [22].	13
2.5 Neurona biológica y Neurona artficial [24].	14
2.6 Arquitectura de red profunda con múltiples capas [27].	15
2.7 Diferentes funciones de activación [28].	16
2.8 Proceso de aprendizaje de una Red Neuronal: <i>Forward Propagation</i> , Cálculo de la pérdida y <i>Back Propagation</i> [29].	17
2.9 División del total de los datos en los conjuntos de Entrenamiento, Validación y Prueba.	19
2.10 Varianza y sesgo del modelo [34].	20
2.11 Esquema general del entrenamiento en redes neuronales. <i>Imagen obtenida de las diapositivas de la asignatura Métodos Generativos.</i>	21
2.12 <i>Overfitting</i> , <i>Underfitting</i> y ajuste Óptimo [35].	22
2.13 Características de las imágenes que activan a las neuronas pertenecientes a capas de distinta profundidad [36].	25
2.14 Estructura de una CNN [38].	26
2.15 Operación convolucional.	27
2.16 Estructura de la ResNet34 [43].	28
2.17 Pérdida de entrenamiento en relación al número de capas de una red plana y ResNet.	30
2.18 Arquitectura de un Transformer.	31
2.19 Analogía lógica con <i>word embeddings</i>	32
2.20 Bloque encoder de un Transformer [50].	34
2.21 Bloques decoder de un Transformer [50].	35
2.22 Analogía del mecanismo de <i>self-attention</i> [55].	36
2.23 Estructura del Vision Transformer [56].	37

2.24 Comparación Swin Transformer y Vision Transformer [59].	39
2.25 Desplazamiento de ventana de atención entre dos capas consecutivas [62].	40
2.26 Ejemplo de segmentación de una imagen por clases [63].	41
2.27 Estructura de un <i>AutoEncoder convolucional</i> [66].	42
2.28 Arquitectura de la U-Net [68].	43
2.29 Variantes de la U-Net [70].	44
2.30 Plataforma <i>Hugging Face</i>	46
2.31 <i>Transfer Learning</i> [76].	47
2.32 Paradigmas de aprendizaje [80].	50
2.33 <i>LeCake</i> : Metáfora de la tarta de Yann LeCun para explicar el SSL [82].	51
2.34 Enmascarado de <i>tokens</i> durante el entrenamiento de BERT [87].	52
2.35 Diferentes técnicas de Aumento de datos [90].	53
2.36 Arquitectura de SimSiam [96].	55
2.37 Arquitectura de MAE [97].	56
2.38 Arquitectura DSSL [98].	57
2.39 Mapeo al espacio latente de vistas <i>standard</i> y <i>heavy</i> [98].	58
 3.1 Frecuencia anual de los artículos publicados sobre SSL [101].	59
 4.1 Diferencia de tiempo entre CPU y GPU	64
4.2 Arquitectura global del entorno de trabajo.	65
4.3 Comparación entre Keras, TensorFlow y PyTorch [109].	66
4.4 Fotogramas y etiquetas correspondientes de CholecSeg8k [113].	67
4.5 Estructura de directorios de CholecSeg8k.	69
4.6 Flujo de trabajo de la nnUNet [121].	71
4.7 Arquitectura Uformer	72
4.8 Estructura de carpetas en el formato de la nnUNet.	74
4.9 Arquitectura del Modelo nnUNet.	80
4.10 Ejemplo del contenido del <i>dataset</i> HyperKvasir [120].	84
4.11 Resolución de las imágenes del HyperKvasir [120].	86
4.12 Arquitectura RepRec [119].	87
4.13 Arquitectura Discriminativa.	89
4.14 Arquitectura Generativa [119].	92
4.15 Entrada de los <i>feature maps</i> al <i>encoder generativo</i> [129].	93
4.16 Arquitectura Modelo Final.	95
 5.1 Intersección sobre la Unión (IoU) [131].	99

5.2	Curva de aprendizaje del nnUNet	100
5.3	Curva de aprendizaje del Uformer	101
5.4	Curva de aprendizaje de RepRec	102
5.5	Matriz de Confusión Normalizada de la nnUNet	103
5.6	Matriz de Confusión Normalizada del Uformer	104
5.7	Matriz de Confusión Normalizada de RepRec	105
5.8	Predicciones de Segmentación sobre CholecSeg.	108
A.1	Predicciones de Segmentación sobre CholecSeg (A).	115
A.2	Predicciones de Segmentación sobre CholecSeg (B).	116
A.3	Predicciones de Segmentación sobre CholecSeg (C).	117
A.4	Predicciones de Segmentación sobre CholecSeg (D).	118

Índice de tablas

4.1	Nombre, Código Hexadecimal RGB y Porcentaje del Total para todas las clases de CholecSeg8k.	68
4.2	División del <i>dataset</i> total en conjuntos de Entrenamiento, Validación y Prueba, con número total de muestras n.	73
4.3	Estadísticas extraídas por nnUNet del <i>dataset</i> antes de normalizar.	78
4.4	Estadísticas extraídas por nnUNet del <i>dataset</i> después de normalizar.	78
4.5	Resultados comparativos de diferentes métodos sobre distintos datasets[119].	88
5.1	Características Globales.	106
5.2	Coeficiente Dice e IoU por clases.	107

Índice de listados

4.1 Transformación del tamaño de las imágenes para el entrenamiento del Uformer.	75
--	----

Índice de ecuaciones

2.1	Salida linear de una neurona (2.1)	14
2.2	Salida no lineal de una neurona (2.2)	16
2.5	Mean Absolute Error (2.3)	18
2.5	Mean Squared Error (2.4)	18
2.5	Binary CrossEntropy (2.5)	18
2.7	Actualización de los pesos tras retropropagación (2.6)	18
2.7	Actualización de los <i>bias</i> tras retropropagación (2.7)	18
2.9	Regularización L1 (<i>Lasso</i>) (2.8)	23
2.9	Regularización L2 (<i>Ridge</i>) (2.9)	23
2.10	<i>Weight Decay.</i> (2.10)	23
2.11	Dropout (2.11)	24
2.12	Normalización por lotes (2.12)	24
2.13	Relación entre las dimensiones de entrada y de salida de una capa convolucional (2.13)	27
2.14	Aprendizaje de la función de identidad mediante conexión residual (2.14)	29
2.16	Similitud entre dos pares de <i>word embeddings</i> (2.15)	32
2.16	Similitud Coseno (2.16)	32
2.18	Función seno para <i>positional embeddings</i> (2.17)	33
2.18	Función coseno para <i>positional embeddings</i> (2.18)	33
2.19	<i>Self-attention</i> (2.19)	37
2.21	Mapeo lineal de la entrada al espacio latente, realizado por el encoder. (2.20)	43
2.21	Mapeo lineal del espacio latente a la salida, realizado por el decoder. (2.21)	43
4.1	Normalización Z-Score (4.1)	75
4.2	Función de pérdida de los <i>baselines</i> (4.2)	76
4.3	Función de pérdida <i>Categorical CrossEntropy</i> (4.3)	77
4.4	Función de pérdida <i>Dice Loss</i> (4.4)	77
4.5	Función de pérdida <i>Charbonnier Loss</i> (4.5)	81

4.7 Similitud Coseno Negativa (4.6)	90
4.7 Función de pérdida Contrastiva (4.7)	90

1.

Introducción

1.1. Contexto y Justificación

La cirugía laparoscópica, una técnica quirúrgica mínimamente invasiva (MIS), ha revolucionado la medicina moderna al reducir considerablemente los tiempos de recuperación del paciente, minimizar el dolor postoperatorio y disminuir el riesgo de infección en comparación con las cirugías tradicionales abiertas. Sin embargo, la limitación del campo visual, la difícil coordinación mano-ojo y la maniobrabilidad restringida inherentes a los procedimientos laparoscópicos representan desafíos significativos para los cirujanos [1], los cuales necesitan un entrenamiento quirúrgico profundo y extenso para dominarlos. Una visualización mejorada que asista a los profesionales durante el procedimiento operatorio y una navegación precisa dentro del campo quirúrgico son cruciales para superar estas limitaciones y garantizar resultados exitosos [2] que no comprometan la salud del paciente.

En este contexto, la segmentación de imágenes (definida en el subapartado 2.2.4) ha surgido en los últimos años como una tecnología crucial en la imagenología médica, debido a su potencial para mejorar la precisión y la eficiencia quirúrgica. Esta tecnología no solo ayuda en la navegación y toma de decisiones en tiempo real, sino que también facilita aplicaciones avanzadas como superposiciones de realidad aumentada, guía quirúrgica automatizada y análisis postoperatorio, sirviendo como una herramienta de retroalimentación visual en tiempo real.

La integración de la segmentación de imágenes en las cirugías laparoscópicas aprovecha los avances en *Computer Vision* (CV), Aprendizaje Automático e Inteligencia Artificial (IA), utilizando algoritmos diseñados para interpretar datos visuales complejos y proporcionando a los cirujanos una mayor conciencia situacional y precisión en su trabajo. Los métodos tradicionales de aprendizaje supervisado han sido fundamentales en la segmentación de

imágenes médicas, pero su efectividad está estrechamente ligada a la disponibilidad de grandes conjuntos de datos anotados manualmente para el entrenamiento, lo cual puede ser costoso y llevar mucho tiempo adquirir en dominios médicos [3, 4].

Este trabajo explora la aplicación del paradigma de aprendizaje Auto Supervisado dentro del ámbito de la laparoscopia, recientemente empleado con éxito para el entrenamiento de los grandes modelos de lenguaje (LLM) mostrando resultados muy prometedores. Las técnicas de aprendizaje Auto Supervisado aprovechan grandes cantidades de datos no etiquetados para pre-entrenar modelos, que luego se afinan con conjuntos de datos etiquetados más pequeños para tareas específicas. De este modo, se pretende investigar la efectividad comparativa de los métodos tradicionales supervisados, frente a los enfoques Auto Supervisados en aspectos clave como el rendimiento, tiempo de entrenamiento y relación costo-resultado.

Al analizar y comparar estas metodologías, este PFG busca investigar el potencial de integrar el aprendizaje Auto Supervisado en el campo de la segmentación de imágenes laparoscópicas. Todo esto, permite avanzar en el desarrollo de herramientas más eficientes y confiables que puedan apoyar a los cirujanos para lograr mejores resultados, mediante una guía visual mejorada durante procedimientos MIS.

1.2. Motivación

El impulso detrás de este proyecto surge de mi profundo interés por la inteligencia artificial (IA) y su potencial aplicación en ámbitos tan importantes como la tecnología médica, abordando desafíos prácticos desde su concepción hasta su implementación. Durante los últimos dos años, desde que descubrí y empecé a estudiar en detalle la Inteligencia Artificial, mi fascinación por el campo ha crecido notablemente al ir explorando sus capacidades y su relación con el aprendizaje humano. He podido apreciar cómo la IA no solo ofrece soluciones técnicas avanzadas, sino que también plantea interrogantes claves sobre la naturaleza del conocimiento y la cognición. Este proceso de exploración me ha llevado a valorar aún más la capacidad de la IA para aprender de manera autónoma y adaptarse a contextos cambiantes, carac-

terísticas que reflejan sorprendentemente cómo los seres humanos adquirimos y aplicamos conocimientos en diversos campos.

Por otro lado, las cirugías laparoscópicas representan un avance significativo en la medicina moderna debido a su carácter mínimamente invasivo y su capacidad para acelerar la recuperación de los pacientes. La integración de la IA en estos procedimientos ofrece una oportunidad significativa para mejorar la precisión quirúrgica y los resultados. Específicamente, la IA puede asistir y complementar a los cirujanos y profesionales médicos durante las operaciones, facilitando las cirugías remotas y ampliando el acceso a cuidados especializados en áreas menos accesibles por todo el mundo.

Al abordar este desafío, aspiro a contribuir en la medida de lo posible a esta intersección entre la IA y la medicina. Este proyecto no solo me permite aplicar conocimientos teóricos, sino también innovar de manera práctica desarrollando soluciones con impacto directo en el dominio clínico. A través de este esfuerzo, espero mejorar mis habilidades en IA y realizar una contribución tangible, trabajando para mejorar el cuidado de los pacientes y los resultados quirúrgicos a nivel global.

1.3. Objetivos

El estudio y desarrollo de las distintas metodologías que se han llevado a cabo en el proyecto, tienen como objetivo principal implementar un modelo capaz de realizar segmentación de imágenes laparoscópicas de manera precisa, fiable y eficiente. De esta manera, se plantean los siguientes objetivos específicos:

- **Explorar en detalle la naturaleza de la técnica laparoscópica:** Comprender sus fundamentos, aplicaciones clínicas y los desafíos específicos que presenta la segmentación de imágenes en este contexto, con el fin de adaptar y optimizar los modelos desarrollados para las particularidades de este tipo de imágenes. De esta manera se sientan unas bases sólidas para desarrollar soluciones avanzadas y efectivas que puedan ser implementadas con éxito en escenarios clínicos reales.
- **Estudiar el desempeño de modelos basados en el paradigma de Apre-**

dizaje Supervisado: Evaluar la efectividad y la precisión de estos modelos, analizando cómo se adaptan y responden a las variaciones y desafíos específicos de este entorno médico. Para ello, se implementarán dos baselines, uno basado en CNNs y otro basado en Transformers. Estos servirán como representantes de las dos arquitecturas que mejores resultados han ofrecido tradicionalmente en este tipo de problemas. Estos modelos utilizarán imágenes laparoscópicas junto con sus segmentaciones como entrada, con el objetivo de analizar y captar sus características principales.

- **Investigar y desarrollar algoritmos de Pre-entrenamiento Auto Supervisado:** Se propone aplicar el paradigma de Aprendizaje Auto Supervisado (SSL) en el ámbito de las imágenes médicas, para reducir los tiempos de entrenamiento y mejorar los resultados. De este modo, se pretende prescindir de un conjunto de datos con imágenes segmentadas por expertos. Se aplicarán técnicas discriminativas y generativas con el fin de explotar al máximo el potencial de estos enfoques.
- **Contrastar el desempeño de las distintas arquitecturas neuronales:** Se realizará una comparación de los distintos modelos entrenados, mediante el cálculo de distintas métricas relevantes, que ayudarán a tangibilizar los resultados obtenidos y extraer conclusiones trascendentales.

1.4. Estructura de la memoria

El presente documento se estructura de la siguiente manera. En primer lugar, en la sección *Marco Teórico* se ofrece una visión general de los conceptos teóricos indispensables para comprender el proyecto en su totalidad, abarcando los fundamentos de la cirugía laparoscópica y el *Deep Learning* (DL), así como las distintas arquitecturas y paradigmas utilizados.

Más adelante, en la sección *Trabajos Relacionados* se realiza un análisis de la literatura actual y los trabajos más destacados dentro de este ámbito. Además, en la *Metodología* se explica detalladamente el proceso de desarrollo y evolución de los diferentes modelos y configuraciones llevadas a cabo para abordar los objetivos del PFG, mientras que en la sección *Resultados* se exponen los hallazgos más destacables y relevantes para derivar conclusiones

pertinentes.

En el capítulo de *Conclusiones y Proyección a futuro*, se añaden las reflexiones finales del trabajo y los próximos pasos o perspectivas futuras del proyecto en el marco de la industria y su aplicación real. Finalmente, en el *Impacto del Proyecto* se discuten las diversas consecuencias sociales, ambientales y profesionales del PFG.

2.

Marco Teórico

2.1. Cirugía Laparoscópica

La cirugía laparoscópica ha revolucionado el campo de la medicina, marcando un cambio significativo en la manera en que se llevan a cabo las intervenciones quirúrgicas. Desde sus inicios hasta la actualidad, la laparoscopia ha pasado de ser un procedimiento puramente diagnóstico a una técnica quirúrgica completa, con aplicaciones en múltiples disciplinas médicas. Este desarrollo ha sido posible gracias a una serie de innovaciones tecnológicas y a la dedicación de pioneros que han impulsado su evolución.

2.1.1. Definición y Origen

Según el Instituto Nacional del Cáncer estadounidense (NCI por sus siglas en inglés) la **laparoscopia** es un procedimiento que utiliza un laparoscopio, un tubo delgado con luz y lente insertado a través de la pared abdominal, para examinar el interior del abdomen [5]. Se hace una pequeña incisión cerca del ombligo y se inserta un trócar¹, que actúa como conducto para el laparoscopio y otros instrumentos quirúrgicos. Luego, se insufla gas de dióxido de carbono (CO_2) en la cavidad abdominal para crear un neumoperitoneo, elevando la pared abdominal y proporcionando al cirujano una vista clara y más espacio para operar [6]. Esto se consigue insertando un laparoscopio con cámara y luz que transmite imágenes en tiempo real a un monitor de video, que servirá de gran ayuda al doctor para cortar tejido, suturar, extraer órganos o cauterizar vasos sanguíneos durante la intervención. Al finalizar, se retiran los instrumentos y los trócares, se libera el gas CO_2 de la cavidad abdominal y se

¹Un trócar es un instrumento quirúrgico con una punta afilada que se utiliza para crear una apertura en la pared corporal, permitiendo la inserción de otros instrumentos como el laparoscopio.

cierran las pequeñas incisiones con suturas, grapas o pegamento quirúrgico [7]. En la figura 2.1 se muestra un ejemplo de apendicectomía² laparoscópica.

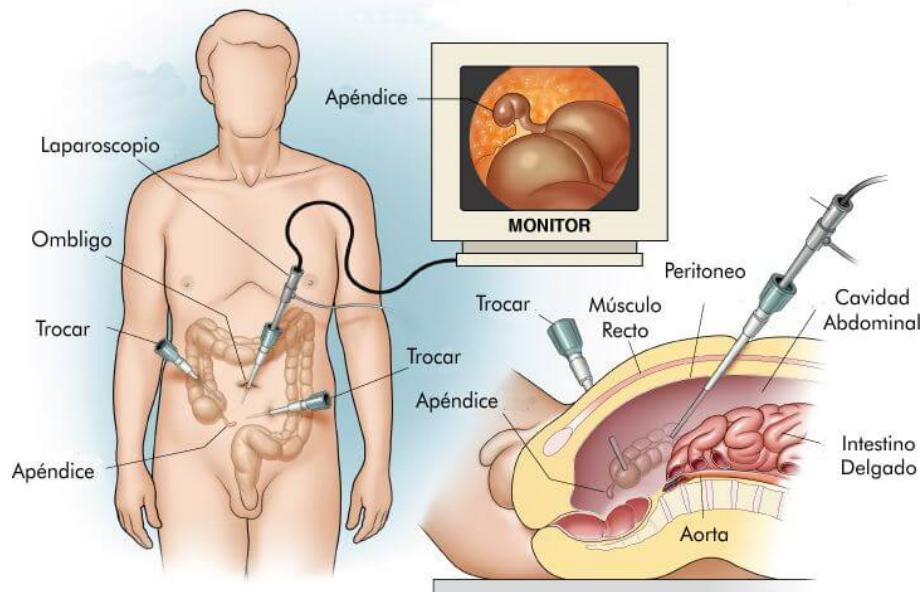


Figura 2.1. Ejemplo de apendicectomía laparoscópica. [8].

La laparoscopia, derivada de los términos griegos “lapara” (las partes blandas del cuerpo entre las costillas y las caderas) y “skopein” (ver o examinar), tiene sus raíces en la antigua Grecia, donde Hipócrates ya describía el uso de especulos para examinar el recto [9]. Más tarde ya en el siglo XX, Georg Kelling, un cirujano alemán, realizó en 1901 el primer procedimiento laparoscópico en un perro [10, 11], utilizando un cistoscopio para visualizar la cavidad abdominal. Nueve años después, Hans-Christian Jacobaeus realizó la primera laparoscopia en humanos, introduciendo el término en el ámbito médico [12]. Estos primeros procedimientos eran principalmente diagnósticos y no se realizaban intervenciones quirúrgicas complejas.

²La apendicectomía es una intervención quirúrgica que tiene como objetivo extirpar el apéndice, comúnmente para tratar la apendicitis.

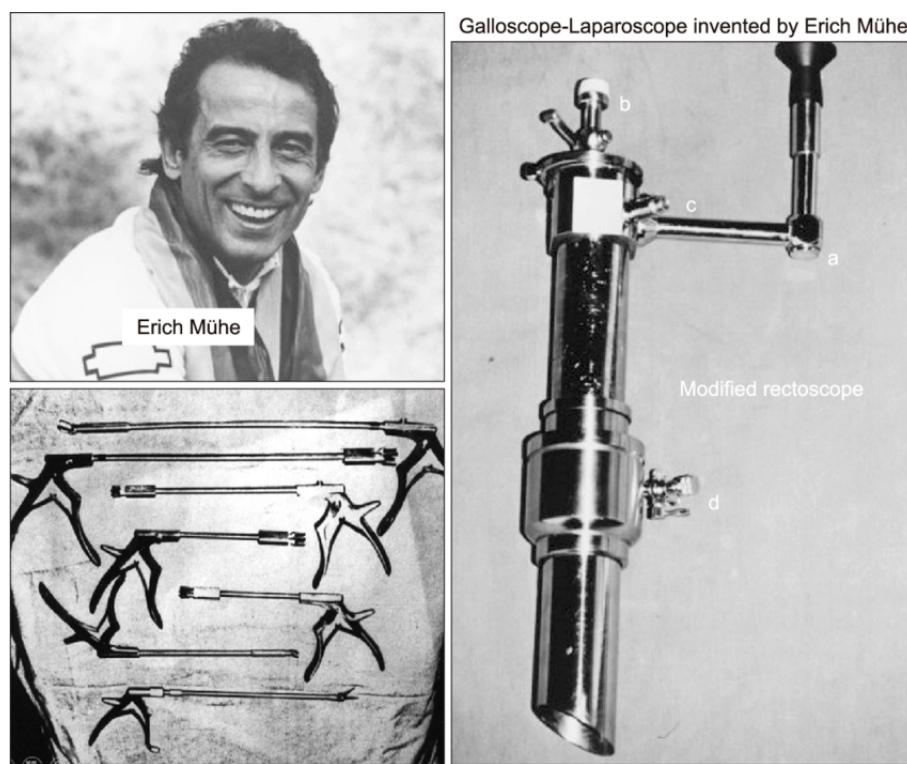


Figura 2.2. El cirujano alemán Erich M. Mühe junto a un rectoscopio modificado e instrumentos de empuñadura de pistola con óptica e insuflación de CO₂. a = óptica de vista lateral; b = canal de instrumentación con válvulas; c = conductor de luz; d = canal de gas [13].

En las décadas de 1960 y 1980, la laparoscopia experimentó avances significativos que la transformaron en una técnica quirúrgica viable, protagonizados por el Dr. Erich Mühe en Alemania que llevó a cabo la primera colecistectomía³ (figura 2.2), Raoul Palmer y Kurt Semm. Estos últimos fueron dos de los pioneros más influyentes en esta transformación. Palmer, en París, desarrolló técnicas para la visualización del abdomen, mientras que Semm, en Kiel, realizó la primera apendicectomía laparoscópica en 1980. Este procedimiento que más adelante demostró el potencial de la laparoscopia para reducir el trauma quirúrgico y mejorar la recuperación del paciente, fue inicialmente recibido con escepticismo y oposición por parte de la comunidad médica. A continuación se muestra un fragmento de una crítica publicada en 1983 por la revista *Medical Tribune* sobre el trabajo de Kurt Semm:

³La colecistectomía es la intervención quirúrgica consistente en la extracción de la vesícula biliar y es el método más común para tratar distintas patologías de este órgano.

"Semm exaggerates the problem of adhesions only in order to find a justification for his key-hole surgery...Thanks to modern methods of anesthesia, laparotomy today no longer poses a problem. This is the only way for a surgeon to be able to view the entire abdomen and to direct his procedure accordingly. Post-operative adhesions can lead to complications, but they in no way occur with such frequency that one must switch to endoscopic operations, believes Prof. Bruecke. Many superfluous operations are being carried out even today. The danger in expanding the endoscopic appendectomy, which only seems to be easier and less dangerous to perform than conventional methods, is that still more unnecessary appendectomies will be performed than have been to date. We thus face the following fundamental question: Do the advantages of endoscopic operations—avoidance of laparotomy, diminishing the pain of the incision, early mobilization, and avoidance of post-operative adhesions—outweigh the disadvantages—greater expenditure on technology and more complicated methods of operating?" [14]

2.1.2. La técnica Laparoscópica en la Era Moderna

Hoy en día, la laparoscopia se utiliza en una amplia variedad de procedimientos quirúrgicos, incluyendo la colecistectomía mencionada previamente, la apendicectomía, histerectomía, y cirugías oncológicas, extendiéndose a especialidades como la urología, la ginecología y la cirugía gastrointestinal [15]. Su popularidad y relevancia actual radican en la reducción significativa del tiempo de recuperación del paciente y la disminución del dolor postoperatorio y el riesgo de infecciones comparado con las cirugías abiertas tradicionales.

Para llegar hasta aquí, la tecnología ha jugado un papel crucial. En la era moderna, la incorporación de sistemas de imagenología de alta definición, como la laparoscopia en 4K y 3D, ha mejorado drásticamente la visualización intraoperatoria, como se muestra en la figura 2.3. Estas tecnologías permiten a los cirujanos obtener imágenes más nítidas y detalladas de las estructuras anatómicas, facilitando la identificación de tejidos y órganos.

Además, mediante la introducción de endoscopios delgados y flexibles [16], sistemas de insuflación de dióxido de carbono y demás, se ha conseguido que estas complicadas intervenciones sean sustancialmente más seguras reduciendo, por ejemplo, el porcentaje de lesiones del conducto biliar del 0.69 % reportado en 1999, al 0.22 % reportado en 2015 [17]. A esto se le suma el desarrollo de diferentes técnicas durante los últimos años, como la cirugía endoscópica transluminal a través de orificios naturales (NOTES, de sus siglas en inglés) [18] y la cirugía asistida por robots.

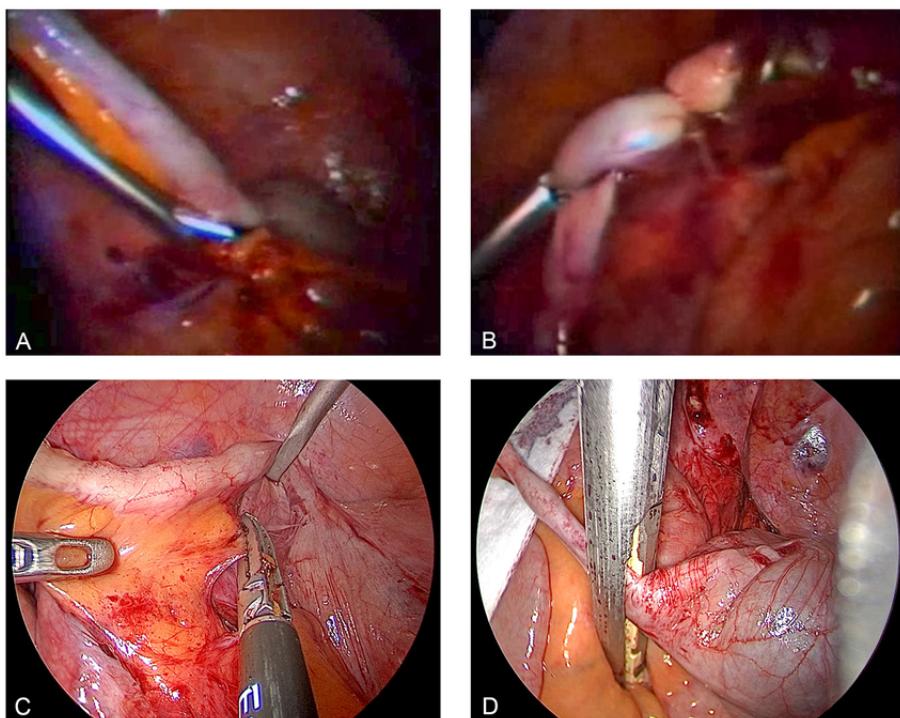


Figura 2.3. Apendicectomía laparoscópica, pasado y presente. (A, B) son imágenes originales de la primera appendicectomía laparoscópica realizada por Semm en 1980: la esqueletización del apéndice fue seguida por su ligadura en la base con un lazo Roeder. (C, D) muestran el estándar actual de la cirugía con el uso de una grapadora. La notablemente mejor calidad de imagen y los instrumentos avanzados son evidentes [19].

La cirugía asistida por robots, más conocida como cirugía robótica, representa otro salto cualitativo en la laparoscopia moderna, ya que permite a los cirujanos realizar movimientos extremadamente precisos mediante brazos robóticos que replican los movimientos de las manos del cirujano con mayor estabilidad y reduciendo los temblores. Pese a que en un principio la cabina desde la que opera el doctor fue pensada para estar a pocos metros del paciente, en la misma sala quirújica, poco a poco se ha progresado en la idea de implantar sistemas confiables de cirugía remota [20].

2.2. Deep Learning

A continuación, se definen los fundamentos del *Deep Learning* y las principales arquitecturas utilizadas en el grueso de este PFG, con el objetivo de dar suficiente contexto al lector para comprender satisfactoriamente el trabajo realizado.

2.2.1. Fundamentos del Deep Learning

El **Aprendizaje Profundo** o *Deep Learning* en inglés, es una subdisciplina del aprendizaje automático o *Machine Learning* (ML), que se centra en el uso de redes neuronales artificiales con múltiples capas para modelar y resolver problemas complejos. A diferencia de los métodos tradicionales de ML, que a menudo requieren un proceso manual de elicitation de características o *features*, el DL es capaz de aprender automáticamente representaciones jerárquicas de datos a través de capas sucesivas de transformación.

Más en detalle, la IA engloba a todos los sistemas capaces de realizar tareas que requieren inteligencia humana, como el razonamiento, la planificación y el procesamiento del lenguaje (por ejemplo sistemas expertos, que se basan en la definición manual de una serie de reglas de conocimiento sobre las que el sistema realiza el proceso de inferencia [21]). Dentro de este campo se encuentra la subdisciplina del ML, que se enfoca en crear algoritmos que permiten a las máquinas aprender a partir de una serie de datos de entrada y mejorar su desempeño en tareas específicas sin ser programadas explícitamente para ello. Esta correlación entre los distintos ámbitos se muestra en la figura 2.4:

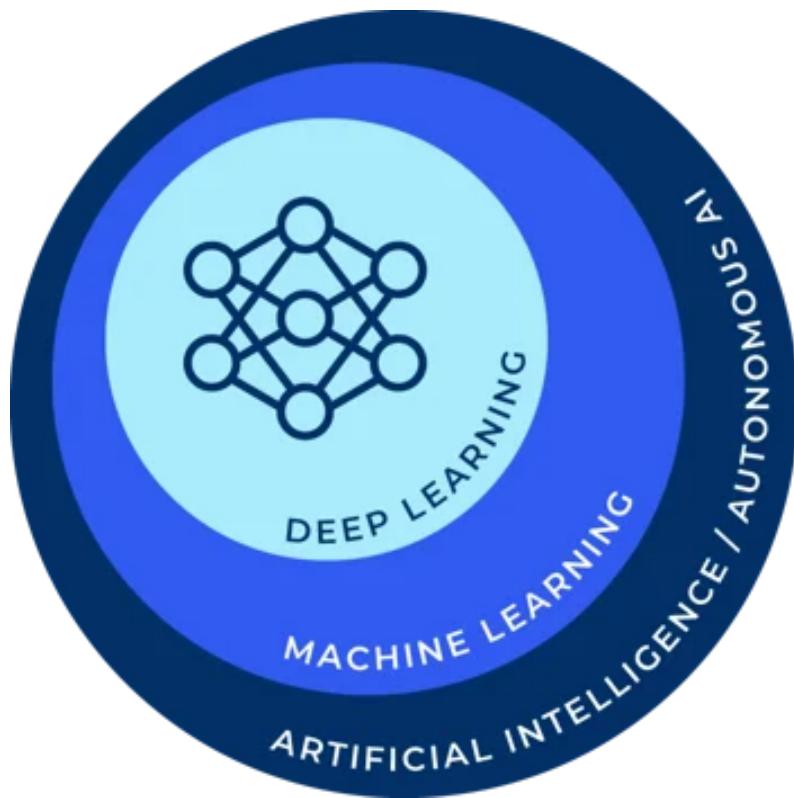


Figura 2.4. Correlación entre Inteligencia Artificial, Machine Learning y Deep Learning [22].

A su vez, como se ha mencionado antes, el DL es una subrama del machine learning que destaca fundamentalmente por el empleo de redes neuronales profundas con múltiples capas densamente conectadas entre sí.

Redes Neuronales

Las **Redes Neuronales** o *Neural Networks* del inglés (NN) fueron propuestas por primera vez por McCulloch y Pitts en 1943 [23] y son sistemas matemáticos capaces de aprender a realizar predicciones a partir de unos datos de entrada, intentando imitar el funcionamiento de las neuronas biológicas. De manera tangible, estas neuronas artificiales no son un objeto físico, sino un concepto implementado en software. Esto es, una unidad de cálculo representada por un trozo de código que se ejecuta en un *hardware* determinado. Su cometido individual es calcular el valor de una salida (*output*) a partir de combinar un bias con los pesos asociados a cada una de las entradas (*inputs*), y aplicarle una función de activación (figura 2.5).

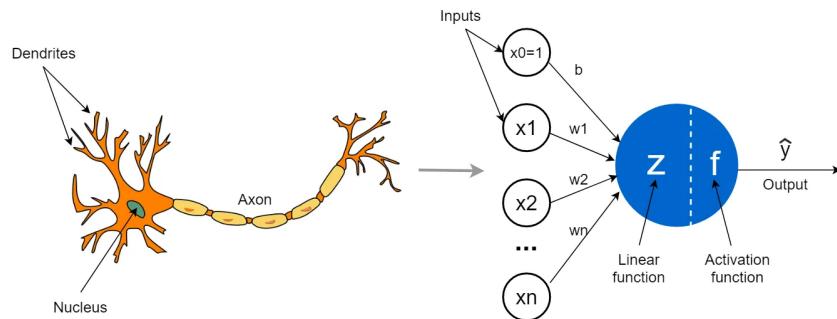


Figura 2.5. Neurona biológica y Neurona artficial [24].

Los **pesos** son parámetros que cambian con cada iteración del entrenamiento y están vinculados a cada entrada de la neurona. De manera simplificada, cuanto mayor es el valor del peso de una entrada, más relevante es su información para realizar el cálculo de la salida. El **bias** por otro lado, es un peso particular que presenta siempre un valor constante, generalmente 1.

Siguiendo el ejemplo de la figura 2.5, la salida z de una neurona viene dada por la ecuación (2.1), donde w , x y b son los pesos, entradas y *bias*, respectivamente:

$$z = \sum_{i=1}^n (w_i \cdot x_i) + b = w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n + b \quad (2.1)$$

Las neuronas se organizan en capas para formar una red neuronal (figura 2.6), contando con una capa de entrada que recibe los datos iniciales, capas ocultas que procesan la información intermedia y extraen características a diferentes niveles de abstracción, y una capa de salida que produce la predicción final o el resultado de la red. En las redes densas, cada neurona en una capa está conectada a todas las neuronas de la siguiente. Formando una estructura compleja que permite a la red aprender y generalizar a partir de los datos de entrenamiento [25]. Para considerar que una red neuronal es profunda, esta debe contar con al menos tres capas ocultas [26].

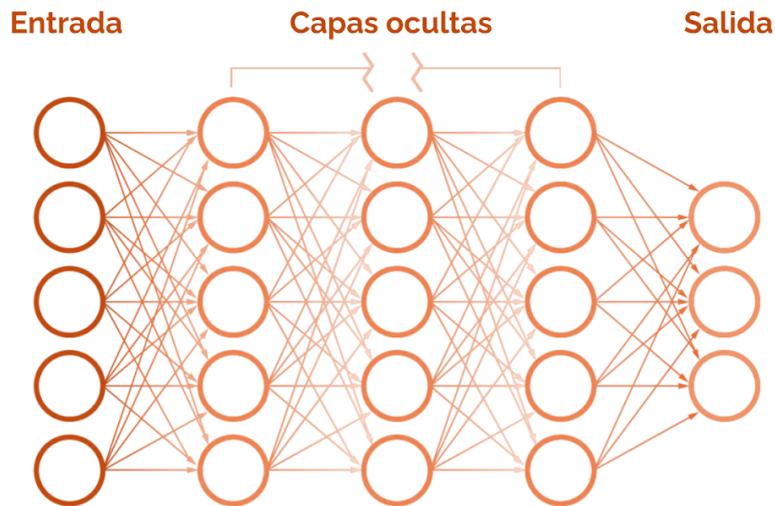


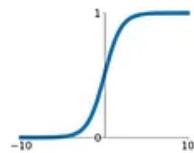
Figura 2.6. Arquitectura de red profunda con múltiples capas [27].

Sin embargo, para que estas redes puedan aprender a imitar los patrones presentes en distribuciones de datos complejas, necesitan una **función de activación**. Sin ella, la red se limitaría a realizar combinaciones lineales de las entradas, independientemente del número de capas. Dichas funciones facilitan la propagación efectiva de señales y gradientes durante el entrenamiento y proporcionan la flexibilidad necesaria para abordar diversos tipos de problemas y datos.

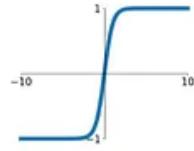
En la figura 2.7 se muestran algunas de las funciones de activación más relevantes. Entre ellas destaca la función **ReLU** (*Rectified Linear Unit*) y sus variantes (LeakyReLU, ELU, GELU, etc.) omnipresentes en la mayoría de dominios del DL, siendo la opción más popular para las activaciones de las capas ocultas por su sencillez y eficacia. Los modelos pueden aprender mucho más rápido que con otras funciones como la sigmoidal o la tangencial, pues mientras la activación sea positiva el gradiente será elevado. Estas últimas son más utilizadas en redes neuronales recurrentes y en las capas finales, pues su naturaleza permite simplificar tareas como la clasificación binaria .

Sigmoid

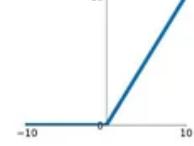
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**tanh**

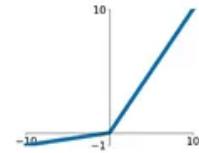
$$\tanh(x)$$

**ReLU**

$$\max(0, x)$$

**Leaky ReLU**

$$\max(0.1x, x)$$

**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

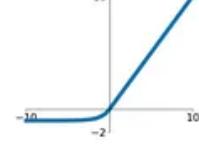


Figura 2.7. Diferentes funciones de activación [28].

De este modo, se define la ecuación (2.2), siendo a la activación, l la capa y $f(x)$ la función de activación:

$$a^{[l]} = f\left(\sum_{i=1}^n (w_i^{[l]} \cdot x_i^{[l-1]}) + b^{[l]}\right) \quad (2.2)$$

Aprendizaje y Optimización

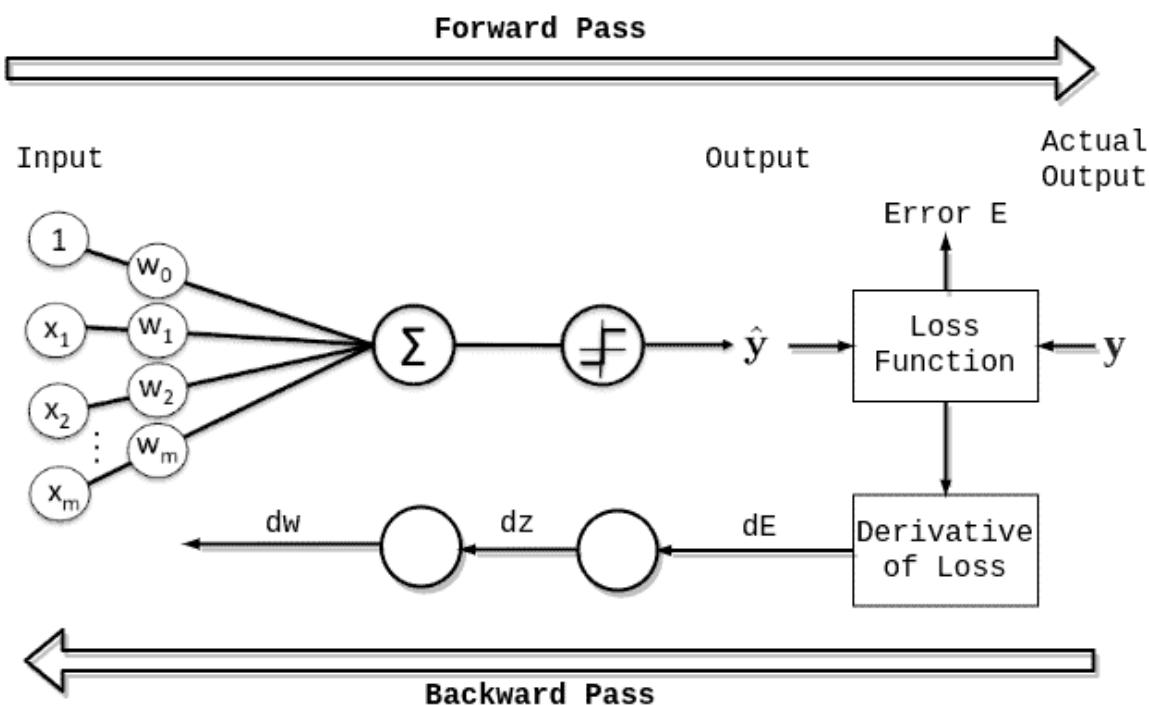


Figura 2.8. Proceso de aprendizaje de una Red Neuronal: *Forward Propagation*, Cálculo de la pérdida y *Back Propagation* [29].

A una red neuronal entrenada para una tarea concreta se le conoce como **modelo** y su aprendizaje ocurre por iteraciones o épocas de manera progresiva. En la primera, los pesos y *bias* de toda la red se inicializan a 0, valores aleatorios o siguiendo algún otro método como los propuestos por Xavier Glorot o Kaiming He [30]. En cada época se lleva a cabo el algoritmo de descenso de gradiente, que consta de tres fases (figura 2.8):

- **Forward Propagation:** Es el proceso mediante el cual una red neuronal procesa una entrada para generar una salida. En este proceso, los datos de entrada se pasan a través de la red, capa por capa, aplicando la ecuación (2.2) en cada una de ellas. Cada neurona en una capa recibe entradas de las neuronas de la capa anterior, aplica los pesos y *bias* correspondientes, produciendo una salida que se convierte en la entrada para la siguiente capa. El resultado final es el **output** de la red.
- **Cálculo de la pérdida:** La salida de la última capa o **predicción** \hat{y} en una época cualquiera, es comparada con la etiqueta o valor esperado

y , utilizando una función determinada $L(\hat{y}, y)$. El objetivo del aprendizaje es reducir este valor para predecir correctamente el mayor número de datos posible. Existe una amplia paleta de funciones de pérdida que pueden ser aplicadas en dominios muy variados. A continuación se presentan las funciones de **MAE** (Mean Absolute Error, 2.3), **MSE** (Mean Squared Error, 2.4) y **BCE** (Binary CrossEntropy, 2.5):

$$\mathcal{L}_{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.3)$$

$$\mathcal{L}_{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.4)$$

$$\mathcal{L}_{BCE} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2.5)$$

- **Back Propagation:** Es el algoritmo encargado de calcular los gradientes parciales de los pesos w y $bias$ de la red, con respecto a la función de pérdida utilizada, para posteriormente actualizarlos haciendo uso la regla de la cadena⁴. Los gradientes se propagan desde la capa de salida hasta las capas de entrada y su importancia en los nuevos valores se pondera mediante la tasa de aprendizaje α . Esta actualización de pesos y $bias$ se lleva a cabo mediante las fórmulas (2.6) y (2.7) respectivamente:

$$w = w - \alpha \cdot \frac{\partial L(\hat{y}, y)}{\partial w} \quad (2.6)$$

$$b = b - \alpha \cdot \frac{\partial L(\hat{y}, y)}{\partial b} \quad (2.7)$$

Cuando el proceso de ajuste de los parámetros del modelo ha alcanzado un punto en el que las mejoras en la función de pérdida se han estabilizado, se dice que el modelo ha **convergido**. Para acelerar esta convergencia pueden utilizarse diversas variaciones del descenso de gradiente, Adam, RMSprop o

⁴La regla de la cadena es una regla fundamental en cálculo diferencial que se utiliza para calcular la derivada de una función compuesta.

mini-batch gradient descent. En concreto, ADAM (*Adaptive Moment Estimation*) es el algoritmo de optimización más extendido actualmente en la mayoría de dominios, debido a la estabilidad, escalabilidad y eficiencia que proporciona con grandes conjuntos de datos [31]. A diferencia del *Stochastic Gradient Descent* (SGD) donde la tasa de aprendizaje es estática y no atiende a valores anteriores de los pesos y *bias*, Adam ajusta dicho hiperparámetro dinámicamente haciendo uso de medias móviles [32].

División de los Datos, Overfitting y Underfitting

Para poder comparar el rendimiento del modelo con los datos sobre los que se ha entrenado y los que nunca ha visto, se suele dividir el grueso total de ejemplos disponibles en los subconjuntos de entrenamiento, validación y prueba (figura 2.9). Las muestras separadas para el **entrenamiento** suponen la gran mayoría de los datos disponibles y se utilizan, como su nombre indica, para entrenar el modelo. Si el número de muestras es bajo, generalmente se asigna al menos un 70 % del total a este conjunto. A medida que aumenta el número de muestras, también lo hace este porcentaje, llegando hasta el 98 % para grandes *datasets* de millones de datos [33]. A medida que el entrenamiento avanza, se evalúa el modelo utilizando el conjunto de **validación**. Este se utiliza para comprobar el rendimiento del modelo sobre datos nuevos, mientras se ajustan los hiperparámetros. Una vez se termina con este proceso, se alimenta a la red con el conjunto de **prueba** para valorar el desempeño global y valorar el grado de generalización del modelo.

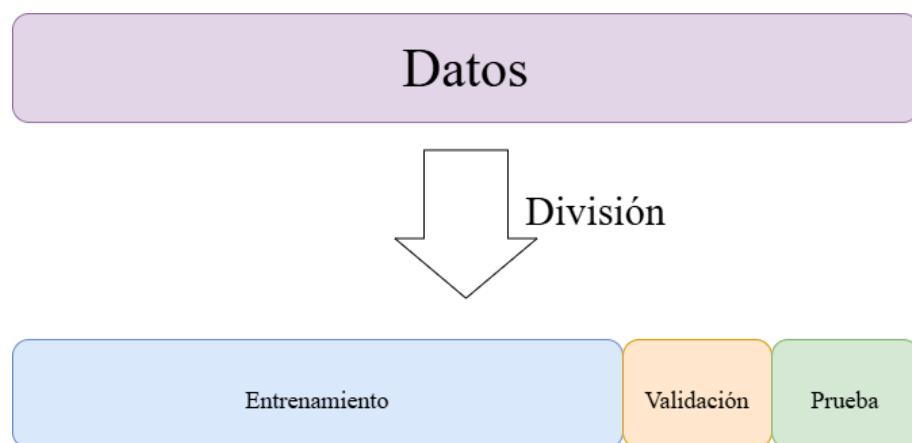


Figura 2.9. División del total de los datos en los conjuntos de Entrenamiento, Validación y Prueba.

A partir del desempeño del modelo sobre estos conjuntos se determina la **varianza** y el **sesgo** de la red. En la figura 2.10 se muestra cómo están relacionados estos conceptos según la complejidad del modelo y las pérdidas del mismo.

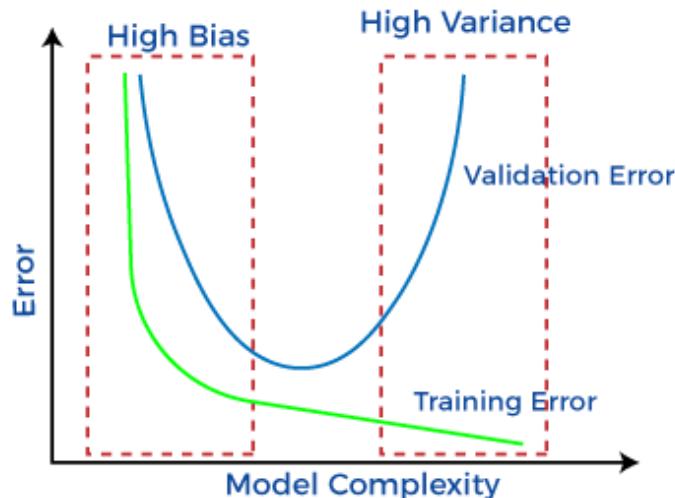


Figura 2.10. Varianza y sesgo del modelo [34].

La varianza se refiere a la diferencia entre la pérdida del conjunto de entrenamiento y el error óptimo de Bayes⁵, mientras que el sesgo se corresponde a la discrepancia entre esta pérdida y la del conjunto de validación. El cálculo de estos valores permite planificar una hoja de ruta con la que mejorar el rendimiento de la red (figura 2.11).

⁵El error óptimo de Bayes (*Optimal Bayes Error*) es el error mínimo teórico que se puede alcanzar al clasificar datos utilizando un clasificador Bayesiano óptimo. Representa la mejor tasa de error posible dada la distribución de los datos y es un límite teórico que los clasificadores no pueden superar en condiciones ideales.

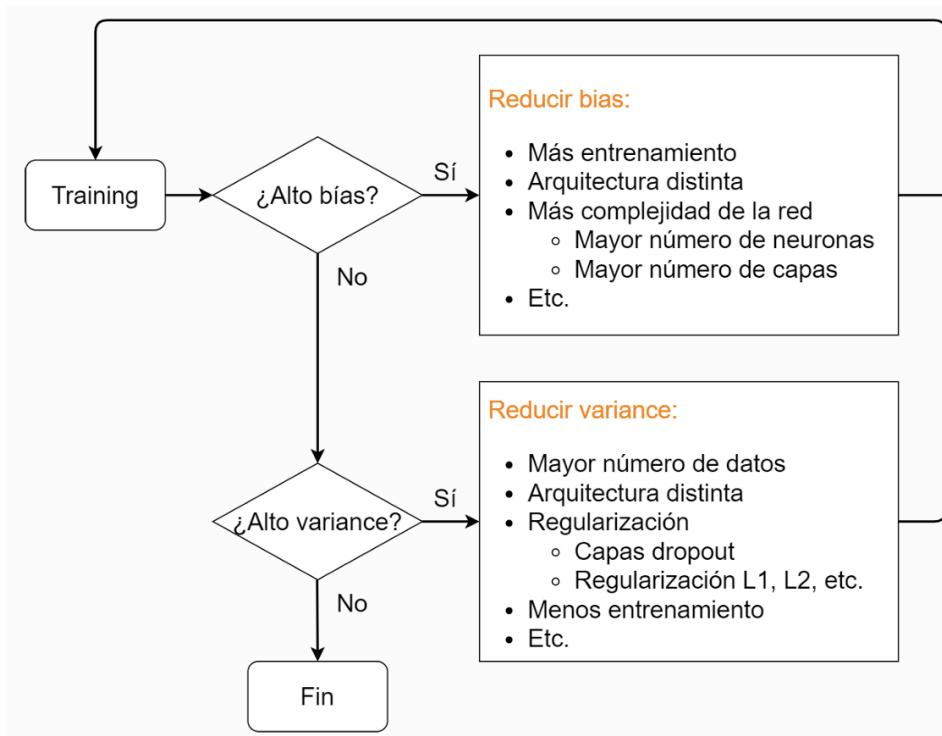


Figura 2.11. Esquema general del entrenamiento en redes neuronales.

Imagen obtenida de las diapositivas de la asignatura *Métodos Generativos*.

Por otro lado, el **Overfitting** y el **Underfitting** son dos problemas habituales en el aprendizaje automático, que afectan a la capacidad de un modelo para generalizar a datos no vistos. El *overfitting* ocurre cuando la red neuronal es excesivamente compleja y se ajusta demasiado bien a los datos que recibe, capturando no solo los patrones subyacentes, sino también el ruido y las anomalías. Como resultado, el modelo es incapaz de generalizar bien con datos que nunca ha visto, debido a que el modelo ha "memorizado" los datos en lugar de aprender los patrones generales. Por otro lado, el *underfitting* ocurre cuando el modelo es demasiado simple para capturar dichos patrones, obteniendo un rendimiento pobre, tanto en datos conocidos como desconocidos. Esto suele deberse a ir relacionado con una necesidad de aumentar la complejidad de la red, de entrenar por más tiempo o utilizar otras arquitecturas.

El objetivo en el aprendizaje automático es encontrar un equilibrio adecuado donde el modelo sea lo suficientemente complejo para capturar los patrones de los datos, pero no tan complejo como para ajustarse al ruido, logrando así una buena capacidad de generalización (figura 2.12).

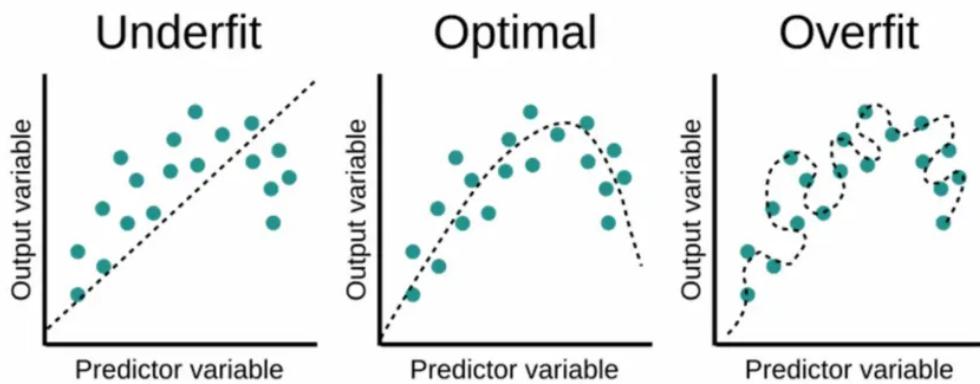


Figura 2.12. *Overfitting, Underfitting y ajuste Óptimo* [35].

Regularización

Con el fin de combatir el fenómeno del *Overfitting*, las técnicas de regularización pretenden agregar términos adicionales a las funciones de pérdida para penalizar los parámetros del modelo en función de su magnitud. Esto fomenta que el modelo prefiera soluciones más simples y suaves, lo que a menudo conduce a una mejor capacidad de generalización en datos nuevos y no vistos. Al controlar la complejidad del modelo, la regularización ayuda a mejorar su capacidad para hacer predicciones precisas y estables en una variedad de situaciones y datos. Los principales métodos de normalización son los siguientes:

- La regularización **L1**, también conocida como **regularización de Lasso**, penaliza los coeficientes del modelo basándose en la suma de sus valores absolutos. Esto fomenta la esparcidad, lo que significa que al-

gunos coeficientes pueden volverse exactamente cero, reduciendo así la complejidad del modelo y seleccionando características relevantes. Por otro lado, **L2**, conocida como **regularización de Ridge**⁶, penaliza los coeficientes basándose en la suma de los cuadrados de estos coeficientes, lo que ayuda a evitar coeficientes grandes sin anularlos.

Las ecuaciones (2.8) y (2.9) que definen ambas técnicas se encuentran a continuación, donde θ es un parámetro de la red y p es el número total de parámetros:

$$\text{L1 (Lasso): } \mathcal{L}_{\text{L1}}(\theta) = \|\theta\|_1 = \sum_{i=1}^p |\theta_i| \quad (2.8)$$

$$\text{L2 (Ridge): } \mathcal{L}_{\text{L2}}(\theta) = \|\theta\|_2^2 = \sum_{i=1}^p \theta_i^2 \quad (2.9)$$

Al aplicar una de ellas, por ejemplo L2, a una función de pérdida cualquiera \mathcal{L} , se obtiene la expresión (2.10), donde λ es el **índice de regularización** (también conocido como *weight decay* en el caso de L2).

$$\mathcal{L}_{\text{Final}} = \mathcal{L} + \frac{\lambda}{2} \cdot \text{L2} \quad (2.10)$$

- Por otra parte, el **dropout** es una técnica de regularización específica para redes neuronales que consiste en desactivar aleatoriamente un conjunto de unidades neuronales durante el entrenamiento. Esto ayuda a prevenir el sobreajuste al evitar que las neuronas se especialicen demasiado en los datos de entrenamiento específicos y, en cambio, fomenta que cada neurona contribuya de manera más general a las predicciones del modelo. El *dropout* se aplica únicamente durante el entrenamiento, pues durante la validación y el testeo todas las unidades deben estar activas para generar predicciones finales.

De este modo, con una probabilidad p de *dropout*, cada activación intermedia a se reemplaza por una variable aleatoria a' siguiendo la fórmula (2.11):

⁶también llamada en ocasiones *Weight Decay*

$$a' = \begin{cases} 0 & \text{con probabilidad } p \\ \frac{a}{1-p} & \text{en otro caso} \end{cases} \quad (2.11)$$

- Por último, la **normalización por lotes** (*batch normalization*, BN) es una técnica que normaliza las entradas de cada capa de una red neuronal, estabilizando y acelerando el proceso de entrenamiento. Esto lo consigue ajustando y escalando los valores de mini lotes de datos para que tengan una media cercana a cero y una varianza cercana a uno. Esto facilita el entrenamiento de redes neuronales profundas al mitigar problemas como el desvanecimiento del gradiente y mejorar la convergencia del modelo.

En la ecuación (2.12) la variable z representa la salida lineal de una neurona antes de aplicar la función de activación. Se utiliza μ para denotar su media y σ^2 para representar su varianza, con ϵ como un pequeño término añadido para mantener la estabilidad numérica. z_{norm} es la salida lineal resultante después de normalizar, ajustada por los parámetros entrenables γ y β , que representan respectivamente el factor de escala aprendido y el sesgo aprendido durante el entrenamiento del modelo. Por último, \tilde{z} es la salida lineal a la que se le aplicará la función de activación de la neurona:

$$\begin{aligned} z_{\text{norm}} &= \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}}, \\ \tilde{z} &= \gamma \cdot z_{\text{norm}} + \beta \end{aligned} \quad (2.12)$$

2.2.2. Redes Neuronales Convolucionales

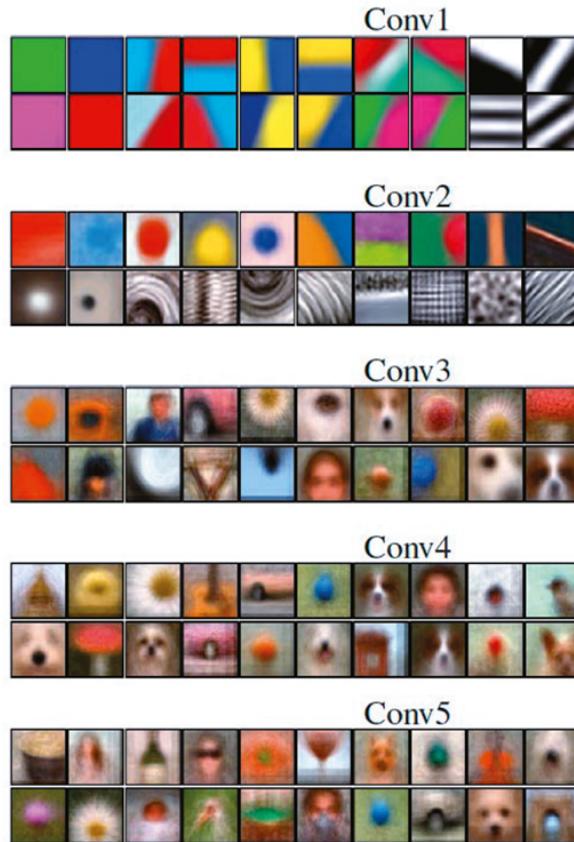


Figura 2.13. Características de las imágenes que activan a las neuronas pertenecientes a capas de distinta profundidad [36].

Las **Redes Neuronales Convolucionales** (CNN por sus siglas en inglés), son un tipo de red neuronal profunda diseñada específicamente para procesar datos no estructurados, como imágenes, vídeos o audio. Utilizan capas convolucionales que aplican filtros o *kernels* deslizantes sobre la entrada para capturar características locales y espaciales [37], produciendo **feature maps**. En las capas menos profundas de la red, las neuronas aprenden a identificar patrones simples como bordes o texturas, mientras que en las últimas capas se detectan estructuras complejas (figura 2.13).

Principios

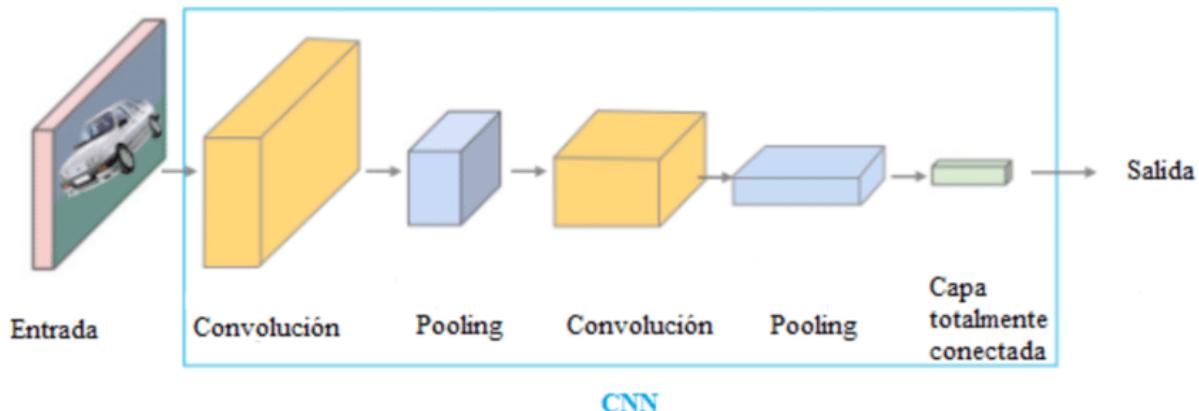


Figura 2.14. Estructura de una CNN [38].

Las CNN nacen como alternativa a las redes densas o *Fully Connected* en inglés (FC), muy costosas computacionalmente e ignorantes del contexto espacial. Estos inconvenientes radican en que, en la capa de entrada, se realiza una operación de *Flatten*⁷, donde cada neurona procesa el valor de un único píxel.

En consecuencia, las CNN (figura 2.14) proponen una solución mucho más eficiente donde un *kernel*, de menor tamaño que la entrada, se mueve por la imagen (figura 2.15) a pasos de tamaño s (*stride*), con un relleno en los bordes p (*padding*). Ajustando estos dos hiperparámetros y el número de filtros n_f , se pueden controlar las dimensiones de la salida de la capa, para reducir o condensar la información de la entrada en representaciones mucho más profundas (o con más canales).

⁷La operación de *Flatten* consiste en transformar una matriz o tensor multidimensional en un vector unidimensional, conservando el orden de los elementos.

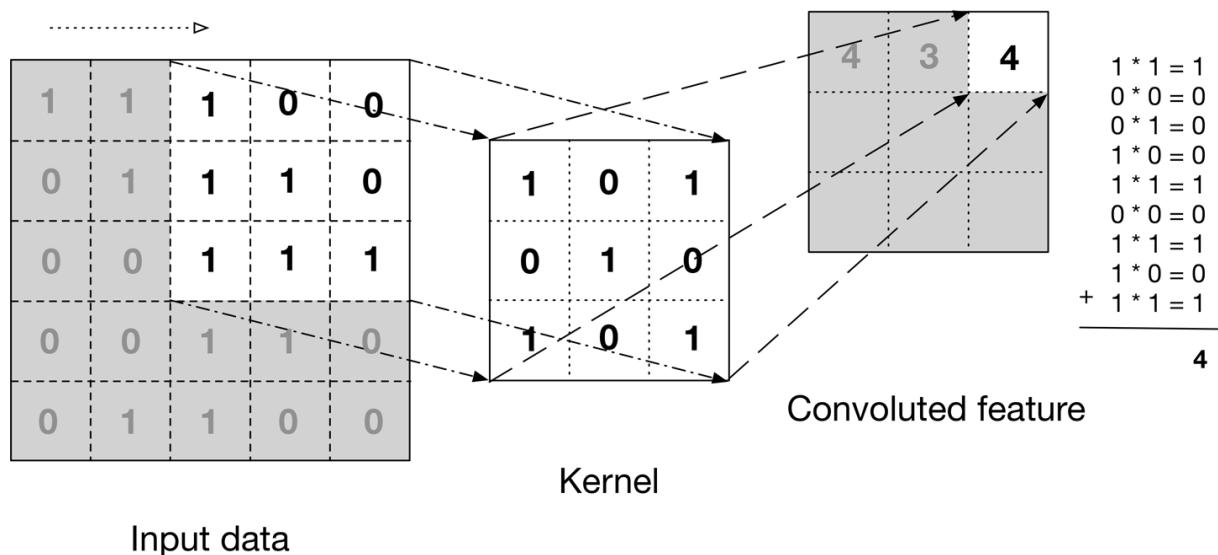


Figura 2.15. Cálculos detallados de la operación convolucional entre datos de entrada de 5x5 y un *kernel* de 3x3, con $s = 1$. [39].

Esta relación dimensional entre la entrada de tamaño (n, n, n_c) y la salida, viene dada por la expresión (2.13). El operando $*$ simboliza la convolución y las dimensiones del kernel se describen como (f, f, n_c) , siendo n_c el número de canales de la entrada.

$$(n, n, n_c) * (f, f, n_c) = \left(\frac{n + 2p - f}{s} + 1, \frac{n + 2p - f}{s} + 1, n_f \right) \quad (2.13)$$

Con el cometido de obtener una representación concentrada e informativa de la naturaleza de la imagen que permita reducir el coste computacional, se busca minorar el alto y ancho de los *feature maps* en cada bloque convolucional. Las capas de *pooling* o *downsampling* posibilitan esto mediante distintas técnicas de reducción que no utilizan parámetros entrenables [40, 41]. De este modo, en las capas finales no supone tan costoso incluir una capa FC para clasificar la imagen de entrada sin tener que renunciar a un rendimiento competente.

ResNet

Una de las arquitecturas convolucionales más relevantes es la **ResNet** (figura 2.16), propuesta por Kaiming He en 2015 [42]. Tiene como característica principal la introducción de conexiones residuales o *skip connections* en inglés, que permiten que la información fluya entre capas separadas de manera inmediata, facilitando el entrenamiento de redes muy profundas. Esto era difícil de lograr con métodos anteriores debido a la pérdida de información y la dificultad para propagar los gradientes a través de tantas capas.

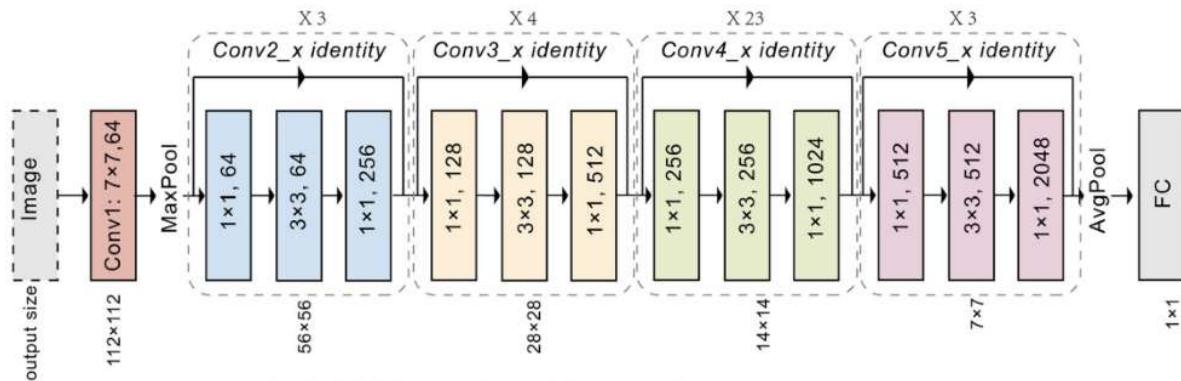


Figura 2.16. Estructura de la ResNet34 [43].

Más en detalle, cada bloque residual consta de una o varias capas convolucionales seguidas de una conexión residual que suma o concatena la entrada del bloque a la salida del mismo. La lógica detrás de esta estructura se basa en permitir que los gradientes fluyan libremente por donde la red estime conveniente.

Como las capas intermedias usan ReLU como activación, la probabilidad de que en capas más profundas el gradiente tienda a 0 es relativamente elevada. De esta manera, la red tiene muchas dificultades en sus últimas capas para aprender incluso funciones sencillas como la de identidad. A esto se le conoce como **desvanecimiento del gradiente** [44], un problema muy común en redes profundas o con exceso de regularización, que dificulta en gran medida el aprendizaje. Las *skip connections* funcionan tan bien porque pueden aprender funciones sencillas de manera inmediata provenientes de capas anteriores.

En la ecuación(2.14) se muestra la facilidad con la que una capa $l + 2$, cuya activación lineal $z^{[l+2]}$ es 0, conectada mediante una *skip connection* a la capa l , puede aprender la función de identidad de dicha capa con un solo paso de gradiente:

$$\begin{aligned} a^{[l+2]} &= f(z^{[l+2]} + a^{[l]}) = f(a^{[l]}) \\ a^{[l+2]} &= a^{[l]} \end{aligned} \tag{2.14}$$

Esto permite que, en el caso peor, sea equivalente a una red plana sin conexiones residuales, presentando una gran posibilidad de mejorar el rendimiento del modelo. De hecho, esta estructura residual ha demostrado mejorar significativamente el desempeño en tareas de visión por computadora, como clasificación de imágenes y detección de objetos, estableciendo a ResNet como una de las arquitecturas más influyentes y efectivas en el campo del aprendizaje profundo. En la figura 2.17 se muestra una comparación de la pérdida de entrenamiento en redes profundas, a medida que el número de capas aumenta, entre una red plana y ResNet.

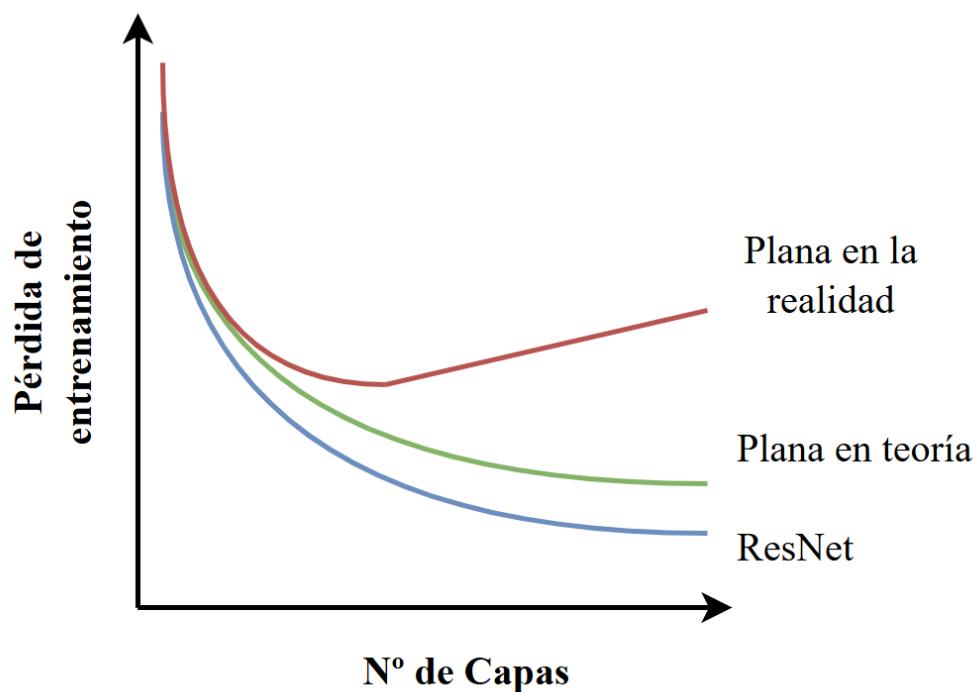


Figura 2.17. Pérdida de entrenamiento en relación al número de capas de una red plana y ResNet.

2.2.3. Transformers

Los **Transformers**, introducidos por Vaswani et al. en 2017 [45], son una arquitectura neuronal diseñada para procesar datos secuenciales, como texto o series temporales, con eficiencia y precisión. Incorporan el novedoso concepto de la atención, representada de múltiples formas por diferentes mecanismos, y popularizan el *transfer learning* (Apartado 2.3.1). Esta arquitectura (figura 2.18) ha revolucionado el campo del procesamiento del lenguaje natural (NLP), impulsando avances significativos en tareas como la traducción automática y el resumen de textos, entre otras.

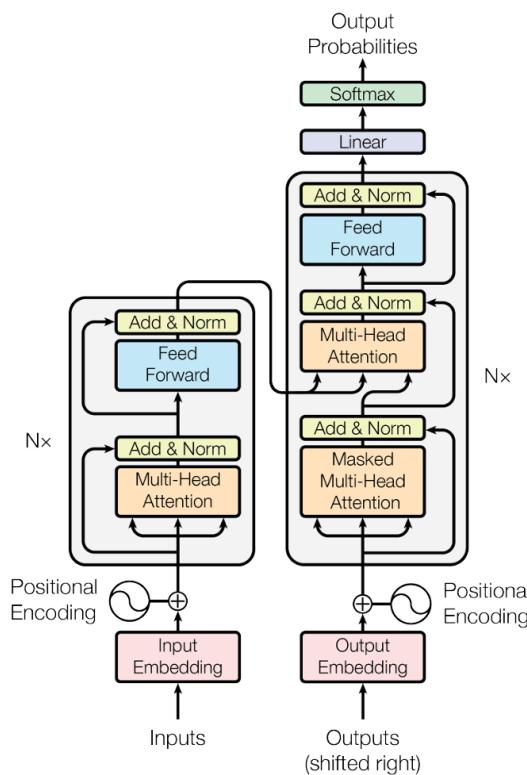


Figura 2.18. Arquitectura de un Transformer.

Tokens y Word Embeddings

En el ámbito del NLP, para conseguir una interpretación de una palabra o una parte de ella (a partir de ahora, solo se hablará de palabras por simplificar) que refleje adecuadamente su naturaleza, se utilizan técnicas de tokenización y *embedding*. Estas metodologías se complementan para asignar a cada pala-

bra un identificador que la representa, llamado **token**, que se corresponderá con un vector numérico o **word embedding** [46]. El grueso de los tokens sobre los que se entrena el modelo componen el vocabulario del mismo.

Estos *word embeddings* permiten aprender características intrínsecas de las palabras, analogías lógicas entre ellas y embedirlas a un punto del espacio vectorial. Aquellas palabras cercanas entre sí reflejan una relación más estrecha que aquellas separadas. De este modo, las diferencias entre dos pares de vectores que representen la misma relación lógica, tendrán un valor parecido. Por ejemplo (figura 2.19), si se quiere hallar qué palabra es a "Rey" lo que "Mujer" es a "Reina", se buscará encontrar la palabra w tal que se maximice la similitud entre $e_w - e_{Rey}$ y $e_{Mujer} - e_{Reina}$, donde e representa un *word embedding*.

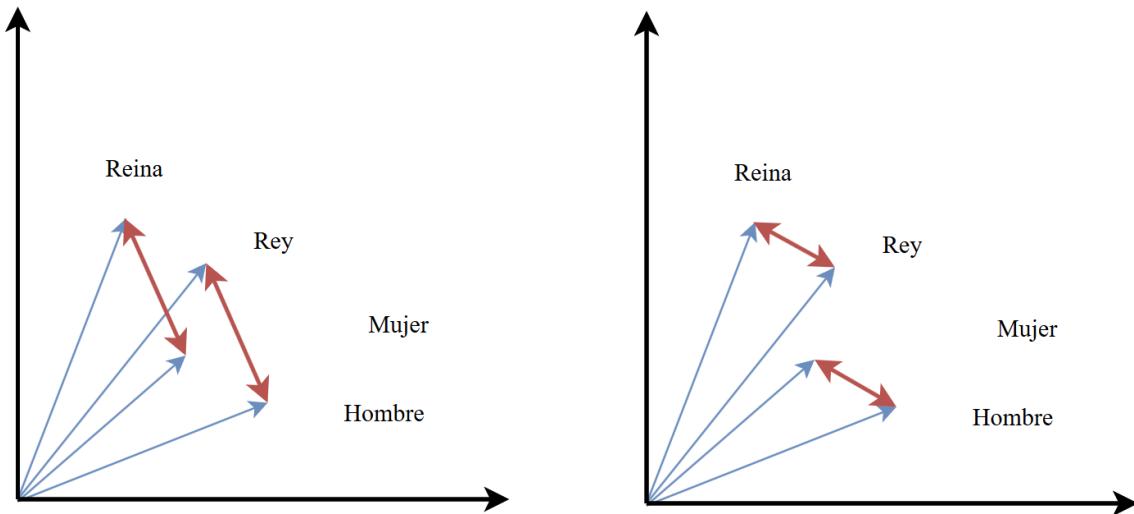


Figura 2.19. Analogía lógica con *word embeddings*.

Esta semejanza se calcula con la similitud coseno, expresada en la ecuación (2.15):

$$\text{sim}(e_w, e_{Mujer} - e_{Reina} + e_{Rey}), \quad \text{donde} \quad (2.15)$$

$$\text{sim}(u, v) = \frac{u \cdot v}{\|u\|_2 \cdot \|v\|_2} \quad (2.16)$$

La manera en la que se consigue que los *word embeddings* tengan información posicional dentro de la frase o texto de entrada, es mediante codificaciones posicionales o *positional encoding*. Tales vectores se suman a los *word embeddings* de entrada para que el modelo pueda distinguir la posición relativa de cada palabra en la secuencia, dando lugar a los *positional embeddings*.

Los *positional embeddings* se definen usando funciones seno y coseno de diferentes frecuencias en las ecuaciones (2.17) y (2.18), respectivamente, donde pos es la posición de la palabra en la secuencia e i es la dimensión:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (2.17)$$

$$\text{PE}(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (2.18)$$

Estructura

Los transformers siguen una estructura *encoder-decoder* que, a diferencia de las arquitecturas recurrentes anteriores (*Recurrent Neural Networks* [47], *Gated Recurrent Unit* [48] o *Long Short Term Memory* [49]), permite paralelizar el entrenamiento. Este factor es clave, ya que permite que estos modelos puedan recibir cantidades de datos enormes sin renunciar a un gran rendimiento. Los dos componentes principales que conforman su estructura son:

- **Encoder:** Se encarga de transformar los *positional embeddings* que recibe como entrada en representaciones intermedias y finales llamados estados ocultos o *hidden states*. La relevancia de este nuevo concepto reside en la capacidad de prestar atención a las partes más importantes de su contexto y entender matices sutiles inherentes al lenguaje. Esto se consigue gracias a una estructura formada por los mecanismos de atención(descritos más adelante), capas de normalización y capas *feed forward* (FF) que aplican una transformación no lineal a cada posición de la secuencia. Los encoders están formados por una pila de N bloques (figura 2.20) de codificación idénticos.

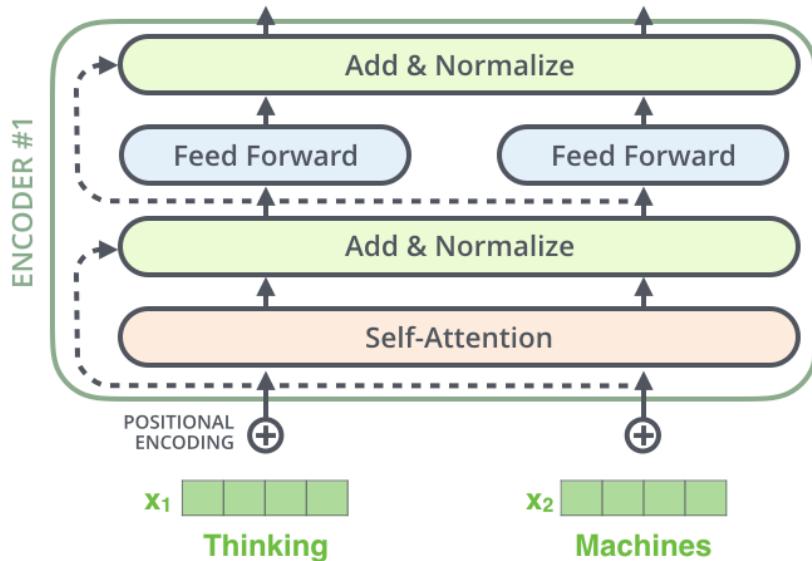


Figura 2.20. Bloque encoder de un Transformer [50].

- **Decoder:** Genera la secuencia de salida a partir de los *hidden states* producidos por el *encoder*. Al igual que estos últimos, también están compuestos por M bloques iguales (figura 2.21), aunque presentan una estructura algo más compleja. Esto ocurre porque en el último bloque, los *decoders* tienen una o varias capas densas seguidas de una activación softmax⁸, con tantas clases como palabras hay en el vocabulario. El vector resultante muestra la probabilidad de cada *token* de ser la siguiente palabra a predecir en la secuencia. Se suele elegir el *token* con mayor valor o un *token* aleatorio entre los K *tokens* de mayor probabilidad. De este modo se evita que, para la misma entrada, la predicción sea siempre la misma.

⁸La función de activación softmax es utilizada para convertir un vector de valores arbitrarios en una distribución de probabilidad.

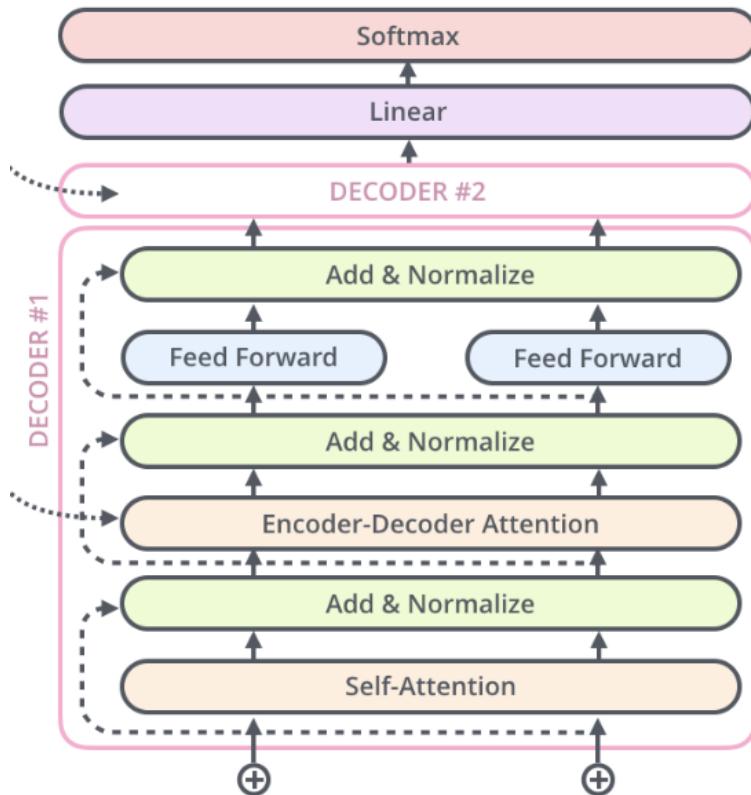


Figura 2.21. Bloques decoder de un Transformer [50].

Sin embargo, no todos los transformers presentan estos dos componentes. Existen arquitecturas *encoder-only*, utilizadas para generar representaciones de la entrada, muy populares en tareas de clasificación de texto, como *Bert* [51]. Por otro lado, las *decoder-only* son muy usadas en generación de lenguaje natural, siendo su principal exponente la familia GPT y el archiconocido *ChatGPT* [52], mientras que las *encoder-decoder* se utilizan en problemas de secuencia a secuencia, como la traducción automática. Destacan entre ellos, *MarianMT* [53] y *Bart* [54].

Mecanismos de Atención

El componente distintivo de los Transformers es el mecanismo de *self-attention*, que permite al modelo evaluar la importancia de cada palabra en la secuencia con respecto a las demás. Esto se logra mediante el entrenamiento de

tres vectores: *Query* (**Q**), *Key* (**K**) y *Value* (**V**).

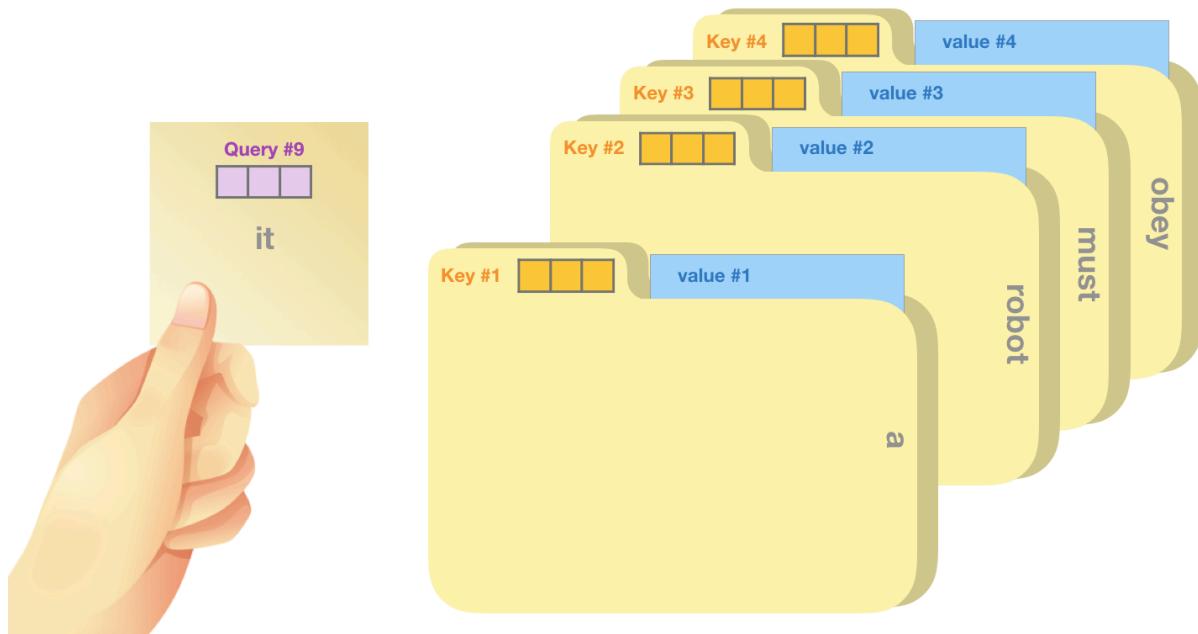


Figura 2.22. Analogía del mecanismo de *self-attention* [55].

Siguiendo la metáfora de la figura 2.22, se puede entender la *Query* como un *post-it* donde tienes apuntado la información que quieras encontrar, mientras que las *Keys* se corresponderían con las etiquetas de las carpetas del archivador. Cuando la etiqueta coincide con la información del *post-it*, sacamos el contenido de esa carpeta, el *Value*. En nuestro caso, nos interesan todos los *Values* ponderados por la similitud de nuestra *Query* y sus *Keys* asociadas.

Matemáticamente esto se traduce en calcular el producto escalar entre Q y K y aplicarle una operación *softmax* al resultado. Posteriormente, se divide entre la raíz cuadrada de la dimensión d_k del vector K para facilitar la estabilización de los gradientes y se multiplica por su V correspondiente. En el contexto de una oración, en cada bloque de atención se realiza este procedimiento, obteniendo las correlaciones contextuales entre todas las palabras.

Para agilizar los cálculos, los vectores se agrupan en matrices, cuya multiplicación requiere de aplicar la traspuesta a una de ellas para no tener problemas dimensionales, en este caso a K . Su implementación vectorizada se muestra en la ecuación (2.19):

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{Q \cdot K^T}{\sqrt{d_k}} \right) V \quad (2.19)$$

Si repetimos este mismo procedimiento N veces para cada bloque de atención, podremos tener una representación más completa de la oración, sin estar supeditado al sesgo que puedan introducir unas matrices de atención particulares. De esta manera, nace el término *multi-head self-attention* (MSA), que permite a estos modelos hayar las dependencias directas dentro de la secuencia.

Vision Transformer

El **Vision Transformer** (ViT) [56] representa una innovación en el campo del procesamiento de imágenes y *computer vision*, al introducir los transformers como alternativa a las CNNs tradicionales. Tienen una arquitectura *encoder only* y son capaces de aplicar la atención para tener una visión más general y aprender relaciones entre partes alejadas de las imágenes.

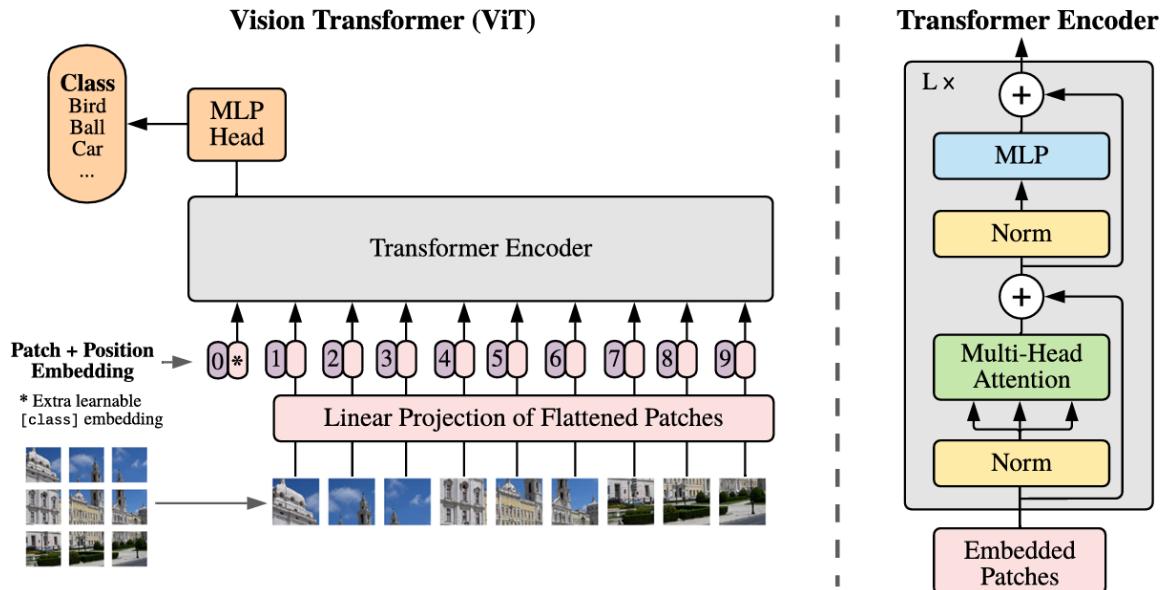


Figura 2.23. Estructura del Vision Transformer [56].

Como se muestra en la figura 2.23, el modelo comienza dividiendo la imagen de entrada en parches cuadrados de tamaño fijo (por ejemplo, de 16x16 píxeles como se propone en el paper original) que simularán una secuencia. A cada parche se le aplica una operación de *Flatten*, formando vectores, que serán posteriormente proyectados linealmente, a través de una o varias capas densas para obtener los *patch embeddings*. Al igual que en los transformers originales, se les añade un *positional encoding* para aportar contexto espacial, obteniéndose una secuencia de N *patch embeddings*, donde N es el número de parches en la imagen. Además, en tareas de clasificación, se suele incorporar un parche especial entrenable que toma el nombre de *Class token*, resultando en una secuencia de longitud $N + 1$.

Por último, estos *embeddings* pasan por los distintos bloques del *encoder* y se proyecta su salida a través de un perceptrón multicapa (MLP) [57], que generalmente consta de dos capas lineales separadas por una función de activación no lineal, en este caso GELU. Si a este resultado se le pasa por una función *softmax*, se puede clasificar la imagen de entrada por clases.

En caso de ser entrenados con grandes conjuntos de datos, los modelos ViT han demostrado que pueden superar a los modelos CNN tradicionales, debido a su capacidad para aprender y generalizar sin necesidad de los sesgos inductivos inherentes a las CNNs, como la equivariancia de traslación y la localidad [58]. Sin embargo, las grandes variaciones en la escala de entidades visuales (objetos, características o elementos) y la alta resolución de los píxeles en las imágenes, requieren de una gran capacidad computacional.

Con la intención de solventar este problema y reducir la complejidad del ViT sin tener que renunciar a un gran rendimiento, aparece el Swin Transformer [59].

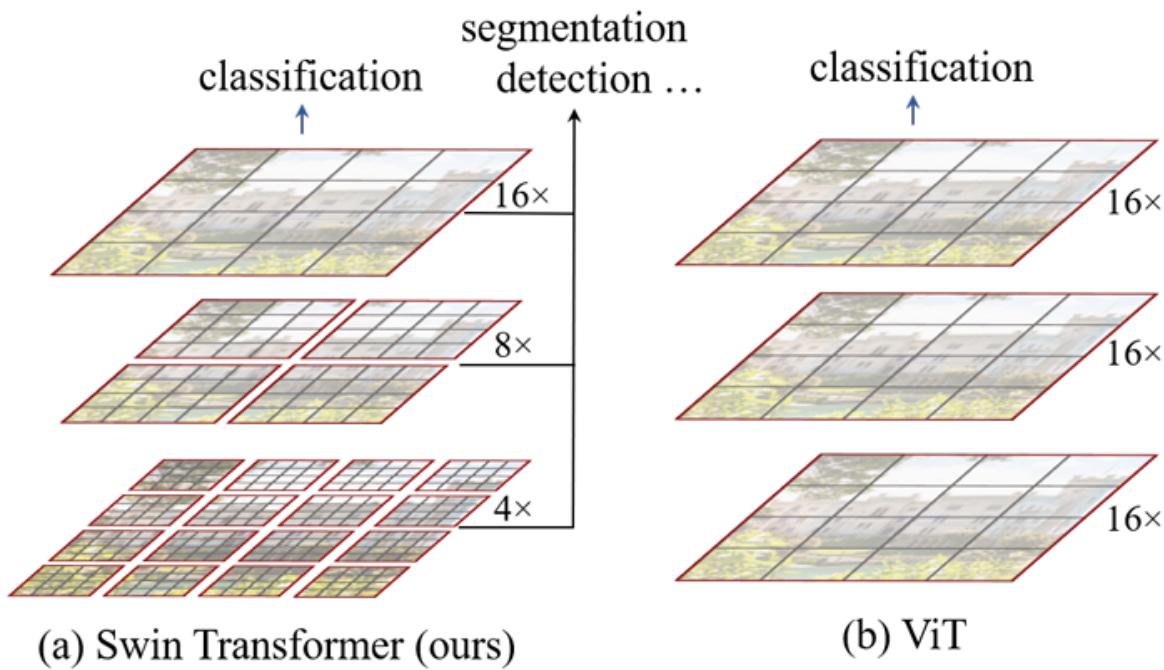


Figura 2.24. Comparación Swin Transformer y Vision Transformer [59].

El **Swin Transformer** (figura 2.24) propone sustituir el mecanismo de atención *vanilla* por el método *window-based self-attention*. Básicamente, este consiste en agrupar los parches por ventanas y calcular la *self-attention* dentro de ellas y no de manera global, limitando el cálculo a áreas locales y reduciendo la complejidad computacional (en concreto, se reporta que consiguen una complejidad de orden lineal). El número de parches que contienen dichas ventanas es directamente proporcional a la profundidad de las capas de la red. Esto es, con muchas ventanas pequeñas en las primeras capas y con pocas más grandes en las últimas.

Para garantizar que se presta atención a todas las posibles dependencias dentro de la imagen, independientemente de su posición, las ventanas se desplazan entre las capas de atención. Además, se crean conexiones cruzadas entre las capas, proporcionando una conectividad global eficiente que reduce la latencia en comparación con las ventanas deslizantes tradicionales [60, 61].

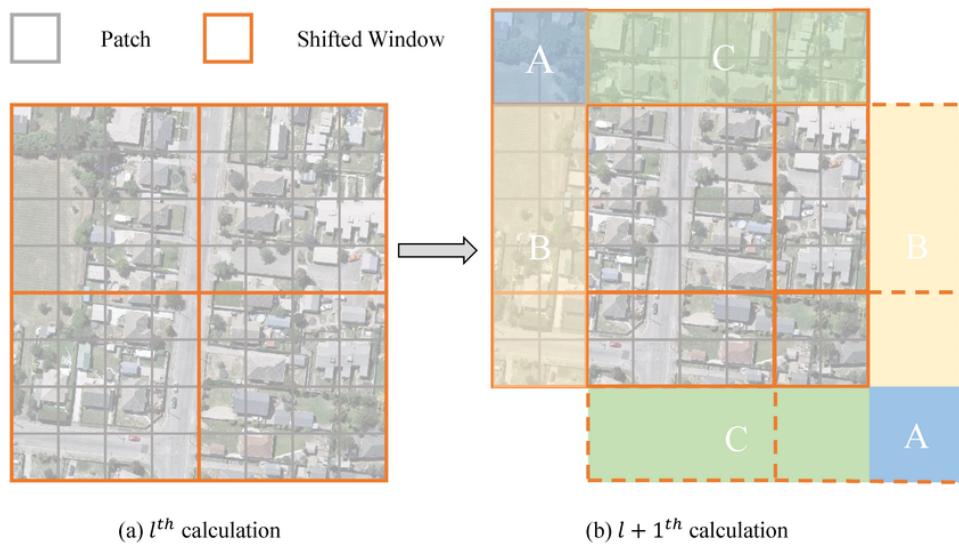


Figura 2.25. Desplazamiento de ventana de atención entre dos capas consecutivas [62].

Sin embargo, cuando una ventana se desplaza, es posible que una parte de ella se salga de los límites de la imagen. Para gestionar esto, se utiliza un enfoque de llenado simétrico conocido como *cyclic shift*, manteniendo así la integridad de las relaciones espaciales y contextuales capturadas por la ventana deslizante.

Como podemos observar en la figura 2.25, tras desplazar las ventanas dos parches hacia la derecha y dos parches hacia abajo, tres de ellas se salen de la imagen. Siguiendo la técnica de *cyclic shift*, el espacio *B* sobrante por la derecha se rellena con el espacio *B* libre de la izquierda, haciendo lo propio para *C* y *A*.

2.2.4. Segmentación de Imágenes

La **segmentación** es una tarea derivada de la clasificación que implica la división de la imagen en diferentes segmentos o regiones, las cuales se corresponden a distintos objetos o zonas de interés. Para ello, se clasifica cada píxel, de manera que aquellos pertenecientes a la misma clase forman parte de la misma entidad o categoría, como se muestra en la figura 2.26. Esta tarea se aborda frecuentemente utilizando **modelos generativos**, que son capaces de aprender las distribuciones de probabilidad de los datos y generar nuevos ejemplos similares a los observados en el conjunto de entrenamiento.

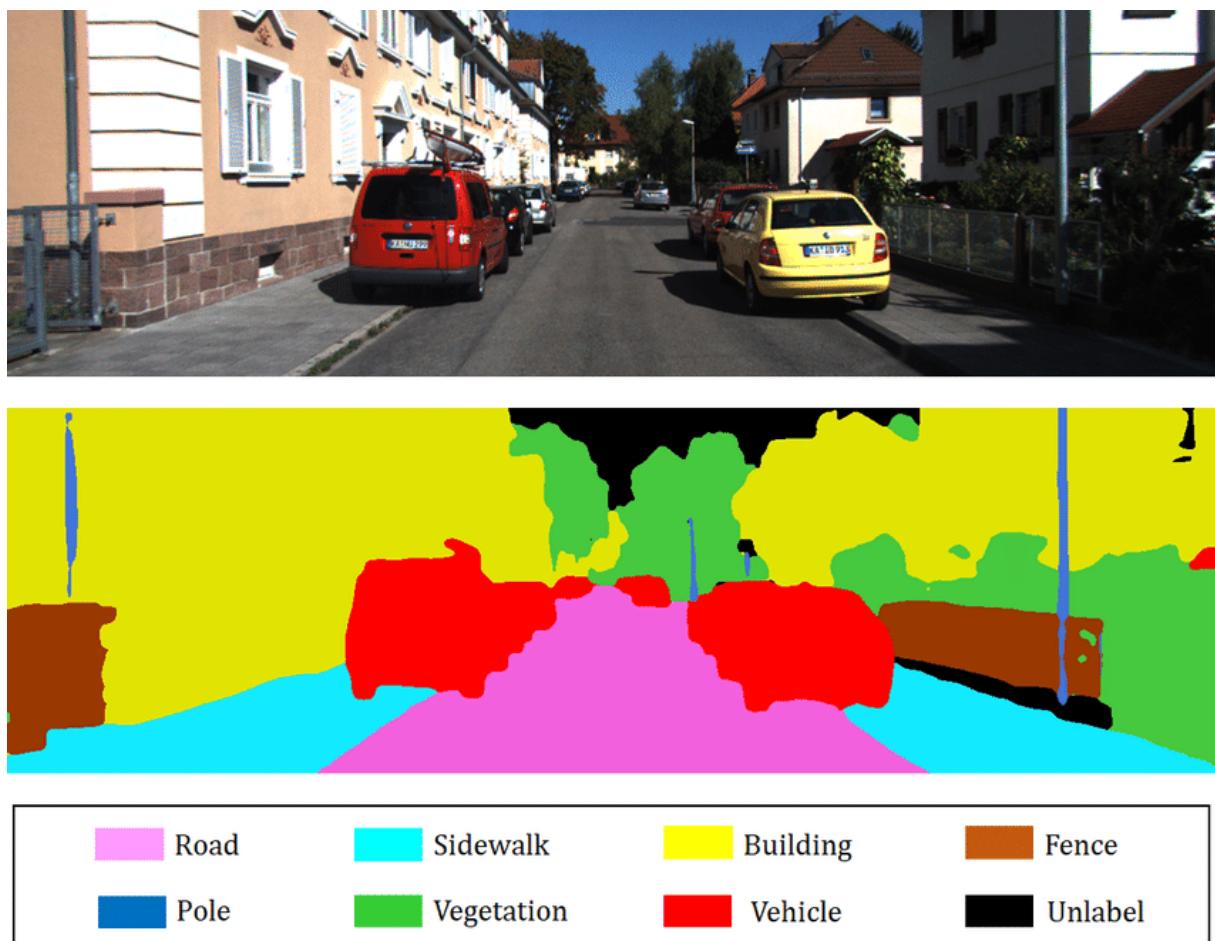


Figura 2.26. Ejemplo de segmentación de una imagen por clases [63].

Primeras Aproximaciones

Debido a que la tarea de predecir píxel a píxel toda la imagen sería extremadamente costosa e ineficaz, una primera aproximación para abordar este problema de manera intuitiva, sería dividir la imagen por ventanas [64]. En cada una de ellas, se clasificaría el píxel central, que ejerce de representante, y se le asignaría dicho valor a toda la ventana. Sin embargo, como se puede anticipar, esta solución es muy ineficiente, ya que no reutiliza las características compartidas entre ventanas. Es decir, no es capaz de usar la información espacial de manera efectiva y compartir el conocimiento entre ellas. Además, si se fija un tamaño de ventana demasiado pequeño, el modelo perderá la visión general de la imagen, mientras que si se establece muy alto, no sería capaz de hilar fino y comprender los detalles.

Con el objetivo de encontrar una alternativa, se han propuesto multitud de arquitecturas de redes neuronales especializadas en esta tarea. Una de las más relevantes es la red completamente convolucional (del término en inglés **Fully Convolutional Network**, FCN) [65] que, como su propio nombre indica, descarta las capas densas de salida de la arquitectura convolucional tradicional (figura 2.14). En su lugar incorpora capas de *upsampling*, y convoluciones transpuestas, que aumentan progresivamente la resolución de los *feature maps* hasta que su tamaño coincide con el de la entrada original. Esta estructura, mostrada en la figura 2.27, es de tipo encoder-decoder y recibe el nombre de **AutoEncoder (AE) convolucional**.

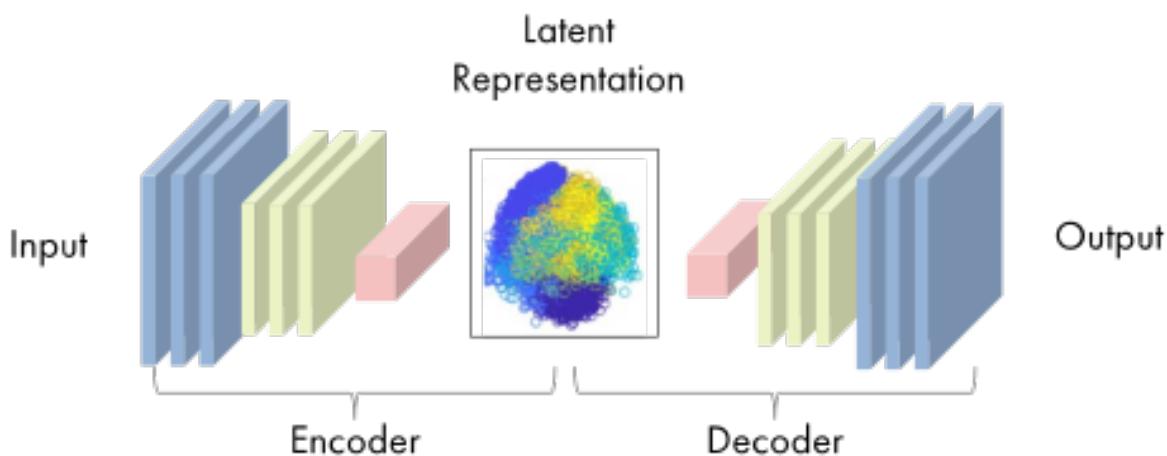


Figura 2.27. Estructura de un AutoEncoder convolucional [66].

El término **espacio latente** se refiere a un espacio abstracto entre el *encoder* y el *decoder*, donde se representan las características de los datos de entrada de manera compacta y significativa. Este concepto es fundamental para comprender la gran capacidad de los AE para trasladar la información de un dominio a otro y generar datos nuevos. Es el punto de la red donde la resolución de las imágenes se reduce al mínimo y su profundidad es máxima, es decir, donde hay más canales de información.

En concreto, existe un único espacio latente L con un mapeo no lineal desde la entrada X (expresión 2.20), y otro hacia la salida Y (2.21). El *encoder* transforma la representación de cada muestra en un "código" en el espacio latente, y el *decoder* es capaz de construir salidas a partir de él. Esto permite desmontar una red de este tipo y utilizar el *encoder* y el *decoder* por separado [67].

$$\text{encoder: } X \rightarrow L \quad (2.20)$$

$$\text{decoder: } L \rightarrow Y \quad (2.21)$$

U-Net

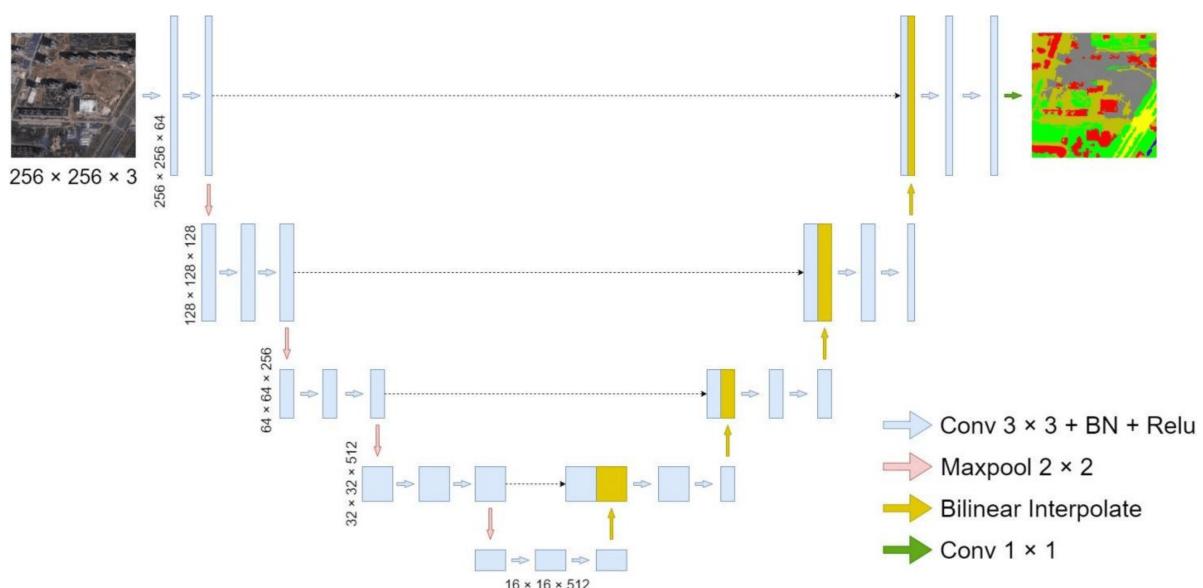


Figura 2.28. Arquitectura de la U-Net [68].

La **U-Net** es una arquitectura de red convolucional que ha demostrado ser extremadamente eficaz en tareas de segmentación de imágenes. Introducida por Ronneberger *et al.* en 2015 [69], su estructura en forma de "U" consta de un *encoder* y un *decoder* simétricos. Es decir, el número de bloques convolucionales de ambos componentes es idéntico y cada uno realiza el proceso inverso a su análogo. De este modo, si entre la segunda y la tercera capa del *encoder* se reduce la dimensión de la imagen de (128, 128, 128) a (64, 64, 256), entre la antepenúltima y la penúltima capa del *decoder* se aumentará de (64, 64, 256) a (128, 128, 128) (figura 2.28).

La principal diferencia que distingue a la U-Net de los AE tradicionales, es la incorporación de conexiones residuales entre capas equidimensionales. En ellas, tal y como se explicó previamente en el subapartado 2.2.2, se concatenan las salidas del *encoder* con las entradas correspondientes del *decoder*, permitiendo que la información se transfiera directamente entre ellas. Esto facilita la recuperación de detalles espaciales durante el proceso de decodificación, lo cual es crucial para la precisión en este tipo de tareas.

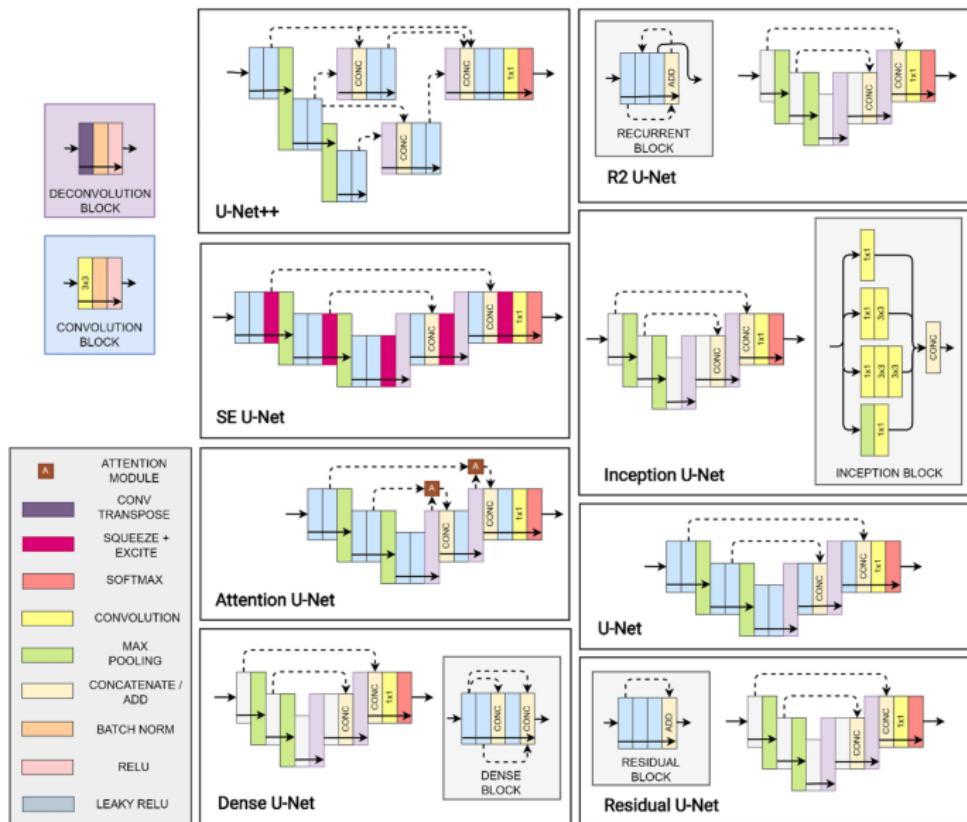


Figura 2.29. Variantes de la U-Net [70].

Debido a su gran repercusión, multitud de variantes se han implementado en los últimos años, con el objetivo de adaptar esta arquitectura a dominios concretos (figura 2.29). Cada una de ellas se caracteriza, o bien por la construcción de los bloques del *encoder* y del *decoder*, o por la manera de aplicar las *skip connections*. Esta gran versatilidad radica en la flexibilidad que otorga la red para comprimir la información relevante en el espacio latente, y adaptar el resto de la arquitectura en consecuencia. Muchas de estas adaptaciones van también enfocadas a reducir el coste computacional [71, 72] o a aplicar nuevas ideas que mejoren el rendimiento del modelo [73, 74].

2.3. Aprendizaje Auto Supervisado

En esta sección se definen los conceptos básicos sobre los que se apoya el paradigma de Aprendizaje Auto Supervisado, discutiendo sus diferentes tipos y aplicaciones en relación a su relevancia en el panorama actual del DL.

2.3.1. Pre-Entrenamiento, *Transfer Learning* y *Fine Tuning*

Uno de los efectos colaterales más positivos del éxito y popularización del ML y el DL en los últimos 15 años, es la democratización del conocimiento en el campo. De manera tangible, esto se traduce en la aparición de numerosos entornos como *Hugging Face* [75] (figura 2.30), donde los desarrolladores pueden compartir los parámetros de modelos ya entrenados, para que otros usuarios puedan aprovecharlos.



HUGGING FACE

Figura 2.30. Plataforma *Hugging Face*.

En este contexto, las técnicas de ***transfer learning*** y ***fine tuning*** son imprescindibles. En lugar de partir de pesos y *bias* inicializados aleatoriamente, se utiliza un modelo **pre-entrenado** sobre una tarea fuente, más amplia o general, para transferir su conocimiento a una nueva tarea objetivo, más espe-

cífica, en la que se suelen alterar las últimas capas. Por lo general, el *dataset* utilizado para la tarea fuente es mucho más grande que el de la tarea objetivo. De esta manera, se evita desperdiciar información valiosa cuya obtención ha consumido numerosos recursos y tiempo significativo de entrenamiento. Además, se fomenta la posibilidad de construir sobre el trabajo de otros, lo cual es altamente beneficioso y productivo.

Aunque en la literatura actual estos términos son prácticamente intercambiables, no se refieren a lo mismo exactamente. En concreto, *transfer learning* hace referencia a "congelar" en el modelo objetivo todos los pesos transferidos desde el modelo fuente, exceptuando a los de las últimas capas (se corresponden con las *Head* de la figura 2.31). Estos últimos se entrenarían para ajustar la red a la tarea final. Por otra parte, pese a que también se alteran las últimas capas, en el *fine tuning* se reentrenan todos los pesos del modelo para intentar garantizar un mejor ajuste.

Transfer Learning

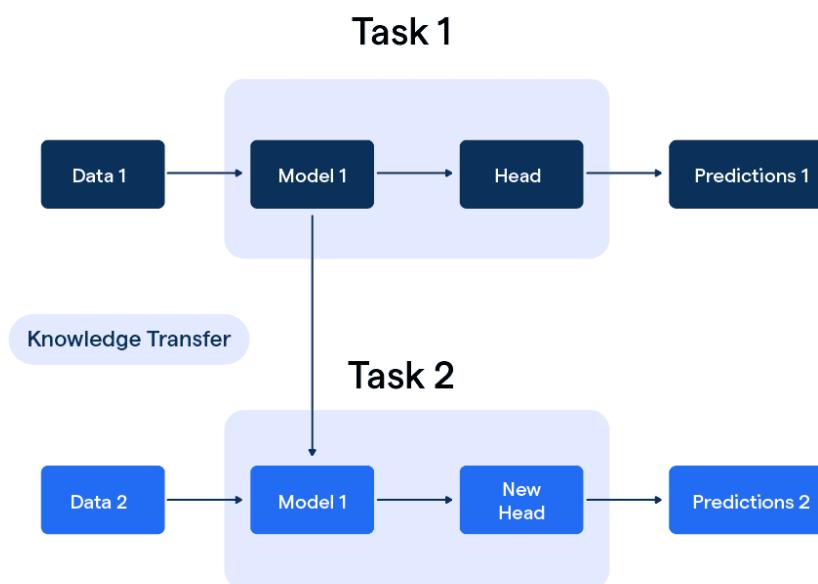


Figura 2.31. Transfer Learning [76].

En ocasiones, es interesante pre-entrenar un modelo sobre el que luego hacer *fine tuning*, para conseguir que la red aprenda la naturaleza de los datos de entrada y aporte un valor real que de otra manera no se podría haber obtenido. Para que ello ocurra, se deben cumplir las siguientes condiciones:

1. Las tareas fuente y objetivo aceptan el mismo tipo de datos.
2. Se disponen de muchos más datos para la tarea fuente que para la objetivo.
3. Las características de bajo nivel extraídas en la tarea fuente pueden ser de ayuda en la tarea objetivo.

2.3.2. Paradigmas de Aprendizaje

Dentro del *Deep Learning* se pueden diferenciar tres principales filosofías o paradigmas de aprendizaje, según su relevancia históricamente (figura 2.32). Cada uno de ellos se refiere a un marco o enfoque particular para el entrenamiento de modelos, determinado por la naturaleza de los datos y cómo se optimiza la red para mejorar su desempeño en tareas específicas:

- **Aprendizaje Supervisado (SL):** En este paradigma, los modelos se entrenautilizando un conjunto de datos etiquetados, donde cada ejemplo de entrenamiento tiene asociada una etiqueta o **ground truth** asignada manualmente por un supervisor. El objetivo es que el modelo aprenda a predecir la etiqueta correcta para nuevos ejemplos no vistos. El rendimiento de estos modelos está limitado por la cantidad y la calidad de las etiquetas disponibles, siendo crucial para la precisión de las predicciones en tareas como clasificación y regresión [77].
- **Aprendizaje No Supervisado (UL):** A diferencia del aprendizaje supervisado, este modelo trabaja con datos no etiquetados. El objetivo es identificar patrones y estructuras subyacentes en los datos, utilizando técnicas como el *clustering* (agrupamiento de datos según similitudes) y la reducción de dimensionalidad para simplificar datos manteniendo su información esencial [78]. Este enfoque se aplica en problemas como segmentación de clientes y detección de anomalías.
- **Aprendizaje por Refuerzo (RL):** Se basa en la idea de un agente que aprende a tomar decisiones secuenciales para maximizar una recompensa acumulada. El agente interactúa con un entorno y recibe retroalimentación en forma de recompensas o castigos [79]. Este enfoque es muy común en la robótica, los juegos y la optimización de sistemas complejos.

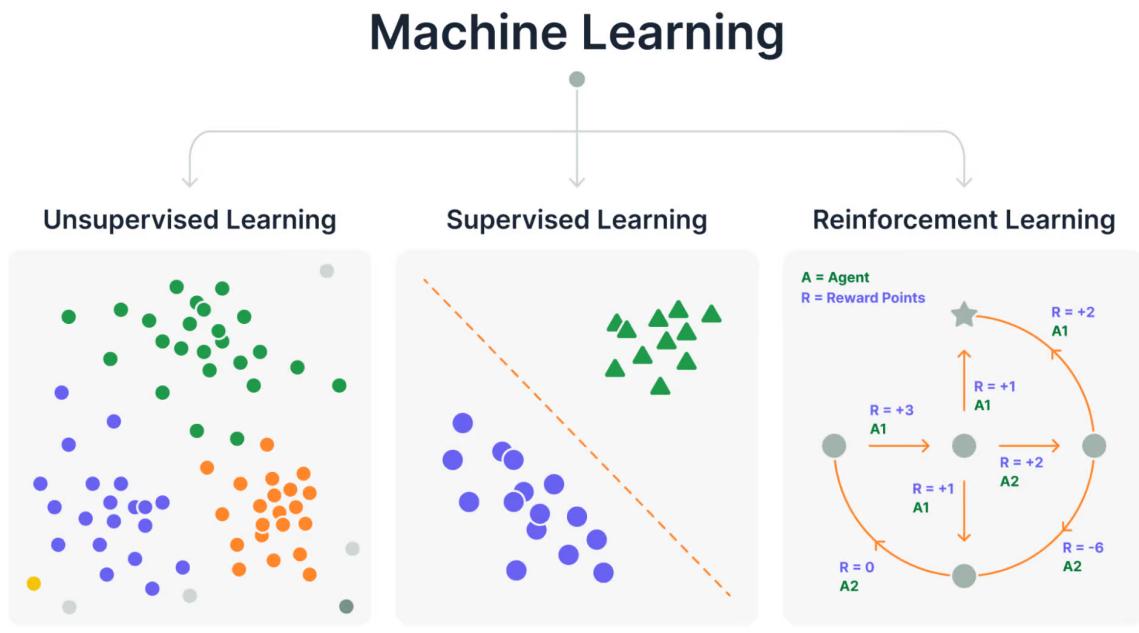


Figura 2.32. Paradigmas de aprendizaje [80].

Con el objetivo de enmendar los problemas de escalabilidad y rendimiento consustanciales a estos paradigmas, nace el **Aprendizaje Auto Supervisado** (SSL, de sus siglas en inglés). El SSL puede ser comparado con el proceso de aprendizaje de un niño pequeño: cuando se enfrenta a una nueva tarea, como atarse los cordones de los zapatos por primera vez, utiliza su conocimiento general del mundo. Incluso antes de aprender específicamente cómo atar los cordones, el niño tiene una comprensión básica y subconsciente de cómo funcionan las cosas en su entorno. Sabe, a base de prueba y error, que los objetos son tangibles y que ciertas acciones no llevarán a consecuencias extremas, como desintegrar su pie. De manera similar, un modelo de SSL pasa por un proceso de pre-entrenamiento supervisado que le prepara para enfrentarse a tareas nuevas, donde poco a poco va comprendiendo la naturaleza intrínseca de los datos. En esta fase previa de aprendizaje "global", se entran de manera supervisada utilizando etiquetas que, en lugar de ser extraídas manualmente, son obtenidas en tiempo real de los propios datos. De este modo, antes de recibir tareas específicas, el modelo ha aprendido representaciones generales útiles que le permiten adaptarse y aprender eficazmente nuevas tareas cuando se le presentan [81].

Y. LeCun

How Much Information is the Machine Given during Learning?

- ▶ “Pure” Reinforcement Learning (**cherry**)
- ▶ The machine predicts a scalar reward given once in a while.
- ▶ **A few bits for some samples**
- ▶ **Supervised Learning (**icing**)**
 - ▶ The machine predicts a category or a few numbers for each input
 - ▶ Predicting human-supplied data
 - ▶ **10→10,000 bits per sample**
- ▶ **Self-Supervised Learning (**cake génoise**)**
 - ▶ The machine predicts any part of its input for any observed part.
 - ▶ Predicts future frames in videos
 - ▶ **Millions of bits per sample**



© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future 59

Figura 2.33. LeCake: Metáfora de la tarta de Yann LeCun para explicar el SSL [82].

Su origen se remonta a un artículo publicado en los años 50 por Wilson L. Taylor [83], donde utilizaba la predictibilidad de las palabras en un texto para calcular la legibilidad del mismo. Sin embargo, no sería hasta finales de 2016 cuando se popularizaría en el ámbito del DL, gracias, en parte, a Yann LeCun, una de las figuras más influyentes en el campo de la IA [84]. Este utilizó una metáfora visual de una tarta para justificar su importancia, en la que la guinda, el glaseado y el bizcocho se corresponden con el potencial de aprendizaje del RL, el SL y el SSL, respectivamente (figura 2.33).

2.3.3. Aplicación en Texto e Imágenes

La aplicación del paradigma auto supervisado a texto usando Transformers es bastante sencilla o directa, donde el pre-entrenamiento está enfocado en predecir un *token* o un conjunto de *tokens* sabiendo el contexto de la oración (figura 2.34). Esos *tokens* constituyen las etiquetas y son elegidos aleatoriamente en tiempo de ejecución (se suelen *maskear* alrededor del 15%). Los grandes modelos de lenguaje que existen en la actualidad, como los previamente mencionados GPT [52], Bert [51], Gemini [85] o LLaMa [86] han demostrado que, si son entrenados con cantidades de texto enormes, producen resultados espectaculares. Esta necesidad de un volumen tan extenso de datos responde a la naturaleza de los transformers y del SSL.

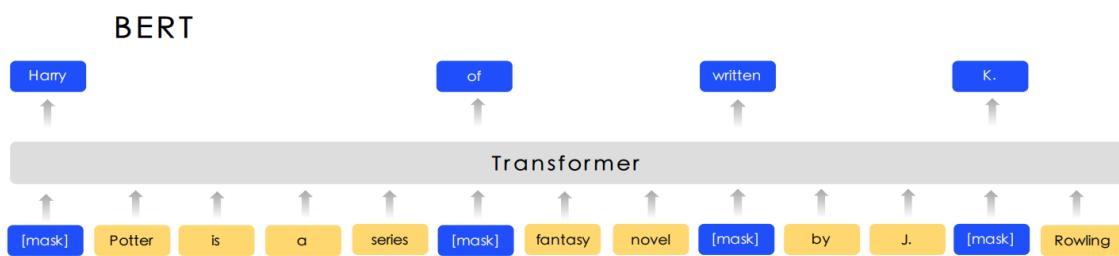


Figura 2.34. Enmascarado de *tokens* durante el entrenamiento de BERT [87].

Lamentablemente, en el campo del **computer vision** no es una tarea tan trivial aplicar estas técnicas. Las imágenes de manera natural presentan una complejidad mucho mayor que los archivos de texto, lo que dificulta su procesamiento, su almacenamiento y la elección de metodologías acordes a sus características. Considerando la gran cantidad de datos que requieren estos modelos auto supervisados, se hace evidente que se necesitan respuestas a todos estos problemas si se quieren obtener resultados a la altura.

Como solución, se utilizan técnicas de **Aumento de Datos** o *data augmentation* en inglés, cuyo objetivo es aumentar sintéticamente el conjunto de imágenes de entrenamiento. Para ello, se busca obtener imágenes nuevas con cierta variabilidad, pero que mantengan la distribución original [88, 89]. Esto se consigue aplicando ligeras alteraciones en su color, textura, forma o composición, mediante técnicas como las mostradas en la figura 2.35.

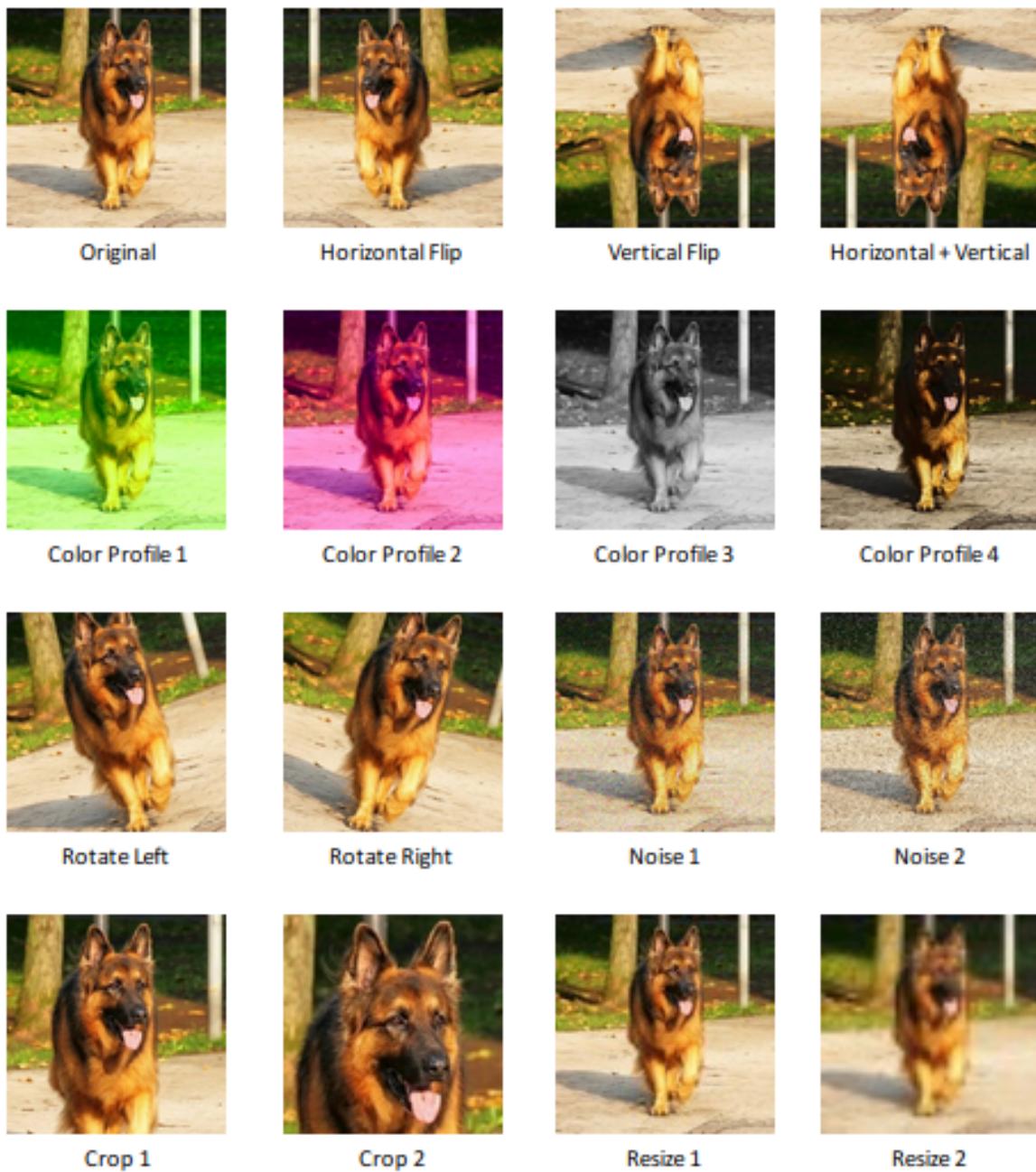


Figura 2.35. Diferentes técnicas de Aumento de datos [90].

2.3.4. Discriminativo vs Generativo

Los métodos de SSL aplicados a imágenes se pueden clasificar en dos grandes categorías, dependiendo del enfoque utilizado durante su pre-entrenamiento:

- **Enfoque Discriminativo:** Se centra en identificar y aprender, a partir de patrones internos dentro de los datos, mediante tareas auxiliares que promueven la discriminación entre diferentes entradas, como predecir la rotación [91] o el contexto de las imágenes [92, 93]. Dentro de esta rama se encuentran también los métodos de **Aprendizaje Contrastivo** donde el modelo aprende a diferenciar entre pares de datos. De este modo, se crean pares positivos de imágenes aumentadas sobre la misma entrada y pares negativos de imágenes aumentadas sobre diferentes entradas. El modelo se entrena para minimizar la distancia entre las representaciones de los pares positivos y maximizar la distancia entre los pares negativos, como en MoCo [94] o SimCLR [95].

Por otro lado, arquitecturas como **SimSiam** [96] proponen no utilizar pares negativos, como se muestra en la figura 2.36. De esta forma, el par de imágenes es procesado por un mismo *encoder*, pero sólamente la representación resultante de una de ellas (imagen *A*) sirve como entrada a un MLP. A la otra imagen (*B*) se le aplica una operación de *stop-grad*⁹ y se intenta maximizar su similitud con la proyección resultante de *A*.

⁹La operación de *stop-grad* se refiere a detener la retropropagación durante el entrenamiento de una red neuronal, lo cual se utiliza para evitar que ciertas capas o parámetros reciban actualizaciones de gradiente.

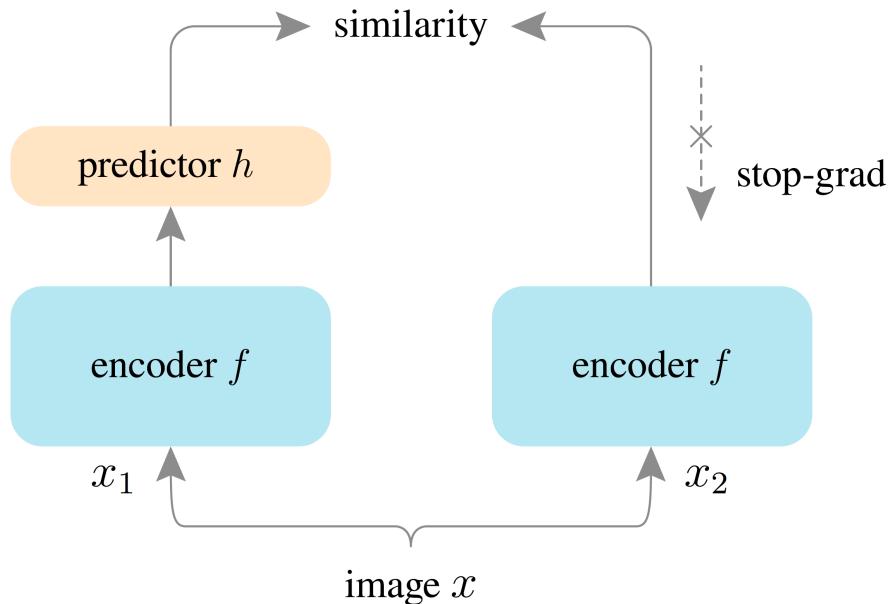


Figura 2.36. Arquitectura de SimSiam [96].

- **Enfoque Generativo:** Se pretende que el modelo aprenda a generar o reconstruir partes de los datos de entrada, para entender su estructura subyacente. Este enfoque utiliza tareas como la predicción enmascarada, previamente mencionada con el modelo de NLP Bert, donde se ocultan partes de la entrada y el modelo se entrena para predecirlas.

En este contexto, los Autoencoders Enmascarados (MAE), propuestos por Kaiming He en 2021 [97], gozan de especial relevancia. Utilizan como base los componentes del ViT (*encoder* y *decoder*), dividiendo las imágenes de entrada en parches y enmascarando el 75 % de ellos. Posteriormente, el 25 % restante se pasa por el *encoder* y se reorganizan de manera que no se pierde la información espacial de los mismos. Por último, el *decoder* se encarga de reconstruir la imagen de entrada a partir de estas representaciones intermedias. La estructura general de este proceso se muestra en la figura 2.37.

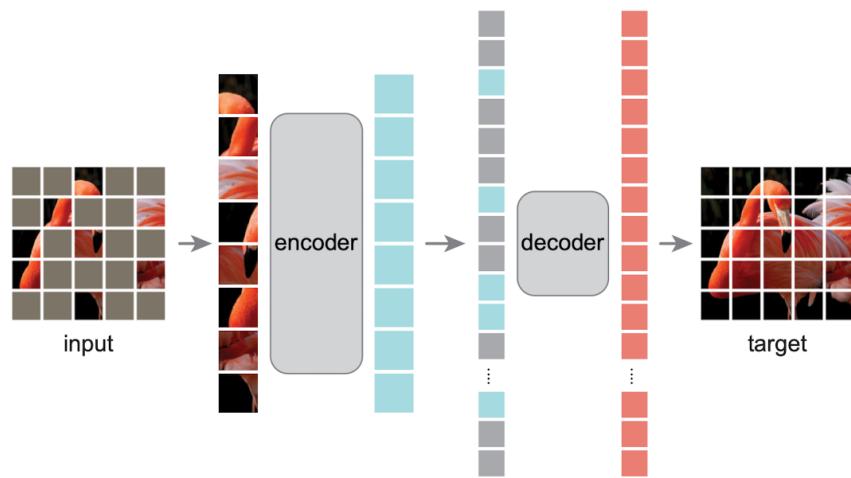


Figura 2.37. Arquitectura de MAE [97].

2.3.5. Arquitectura DSSL

La arquitectura **DSSL** (Directional Self-Supervised Learning), introducida en 2021 [98], plantea una perspectiva "direccional" para aplicar técnicas de *data augmentation* ponderadas por la intensidad de sus transformaciones. De este modo, se distinguen entre **standard augmentations**, como recorte aleatorio, *horizontal flip*, distorsión de color o desenfoque gaussiano, y **heavy augmentations**, como pueden ser *RandAugment* [99] o *Jigsaw* [100].

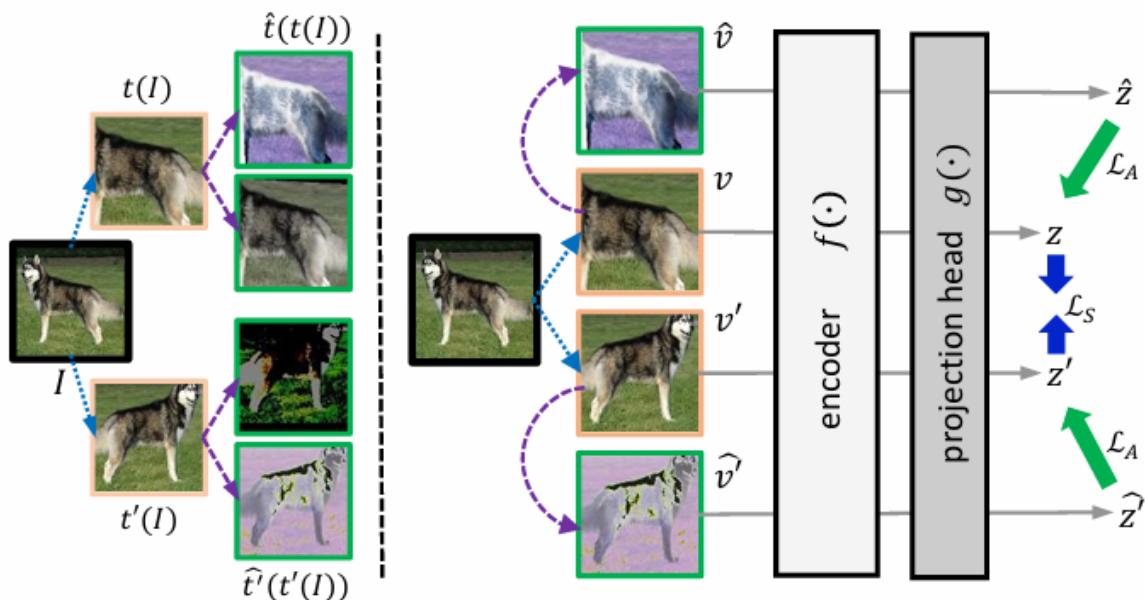


Figura 2.38. Arquitectura DSSL [98].

Tal como se ilustra en la figura 2.38, por cada imagen de entrada se obtienen cuatro vistas aumentadas, de las cuales dos son *standard augmentations* y las otras dos son *heavy augmentations*. Basándose en la estructura de Sim-Siam, se intenta aumentar la similitud entre la imagen original y cada una de las vistas *standard* de manera simétrica. Esto quiere decir que ambas partes tienen "gravedad propia" dentro del espacio latente al que son mapeadas mediante un *encoder*, y se atraen entre sí con la misma fuerza.

Por otro lado, el método para conseguir que las vistas *heavy* aporten información relevante sin confundir al modelo, es maximizando su similitud de manera asimétrica con la imagen inicial. Siguiendo la analogía anterior, en este caso solo presenta gravedad la imagen original, a la que se ven atraídas

las representaciones de las vistas *heavy* (figura 2.39).

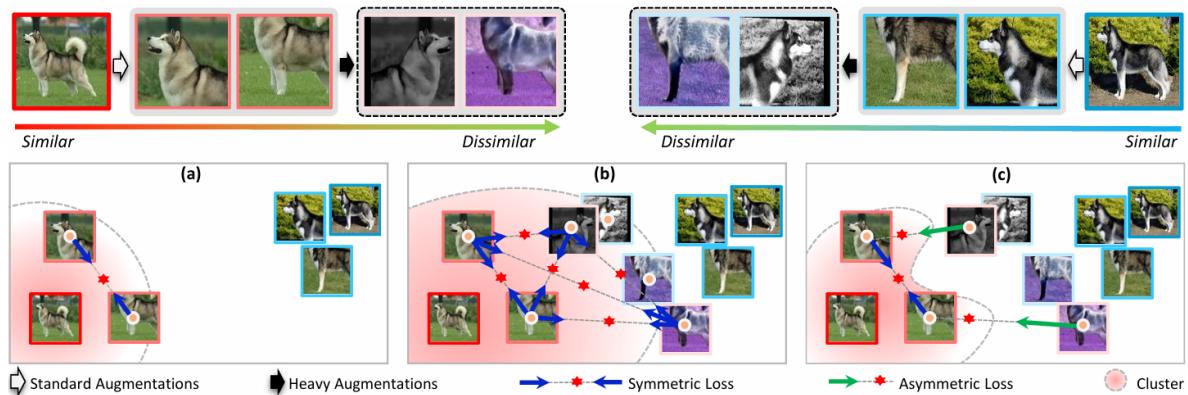


Figura 2.39. Mapeo al espacio latente de vistas *standard* y *heavy* [98].

3.

Trabajos Relacionados

Con la evolución de los métodos de aprendizaje automático, se han desarrollado numerosos modelos y técnicas que han mejorado significativamente la precisión y eficiencia de la segmentación de imágenes . Como se comentó previamente en el apartado 2.2.4, la arquitectura U-Net ha sido especialmente relevante en este proceso, sentando las bases de muchos modelos que hoy en día alcanzan el estado del arte en esta rama.

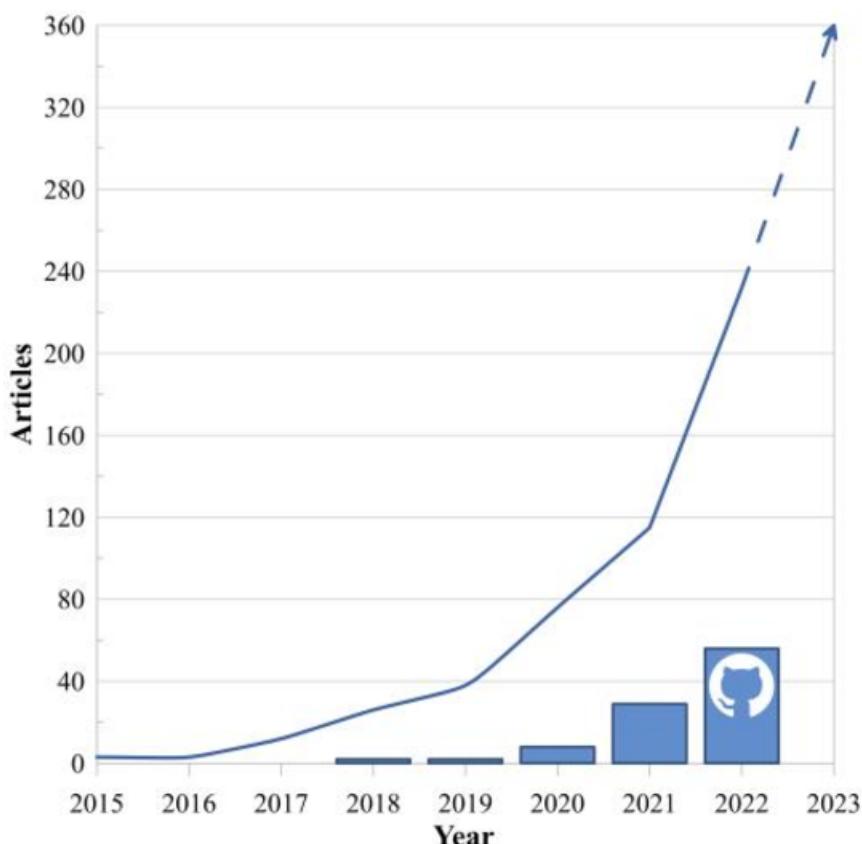


Figura 3.1. Frecuencia anual de los artículos publicados sobre SSL [101].

Sin embargo, debido a la aparición tan reciente del SSL, la gran mayoría de estos trabajos no utilizan técnicas de pre-entrenamiento de este estilo(figura 3.1). Algunos ejemplos son:

- ***UNet 3+, A Full-Scale Connected UNet for Medical Image Segmentation, Huimin Huang et al. (2020)***: Mejora la U-Net original (apartado 2.2.4) mediante una estructura de *encoder-decoder* convolucional más profunda y densa, con conexiones residuales anidadas que permiten una combinación efectiva de características multi-escala. Además, integra un módulo de fusión de características que combina información a diferentes niveles de abstracción [102].
- ***SegFormer, Simple and Efficient Design for Semantic Segmentation with Transformers, Enze Xie et al. (2021)***: Combina un *backbone* basado en Transformers con un cabezal (*head*) relativamente ligero, logrando un equilibrio entre precisión y eficiencia computacional. Utiliza un módulo de atención jerárquica para capturar relaciones espaciales a diferentes escalas en imágenes de alta resolución, superando a muchos modelos convencionales en términos de precisión y velocidad. [103].
- ***Swin UNet, Unet-like Pure Transformer for Medical Image Segmentation, Hu Cao et al. (2021)***: Utiliza Swin Transformers, que emplean una jerarquía de ventanas desplazadas para reducir la complejidad computacional mientras mantienen la capacidad de modelar relaciones a largo alcance. Este enfoque jerárquico permite procesar imágenes a múltiples escalas y resoluciones, capturando tanto detalles finos como estructuras globales, lo que es esencial para la segmentación precisa de imágenes médicas complejas [104].

Todas estas publicaciones presentan los problemas comunes a aquellos modelos supervisados comentados en el apartado 2.3.2. Entre aquellas arquitecturas que sí aplican el SSL para la tarea de segmentación, destacan las siguientes. Nótese que **ninguna es anterior al año 2020**:

- ***DeSD, Self-Supervised Learning with Deep Self-Distillation for 3D Medical Image Segmentation, Yiwen Ye et al. (2022)***: Emplea una estrategia de auto-destilación profunda (similar a Bert) mediante la cual se centra en mejorar las representaciones en capas superficiales del modelo. Está pensado para aplicarse a datos médicos tridimensionales mediante la segmentación 3D, utilizando una combinación de tareas de reconstrucción de imágenes y aprendizaje **contrastivo** [105].

-
- ***Self-Pre-training with Masked Autoencoders for Medical Image Classification and Segmentation, Lei Zhou et al. (2023)***: Este método utiliza MAEs para un preentrenamiento auto supervisado **generativo**, donde se enmascaran partes de la imagen y el modelo aprende a reconstruirlas. De esta manera, el modelo puede aprender representaciones robustas sin necesidad de grandes cantidades de datos etiquetados, mejorando la precisión del previo Estado del Arte, tanto en clasificación como en segmentación de imágenes médicas [106].
 - ***Fully Self-Supervised Learning for Semantic Segmentation, Yuan Wang et al. (2022)***: Este enfoque de SSL **contrastivo** para segmentación semántica, emplea un esquema de aprendizaje que combina tareas de predicción de contextos y reconstrucción de imágenes. Se basa en el uso de técnicas de aumento de datos específicas y la integración de un modelo de atención [107].

No obstante, la aplicación conjunta de técnicas discriminativas y generativas de SSL en esta rama es muy poco representativa. Este PFG se distingue por integrar ambos enfoques en el pre-entrenamiento de modelos específicos de segmentación en imágenes laparoscópicas, para conseguir, incluso con recursos limitados, mejorar el rendimiento de los modelos supervisados tradicionales. Esta combinación no solo mejora la capacidad de generalización del modelo, sino que también aumenta la precisión de la segmentación en un contexto quirúrgico, donde la variabilidad y complejidad de las imágenes representan un desafío significativo.

4.

Metodología

En este capítulo se explicará detalladamente el proceso seguido para desarrollar un *pipeline* de pre-entrenamiento auto supervisado y ajuste fino, que mezcla técnicas discriminativas y contrastivas para la segmentación de imágenes laparoscópicas.

Se plantea como hipótesis principal del PFG que, bajo las mismas restricciones y entornos de entrenamiento, esta propuesta auto supervisada será capaz de mejorar los resultados obtenidos por modelos supervisados en un menor tiempo.

De este modo, se definirán en primer lugar las bases para garantizar una comparación justa entre todas las arquitecturas implementadas y posteriormente se abordarán las particularidades de todas ellas.

4.1. Consideraciones previas

Esta sección desglosa las diferentes tecnologías utilizadas durante el transcurso de los entrenamientos. Además, se otorga un esquema general del procedimiento seguido para la implementación y desarrollo de los modelos junto con la realización de un análisis de los distintos conjuntos de datos utilizados.

4.1.1. Entorno de Trabajo

Debido a los limitados recursos disponibles para realizar este PFG, no se disponía de un entorno local lo suficientemente potente para cargar arquitecturas grandes y entrenar redes neuronales complejas. Además, el espacio en disco no era suficiente para almacenar los distintos *datasets* que se han utilizado

en el desarrollo del proyecto.

Como solución, se ha optado por la alternativa que ofrece *Google Colaboratory*, una plataforma de *Google* que permite la ejecución de cuadernos de *jupyter*¹ en la nube. Para ello, asigna un entorno de desarrollo virtual en tiempo real, en el que el usuario dispone de una determinada memoria RAM y un espacio en disco.

Sin embargo, la principal ventaja de este servicio radica en que ofrece de manera gratuita y por tiempo limitado, la posibilidad de usar tarjetas gráficas (GPUs) optimizadas para realizar cálculos relacionados con el aprendizaje automático. Estas son infinitamente más rápidas y escalables que las CPUs para efectuar operaciones matriciales, que son la base matemática de las NN. En la figura 4.1 se ilustra una comparación entre el rendimiento de ambos tipos de *hardware*. Para este trabajo se ha utilizado únicamente el modelo **Tesla T4**, que cuenta con 15 gigabytes de memoria y proporciona un rendimiento aceptable (es la más barata dentro de *Google Colab*).

¹Los cuadernos de Jupyter son entornos interactivos que permiten combinar código ejecutable, texto explicativo, visualizaciones y otros elementos multimedia en un solo documento, facilitando el desarrollo, la documentación y el análisis de datos de manera integrada y colaborativa.

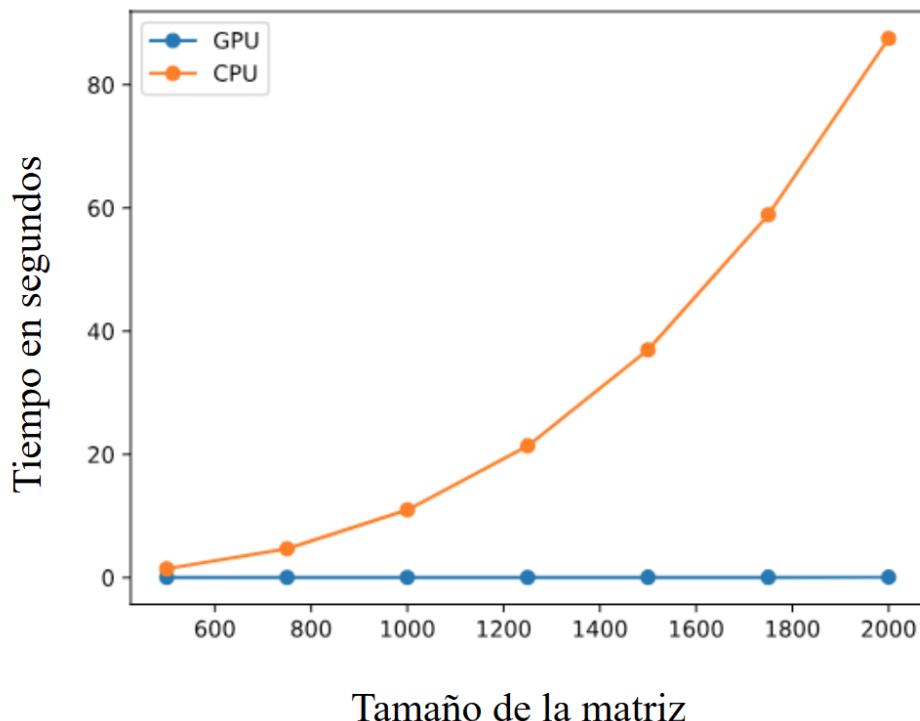


Figura 4.1. Diferencia de tiempo entre CPU y GPU (Nvidia Tesla T100) al realizar operaciones matriciales.

Por otra parte, nos permite vincular nuestra cuenta de *Google Drive* y *Github* (Anexo B), para persistir *datasets* y resultados en la nube, así como controlar las versiones del código. De esta manera, se plantea la arquitectura del entorno de trabajo mostrada en la figura 4.2, donde se aprovecha el ecosistema *Google* y su compatibilidad con *Git*.

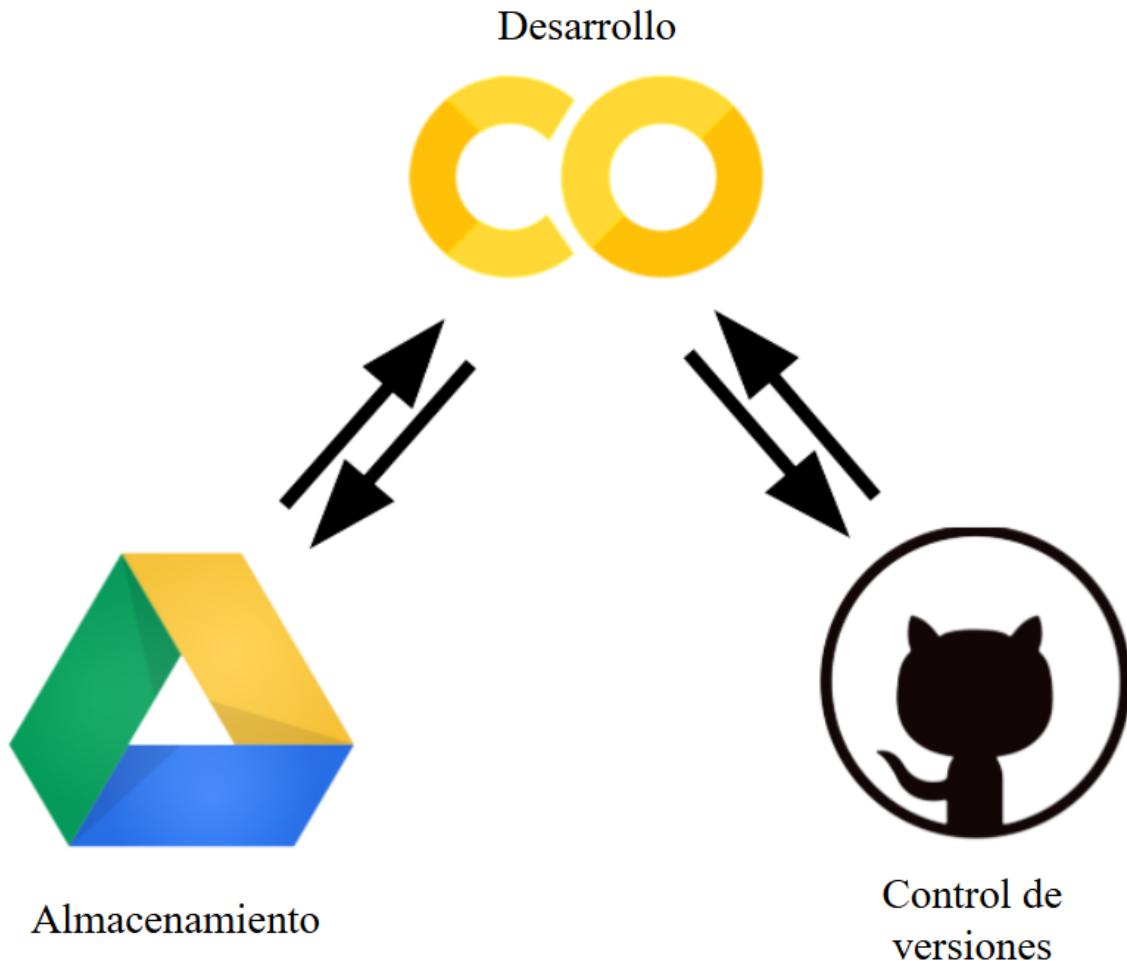


Figura 4.2. Arquitectura global del entorno de trabajo.

En este trabajo, la disponibilidad de GPUs que ofrecía *Colab* en su versión gratuita no ha sido suficiente para realizar los entrenamientos de manera satisfactoria, resultando imperativo la contratación del plan Pro Plus [108]. Además, ha sido también necesario obtener el plan *Standard* de *Google One* que permite el almacenamiento de 200 gigabytes en *Google Drive*.

El lenguaje de programación utilizado principalmente ha sido Python, debido al gran número de librerías que ofrece relacionadas con el procesamiento de datos y el aprendizaje automático. Su popularidad se ha disparado en la última década gracias al auge del DL y el *Big Data*, convirtiéndose en el lenguaje más utilizado por los desarrolladores de este campo. Presenta una sintaxis clara y legible que facilita el desarrollo rápido de prototipos, y es fácilmente integrable con multitud de tecnologías.

	Keras 	TensorFlow 	PyTorch 
Level of API	high-level API ¹	Both high & low level APIs	Lower-level API ²
Speed	Slow	High	High
Architecture	Simple, more readable and concise	Not very easy to use	Complex ³
Debugging	No need to debug	Difficult to debugging	Good debugging capabilities
Dataset Compatibility	Slow & Small	Fast speed & large	Fast speed & large datasets
Popularity Rank	1	2	3
Uniqueness	Multiple back-end support	Object Detection Functionality	Flexibility & Short Training Duration
Created By	Not a library on its own	Created by Google	Created by Facebook ⁴
Ease of use	User-friendly	Incomprehensive API	Integrated with Python language
Computational graphs used	Static graphs	Static graphs	Dynamic computation graphs ⁵

Figura 4.3. Comparación entre Keras, TensorFlow y PyTorch [109].

En concreto, se ha empleado el *framework* de PyTorch [110] para manejar los entrenamientos, puesto que permite un mayor control a bajo nivel que otras alternativas, como TensorFlow [111] o Keras [112]. En la figura 4.3 se pueden apreciar los puntos fuertes de cada uno de ellos.

4.1.2. Estudio del Dataset

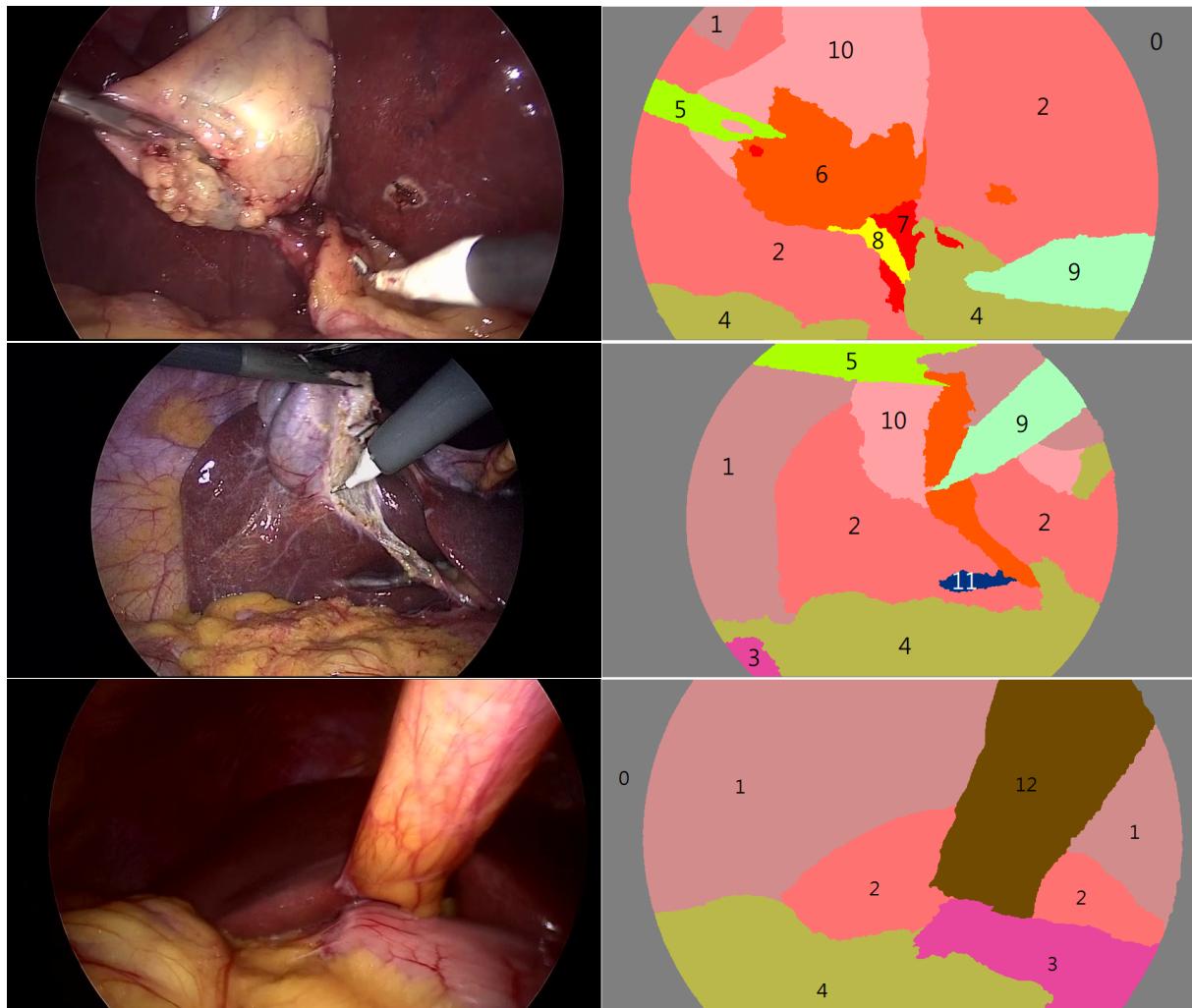


Figura 4.4. Fotogramas y etiquetas correspondientes de CholecSeg8k [113].

El *dataset* seleccionado para realizar el entrenamiento de los distintos modelos es el **CholecSeg8k** [113], pensado desde un principio para la segmentación semántica. Incluye 8080 fotogramas acompañados de sus correspondientes anotaciones, obtenidos de 17 vídeos diferentes grabados por un laparoscopio durante cirugías laparoscópicas de colecistectomía (provenientes de Cholec80 [114]). En la figura 4.4 se pueden observar algunos ejemplos del contenido del *dataset*.

Las imágenes originales tienen una resolución de 854×480 píxeles y tienen tres canales de profundidad (como todas las imágenes a color al estar en for-

mato RGB). Por otra parte, las etiquetas cubren 13 clases diferentes contando con el background negro, que, tal y como se muestra en la tabla 4.1, están distribuidas de manera no uniforme.

Nº de Clase	Nombre de Clase	Código Hexadecimal RGB	Porcentaje del Total
Clase 0	Background Negro	#505050	\
Clase 1	Pared Abdominal	#111111	29,46 %
Clase 2	Hígado	#212121	29,39 %
Clase 3	Tracto Gastrointestinal	#131313	2,56 %
Clase 4	Grasa	#121212	20,23 %
Clase 5	Pinza	#313131	3,32 %
Clase 6	Tejido Conectivo	#232323	3,125 %
Clase 7	Sangre	#242424	0,573 %
Clase 8	Conducto Cístico	#252525	0,05 %
Clase 9	Electrodo L-Hook	#323232	1,824 %
Clase 10	Vesícula Biliar	#222222	8,893 %
Clase 11	Vena Hepática	#333333	0,018 %
Clase 12	Ligamento Redondo del Hígado	#050505	0,561 %

Tabla 4.1. Nombre, Código Hexadecimal RGB y Porcentaje del Total para todas las clases de CholecSeg8k.

Esto quiere decir que si la red únicamente aprendiese a segmentar las tres clases más representadas (1, 2 y 4), su precisión alcanzaría casi el 80% y podría producir una falsa sensación de éxito. Un modelo que no aprende a diferenciar todas las clases de manera consistente y confiable es un modelo inservible, por lo que será primordial asegurar que su rendimiento no se ve afectado por este desbalanceo.

Por último, cabe destacar que ocupa 3 gigabytes en disco y la estructura de carpetas original es la que se ilustra en la figura 4.5. Se usarán las anotaciones *watershed* para el entrenamiento de los *baselines* por la facilidad que ofrecen para ser procesadas, y las *color_mask* para visualizarlas.

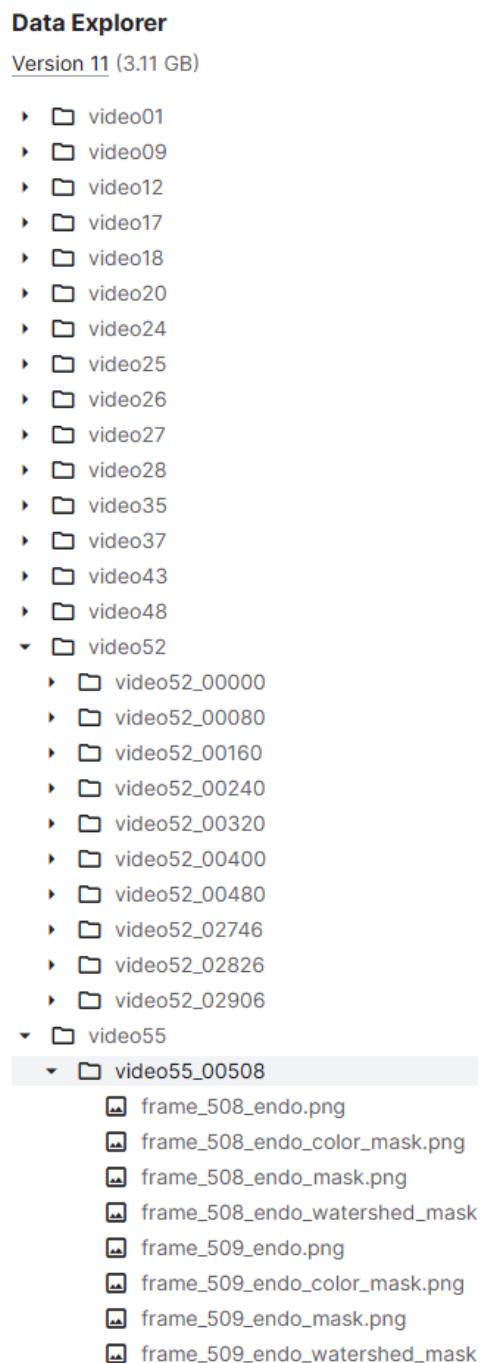


Figura 4.5. Estructura de directorios de CholecSeg8k.

4.1.3. Estructura de la investigación

Para poder contrastar el desempeño de las redes supervisadas y las auto supervisadas, se ha elegido como arquitectura base a la **U-Net**, que está presente en la mayor parte de las publicaciones en los últimos años relacionadas con la segmentación de imágenes [115, 116].

En primer lugar, se desarrollarán dos *baselines* supervisados que servirán como representantes de los dos tipos de redes más relevantes actualmente en el campo del DL: Las CNNs y los Transformers. En concreto, se utilizará el **framework nnUNet** [117] y una adaptación de la arquitectura **Uformer** [118], cuyos detalles se encuentran en el apartado 4.2.

Posteriormente, se implementará un *pipeline* de pre-entrenamiento y *fine tuning* utilizando el paradigma SSL. Para ello, se ha elaborado una arquitectura basada en **RepRec** [119], donde se utilizan técnicas auto supervisadas y discriminativas para pre-entrenar dos redes distintas sobre un *dataset* de mayor tamaño [120]. Más adelante, como se describe en el apartado 4.3.5, se ensamblan en una sola estructura y se aplica *fine tuning* para ajustarse al *dataset* objetivo (CholecSeg8k).

Con este avance incremental, se pretende seguir un principio básico en el desarrollo de cualquier sistema de aprendizaje automático, popularizado por el influyente investigador y profesor Andrew N.G. :

"Build your first system quickly, then iterate"

– Andrew N.G.

La idea principal es empezar por el problema más básico, proponer una solución sencilla rápidamente e iterar hasta tener un resultado satisfactorio. De este modo, se han realizado numerosas pruebas en cada una de las arquitecturas hasta alcanzar el mejor resultado posible, y solo entonces se ha procedido a comparar el rendimiento global de los modelos.

4.2. Desarrollo de los *baselines*

En esta sección se expone minuciosamente el procedimiento seguido para la elaboración de los *baselines*, proporcionando detalles de su implementación y plasmando su evolución desde el planteamiento hasta su validación.

Por una parte, la **nnUNet** (*No New U-net*) [117] es un *framework open source* que puede ser usado por usuarios no expertos como una herramienta mecanizada para segmentar imágenes médicas. De esta manera, se pretende que médicos, doctores o cirujanos cuenten con material adicional que les ayude a la hora de diagnosticar u operar pacientes. Solo tendrían que proporcionar un conjunto de imágenes, en un formato reconocible, para el entrenamiento automatizado de una red convolucional U-Net, sin necesidad de ajustes manuales.

Más en detalle, esta herramienta realiza una preconfiguración automática basada en un conjunto de reglas heurísticas, obtenidas mediante el análisis del *dataset* específico, ajustando parámetros como el tamaño de los patches, la profundidad de la red, el escalado de imágenes y los esquemas de normalización. La nnU-Net incluye múltiples configuraciones (2D, 3D *full resolution*, 3D *low resolution*) y una estrategia de *ensembles* para mejorar la robustez y la precisión de la segmentación, como se puede observar en la figura 4.6. Utiliza técnicas avanzadas de *data augmentation* y aplica una cascada de segmentación en múltiples resoluciones para manejar diferentes escalas y detalles en las imágenes médicas.

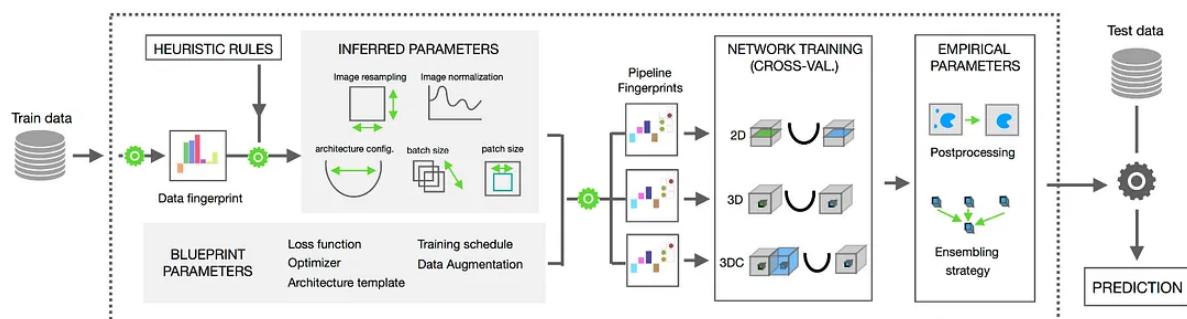


Figura 4.6. Flujo de trabajo de la nnUNet [121].

A diferencia de la nnUNet, la arquitectura **Uformer** sí requiere de modificaciones manuales y conocimientos técnicos. Pretende, de forma similar a la previamente mencionada Swin U-Net [104], aplicar el concepto de los Transformers al marco de la U-Net, capturando dependencias globales y contextuales en las imágenes de manera efectiva.

El Uformer fue originalmente diseñado para tareas de restauración de imágenes y no de segmentación semántica. Pese a ello, utiliza conceptos y componentes novedosos muy interesantes, que llevaron a que se optara por él, en lugar de elegir otras alternativas quizá más sencillas de implementar como la propia Swin U-Net, nnFormer [122], U-Netmer [123] o UNETR [124].

Dichas innovaciones del Uformer se concentran en la inclusión de **Moduladores** y de bloques **LeWinTransformer**, que tal y como se ilustra en la figura 4.7, conforman los bloques del encoder y del decoder.

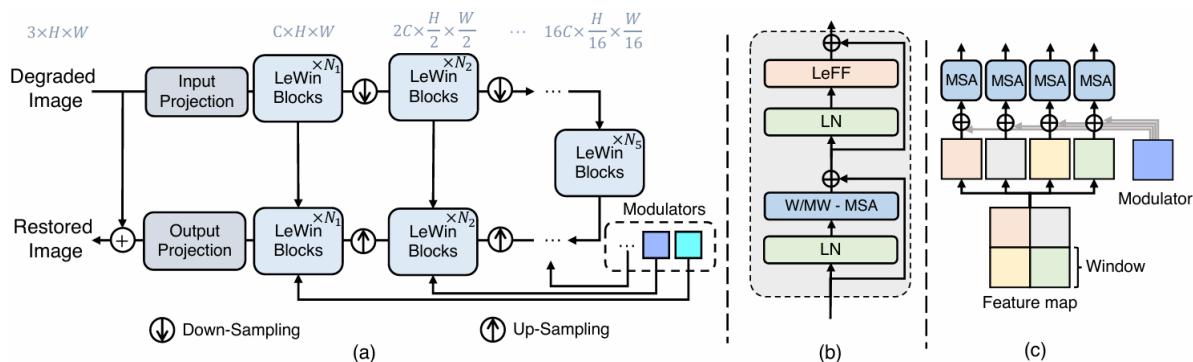


Figura 4.7. Arquitectura Uformer. (a) Visión general. (b) Bloque *LeWinTransformer*. (c) Ilustración de cómo los moduladores modulan las capas de W-MSAs (*Window-based Multi-head Self-Attention*) en cada bloque *LeWinTransformer*, denominado MW-MSA (*Modularized W-MSA*) en (b). [118].

Con mayor precisión, los bloques *LeWinTransformer* (*Locally-Enhanced Window-based Transformer*), dividen la imagen en parches no superpuestos y emplean mecanismos de atención a nivel de ventana (*window-based self-attention*), de manera similar a los Swin Transformers (apartado 2.2.3). Sin embargo, incluyen en su MLP (*Locally-Enhanced Feed Forward*, LeFF) capas convolucionales que le ayudan a capturar dependencias locales de manera mucho más sencilla, como se reporta en su artículo.

Los Moduladores, por otra parte, no son más que *bias* espaciales multiesca-

la que sirven para ajustar las características en múltiples capas del *decoder* del Uformer. Específicamente, se añade un tensor² aprendible a las *features* extraídas por cada bloque *LeWinTransformer*, con el objetivo de aportar información sobre cuánto ha variado la imagen. Este componente supone una adición muy ligera pero relevante, que si bien puede no aportar contexto relevante en algún caso, nunca traba ni ralentiza el aprendizaje del modelo.

4.2.1. Acondicionamiento del Dataset

Antes de entrenar cualquier modelo, es necesario procesar y preparar los datos con el objetivo de dejarlos listos para su utilización. Principalmente, se debe garantizar que son consistentes y que el modelo puede entenderlos y aprender de ellos.

Preparación de los Datos

El conjunto de datos total es relativamente pequeño comparado con los grandes modelos actuales, que utilizan millones de imágenes para producir resultados que son estado del arte, dedicando un porcentaje muy elevado de ellas para el entrenamiento. De este modo, en la tabla 4.2 se puede observar cómo se realiza en este trabajo la división en subconjuntos de **Entrenamiento**, **Validación** y **Prueba** del dataset.

Subconjunto	Nº de ejemplos	Porcentaje
Entrenamiento	5171	64 %
Validación	1293	16 %
Prueba	1616	20 %

Tabla 4.2. División del dataset total en conjuntos de Entrenamiento, Validación y Prueba, con número total de muestras n.

Siguiendo las restricciones que impone la nnUNet sobre el formato de los

²Un tensor es la manera que tienen librerías como Tensorflow, Keras o PyTorch de representar vectores y matrices

directorios y los archivos, se define la carpeta **nnUNet_raw**, donde permite introducir distintos *datasets*, identificados por un número de 3 cifras y un nombre: **Dataset212_CholecSeg8kV2**. En su interior, los datos están estructurados por directorios, tal y como se muestra en la figura 4.8, de modo que cada imagen está asociada a su etiqueta correspondiente, con la que comparte identificador³.

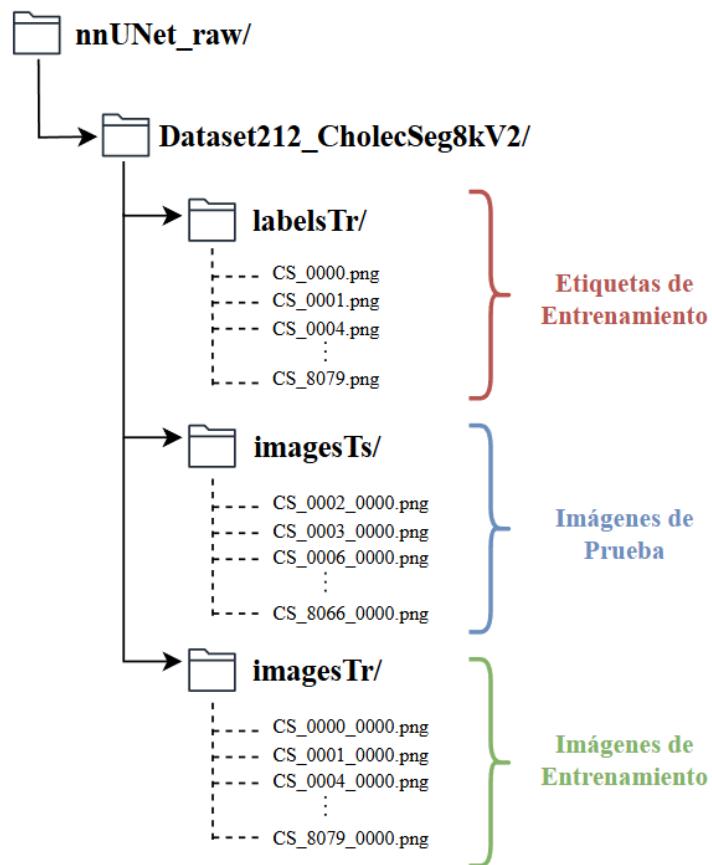


Figura 4.8. Estructura de carpetas en el formato de la nnUNet.

Preprocesamiento de las Imágenes

Para poder alimentar al modelo con las etiquetas, estas se deben transformar en un mapa de una sola dimensión, donde las clases deben estar representadas por valores consecutivos empezando desde el 0. Esto se consigue de

³Cabe anotar que en los directorios *imagesTs* e *imagesTr*, el nombre de los archivos se compone de un identificador del dataset (CS), un identificador de cuatro cifras de la imagen, y un identificador del canal pensado para imágenes 3D

manera sencilla, mappeando los valores de los píxeles de los *labels* al valor de su clase correspondiente.

Por otra parte, los píxeles de las imágenes toman valores en el rango [0, 255], lo que podría llevar a que la red otorgue mayor importancia a aquellos con cifras más altas. Para evitarlo, la nnUNet aplica de manera automática una operación de **normalización** particular, conocida como Z-Score.

La ecuación 4.1 detalla cómo se realiza este cálculo: a cada píxel x se le resta la media μ del valor de todos los píxeles del dataset, para posteriormente dividirlos entre la desviación típica σ .

$$Z = \frac{x - \mu}{\sigma} \quad (4.1)$$

En el Uformer, sin embargo, no se aplicaba ninguna técnica de normalización por defecto, por lo que se ha implementado de manera manual. Además, como la arquitectura original solo aceptaba imágenes cuadradas con dimensiones de potencias de 2, se han realizado una serie de variaciones estructurales sobre el código fuente, que han permitido alimentar al modelo con imágenes estrictamente rectangulares, es decir, donde el largo supone el doble que el alto. De esta forma, se reduce en gran medida la pérdida de información que supondría recortar los datos o alterar su escala, posibilitando una transformación más natural a una resolución de 512×256 , mediante la librería OpenCV (listado 4.1).

Listado 4.1. Transformación del tamaño de las imágenes para el entrenamiento del Uformer.

```
import cv2
from skimage import io

for file_path in dataset:
    imagen = io.imread(file_path)
    nueva_imagen = cv2.resize(imagen, (512, 256), interpolation
        = cv2.INTER_NEAREST)
    io.imsave(file_path, nueva_imagen, check_contrast=False)
```

4.2.2. Pérdidas

Elegir una función de pérdida adecuada es crucial al entrenar un modelo, ya que determina la dirección hacia la cual debe converger. Por lo tanto, para comparar de manera justa ambos *baselines*, es esencial utilizar la misma función de pérdida en cada uno de ellos para asegurar la consistencia.

Del mismo modo, emplear métricas distintas puede resultar en interpretaciones erróneas del rendimiento, dificultando la determinación de cuál modelo es superior en una evaluación equitativa. Utilizar métricas uniformes garantiza que los resultados reflejen el mismo aspecto del desempeño, permitiendo una comparación justa.

Siguiendo, una vez más, la configuración por defecto que trae la nnUNet, se ha optado por definir la pérdida de ambos *baselines* como una suma ponderada entre la **Categorical CrossEntropy** (CE) y la **Dice Loss**. Se expresa matemáticamente en la ecuación (4.2), donde β es un hiperparámetro que controla qué importancia otorgar a cada subcomponente. Tanto para los *baselines* como para el modelo SSL, **se ha establecido β a 0,5** para que ambas pérdidas tengan el mismo peso.

$$\mathcal{L}_{\text{Final}} = \beta \cdot \mathcal{L}_{\text{CE}} + (1 - \beta) \cdot \mathcal{L}_{\text{Dice}} \quad (4.2)$$

Más detalladamente, **Categorical CrossEntropy** es una variante de la BCE (explicada en el subapartado 2.2.1) que puede ser aplicada en contextos donde se tienen más de dos clases. Es una de las pérdidas más usadas para tareas de segmentación semántica, puesto que permite una convergencia rápida y penaliza las predicciones incorrectas mediante un enfoque simple y directo. No obstante, esta sencillez la hace susceptible a los desbalanceos de las clases, que como ya se mencionó anteriormente, suponen un desafío para el conjunto de datos objetivo. Está definida en la fórmula (4.3), donde N es el tamaño del conjunto de entrenamiento, M es el número de clases, y_i son las etiquetas e \hat{y}_i son las predicciones.

$$\mathcal{L}_{\text{CE}} = - \sum_{i=1}^N \sum_{j=1}^M y_i \log(\hat{y}_i) \quad (4.3)$$

Para complementar las ventajas que ofrece la CE y contrarrestar sus carencias, se utiliza la pérdida **Dice**. Esta es especialmente útil para manejar desequilibrios de clases en tareas de segmentación, al centrarse en la superposición entre las predicciones y las etiquetas. En el contexto de la segmentación, las áreas de interés pueden ser zonas muy pequeñas de la imagen y suelen estar subrepresentadas. Por ejemplo, en el dataset CholecSeg8k, la vena hepática, cuya detección es crucial para evitar provocar heridas durante el procedimiento quirúrgico [125], solo está presente en el 0,018 % de los píxeles (tabla 4.1). La *Dice Loss* mitiga este problema ponderando cada clase de manera proporcional a su presencia en la imagen. Esto significa que los errores en las clases minoritarias (por ejemplo, objetos pequeños) tienen un impacto proporcionalmente mayor en la pérdida total, incentivando al modelo a aprender mejor cómo segmentarlas.

La implementación de esta pérdida está basada en la expresión (4.4), donde los vectores y e \hat{y} contienen las etiquetas y las predicciones del modelo, respectivamente.

$$\mathcal{L}_{\text{Dice}} = -\frac{2 \sum_{i=1}^N \hat{y}_i \cdot y_i}{\sum_{i=1}^N \hat{y}_i + \sum_{i=1}^N y_i} \quad (4.4)$$

4.2.3. No New UNet

Como se ha detallado al inicio de esta sección, el *framework* de la **nnUNet** está pensado para funcionar *out-of-the-box*, necesitando configuraciones mínimas a nivel de usuario intermedio. Su *pipeline* está semi-automatizado, requiriendo la ejecución de no más de cinco comandos durante todo el procedimiento, especificando la ruta y los identificadores del *dataset*.

Una de las muchas ventajas que ofrece esta herramienta, es la producción de archivos y *logs* que contienen información relevante sobre los datos. De ellos se obtiene la tabla 4.3, donde se incluyen los valores máximos y mínimos de los píxeles antes de normalizar, separados por los canales RGB, así como la mediana, la media, la desviación típica y los percentiles 0,5 y 99,5.

Canal	Máximo	Mínimo	Mediana	Media	Desviación Típica	Percentil 0.5	Percentil 99.5
0	255.0	0.0	109.0	117.46	60.07	13.0	255.0
1	255.0	0.0	64.0	74.53	54.68	0.0	252.0
2	255.0	0.0	55.0	63.40	47.58	0.0	250.0

Tabla 4.3. Estadísticas extraídas por nnUNet del *dataset* antes de normalizar.

Una vez normalizados por canales aplicando la Z-Score (ecuación 4.1), los valores obtenidos se muestran en la tabla 4.4

Canal	Máximo	Mínimo	Mediana	Media	Desviación Típica	Percentil 0.5	Percentil 99.5
0	2.29	0.0	0.98	1.05	1.05	0.12	2.29
1	3.30	0.0	0.83	0.96	0.71	0.0	3.26
2	4.03	0.0	0.87	1.00	0.66	0.0	3.95

Tabla 4.4. Estadísticas extraídas por nnUNet del *dataset* después de normalizar.

De este modo, utiliza toda esta información para generar reglas heurísticas, que le permiten construir el modelo que mejor se adapte al problema, en este

caso, uno enfocado a la segmentación 2D. Dicho modelo se ha entrenado por **250 épocas** con la siguiente configuración proporcionada por la herramienta:

1. Se utilizan 8 **bloques** en el **encoder**, donde cada uno de ellos contiene 2 **capas convolucionales** con un **kernel de tamaño** 3×3 , que se aplican por **parches de tamaño** 640×384 . Además, la primera convolución de cada bloque presenta un **stride de** 2, mientras que la segunda tiene un **stride de** 1.

La profundidad de las representaciones, dada por el número de **features** a la salida de cada bloque en orden secuencial, es la siguiente: [32, 64, 128, 256, 512, 512, 512, 512]

2. Para el **decoder** se mantiene la misma estructura que en el **encoder**. Sin embargo, entre bloques se utiliza una **capa convolucional transpuesta**, con un **kernel de tamaño** 2×2 y un **stride de** 2.
3. Se emplea un **batch size** de 13 ejemplos, ajustándose a las limitaciones de memoria del entorno de desarrollo.
4. Utiliza la regularización **L2**, con un **weight decay** de 0,00003, mediante **AdamW**. Este es una variación sencilla del optimizador **Adam**, explicado en el apartado [2.2.1](#), pensado para incorporar un índice de regularización.
5. Las capas ocultas se activan mediante la función **LeakyReLU**.
6. Se aplica *Batch Normalization* por instancias (**InstanceNorm**) en cada uno de los bloques, y no se utilizan capas de *dropout*.
7. Se ha utilizado un planificador polinómico de la tasa de aprendizaje (**PolyLRScheduler**) para reducir el valor de dicho hiperparámetro a medida que avanza el entrenamiento. De esta forma se consigue suavizar la agresividad con la que aprende el modelo en las últimas épocas, donde su conocimiento está bastante afianzado. En una primera instancia, el valor del **learning rate** es de 0,01.

En la figura [4.9](#) se ilustra de manera simplificada la arquitectura del modelo nnUNet utilizado.

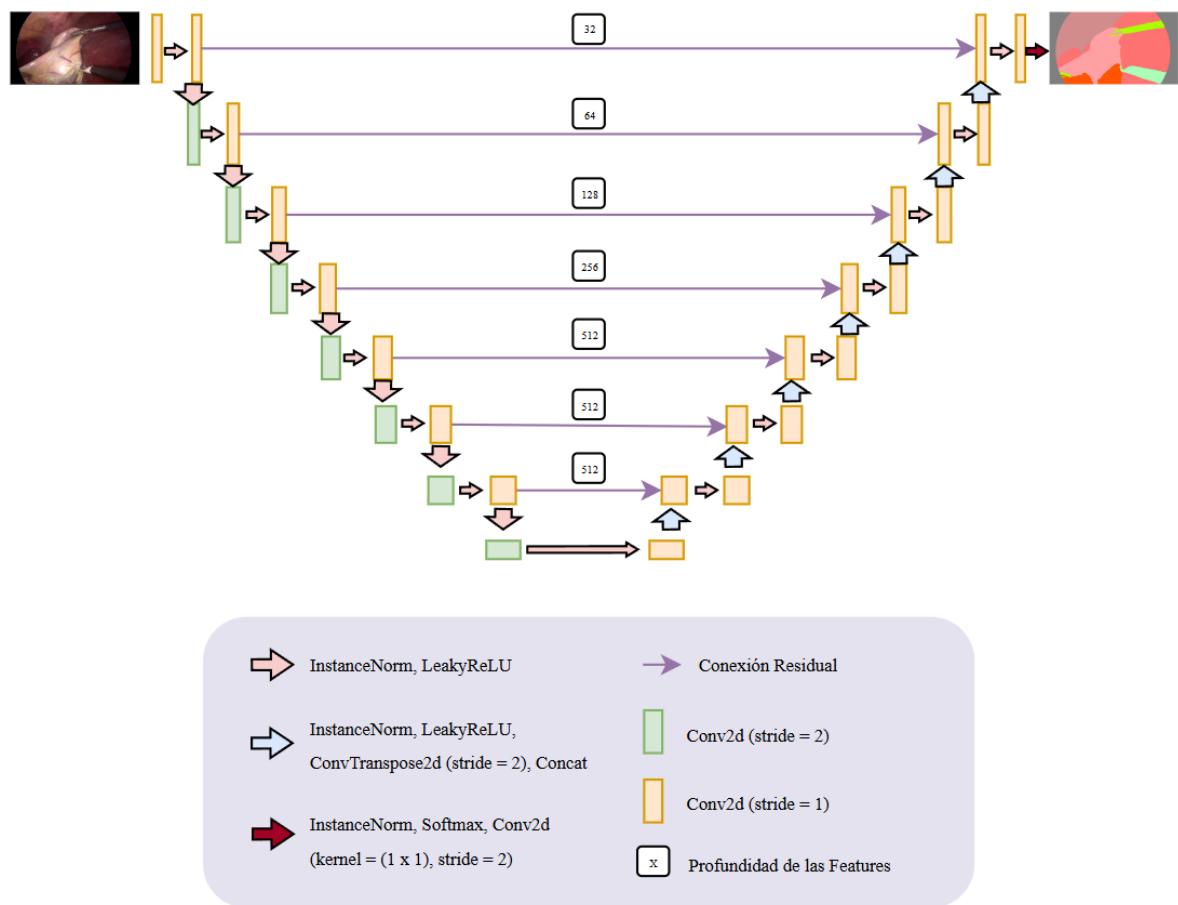


Figura 4.9. Arquitectura del Modelo nnUNet.

4.2.4. Uformer

Para poder utilizar la arquitectura Uformer y comparar su rendimiento al de la nnUnet, se han realizado una serie de cambios y adaptaciones sobre el esquema presentado en la publicación original. En esta sección se proporcionarán todos los detalles al respecto y se explicarán minuciosamente las configuraciones de los modelos utilizados a lo largo de los distintos entrenamientos.

Adaptación para Segmentación Semántica

El **Uformer**, tal y como se ha comentado anteriormente, está orientado a problemas de reconstrucción y eliminación de ruido. En estas tareas es de gran importancia obtener una representación comprimida de la imagen que tenga conciencia espacial, para mantener cierta consistencia contextual con la imagen resultante. Muchos de estos conceptos se pueden aplicar a la segmentación semántica.

Concretamente, es necesario sustituir la función de pérdida que trae por defecto, por la suma ponderada de la CE y la *Dice Loss*, definida en el subapartado 4.2.2. Esto ocurre debido a que la pérdida utilizada por el Uformer en su dominio original es la *Charbonnier Loss*, que calcula la discrepancia entre las distribuciones de probabilidad de las predicciones y las etiquetas sin tener en cuenta las clases. Matemáticamente, se define dicha función en la ecuación (4.5)

$$\mathcal{L}_{Charbonnier} = \sqrt{(\hat{y} - y)^2 + \epsilon^2} \quad (4.5)$$

Por otro lado, para que el modelo devolviese una segmentación con el número correspondiente de clases, ha sido preciso realizar las siguientes modificaciones:

- El objeto **DataLoader** de PyTorch es el responsable de cargar por lotes las imágenes de entrada y las etiquetas en la memoria de la GPU. En caso de pretender cargar todas las imágenes a la vez, no habría espacio suficiente y el entrenamiento terminaría en error. Dicho objeto se ha

retocado para ser compatible con la estructura unidimensional de las etiquetas, asegurando una correcta compatibilidad con el *dataset*.

- Además de modificar los bloques del *LeWinTransformer* para aceptar imágenes rectangulares, ha sido indispensable alterar el número de filtros de la proyección final. De esta manera, se permite configurar el número de clases sobre las que predice el modelo, mediante un parámetro más en la llamada al script: **--num_classes**.
- Se ha eliminado la última operación que se realiza en el Uformer *vanilla*, una suma del valor de la entrada con la salida del *Output Projection*. En el contexto de la segmentación semántica sería contraproducente, puesto que se perdería el concepto de "máscara".
- Se ha añadido el parámetro **--segmentation**, que señala cuándo se quieren realizar tareas de segmentación, facilitando la modularización del código. En lugar de reemplazar la arquitectura original, se añaden nuevas funcionalidades.

Configuración del Modelo

El modelo del Uformer, cuya arquitectura se muestra en la figura 4.7, ha sido entrenado durante 250 épocas con la siguiente configuración de hiperparámetros:

1. Tanto en el encoder como en el decoder se utilizan **4 bloques** de 2 ***LeWinTransformerBlocks*** cada uno, donde se aplica la MW-MSA con un tamaño de ventana de 8×8 píxeles.
2. Los **parches** tienen un tamaño de 384×192 y son mapeados a ***patch embeddings***. La **dimensionalidad** de estos aumenta en cada bloque del encoder por un factor de 2, siendo de 16 en el primero y de 256 en el último.
3. Se aplica normalización por capas (***LayerNorm***) y no se utiliza *dropout*. El **weight decay**, sin embargo, toma un valor de 0,02 en el optimizador ***AdamW***.
4. Al igual que en la nnUNet, se utiliza el planificador ***PolyLRScheduler*** para un ***learning rate*** inicial de 0,001.

5. El **batch size** se establece a 4.

Cabe destacar que, gracias a la detección automática de características por parte de la nnUNet, el proceso de búsqueda e iteración de modelos se ha simplificado considerablemente. Dicho *framework* realiza los cálculos necesarios para establecer un conjunto de hiperparámetros óptimo para su contexto.

Sin embargo, como se puede comprobar, algunos valores han tenido que modificarse mediante un proceso tedioso de prueba y error para garantizar el mejor rendimiento posible de la red. Concretamente, acertar con la tasa de aprendizaje no fue nada fácil, pues tomando el valor propuesto por la nnUNet de 0,01 (que funcionó perfectamente en dicho modelo), sufría de explosión de gradiente e imposibilitaba el entrenamiento.

Algo similar ocurrió con el *batch size*, pues el máximo de ejemplos que cumplieron simultáneamente en memoria fue tres veces menor al de la nnUNet. La explicación detrás de esto es que, pese a ser imágenes de menor tamaño, el peso del modelo del Uformer es bastante mayor, dejando menor espacio para las imágenes de entrada.

4.3. Pipeline de Pre-entrenamiento mediante el Aprendizaje Auto Supervisado

En el presente apartado se describen las diferentes fases que conforman el *pipeline* auto supervisado, desde el pre-entrenamiento hasta el ajuste fino, proporcionando detalles de implementación de todos los modelos entrenados.

4.3.1. Preparación de los Datos

El conjunto de datos utilizado tanto para el pre-entrenamiento discriminativo como para el generativo, es el **HyperKvasir** [120]. Contiene un total de 99417 imágenes sin etiquetar del tracto gastroscópico capturadas por un laparoscopio, lo que le convierte en el *dataset* público más grande de su categoría. En la figura 4.10 se muestran algunos ejemplos de su contenido.

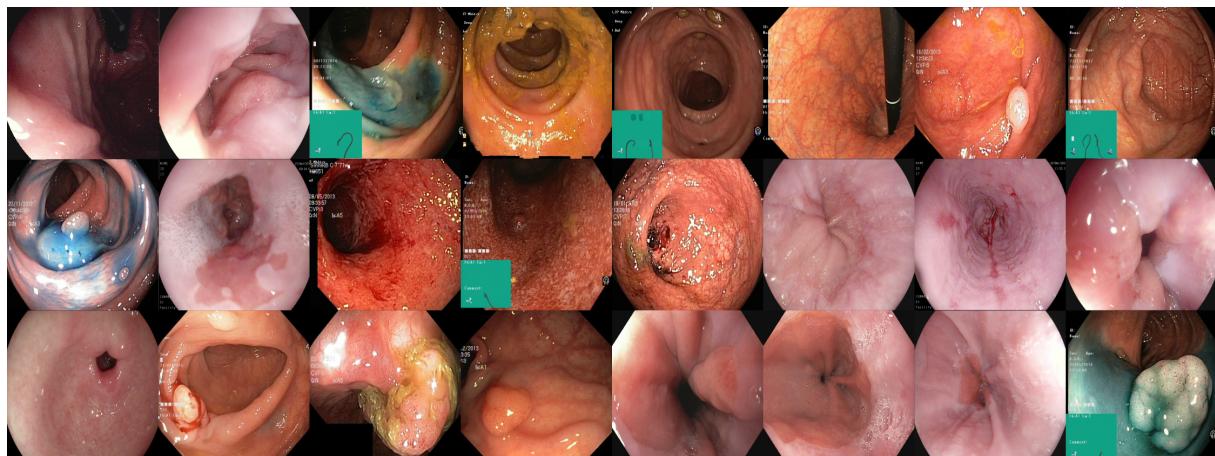


Figura 4.10. Ejemplo del contenido del *dataset* HyperKvasir [120].

Su elección está fundamentada en tres motivos principales:

1. **Correlación con CholecSeg8k:** Ambos conjuntos de datos han sido obtenidos en operaciones parecidas, utilizando el mismo aparato para su

captura. Esto implica que se mantendrá cierta consistencia en la forma y contenido de las imágenes, garantizando que sus distribuciones sean similares (véase, por ejemplo, que en los dos *datasets* aparece el mismo fondo negro en las esquinas, provocado por el angular de la cámara del laparoscopio).

2. **Variedad:** Para comprender la naturaleza intrínseca de los datos, el modelo necesita exponerse a la mayor variedad posible de imágenes que le puedan ser útiles. En este sentido, el HyperKvasir proporciona una amplia gama de ejemplos provenientes de diferentes partes del tracto gastrointestinal y de diversas condiciones patológicas.
3. **Tamaño del dataset:** Como ya se mencionó en el apartado 2.3.1, un requisito indispensable para aplicar la técnica de *fine tuning* es disponer de un conjunto de datos origen considerablemente mayor que el conjunto de datos objetivos. Con el HyperKvasir contamos con un número de imágenes suficiente para cumplir esta regla (la proporción es de aproximadamente 12 a 1).

Por otra parte, tal y como se puede observar en la figura 4.11, las imágenes de este *dataset* no comparten la misma resolución. Las redes neuronales suelen admitir únicamente datos con un tamaño consistente, por lo que se ha optado por su redimensionamiento.

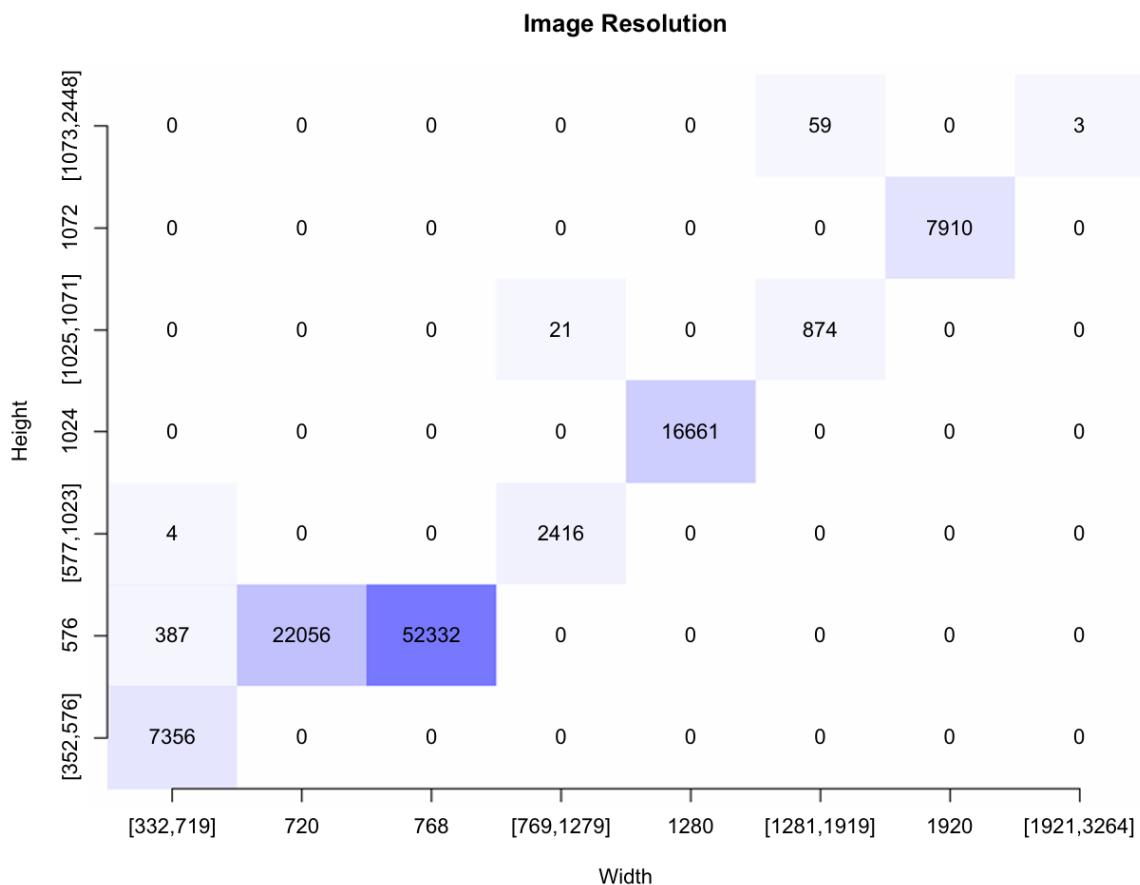


Figura 4.11. Resolución de las imágenes del HyperKvasir [120].

Concretamente, cumpliendo con las particularidades de la red discriminativa, (apartado 4.3.3), se ha decidido que su tamaño final sea de 1024×512 . De este modo, se limita la pérdida de información intrínseca a este proceso y se garantiza una fácil reducción dimensional en las distintas capas de la red (al ser una resolución perfectamente rectangular con potencias de 2).

4.3.2. Estructura General

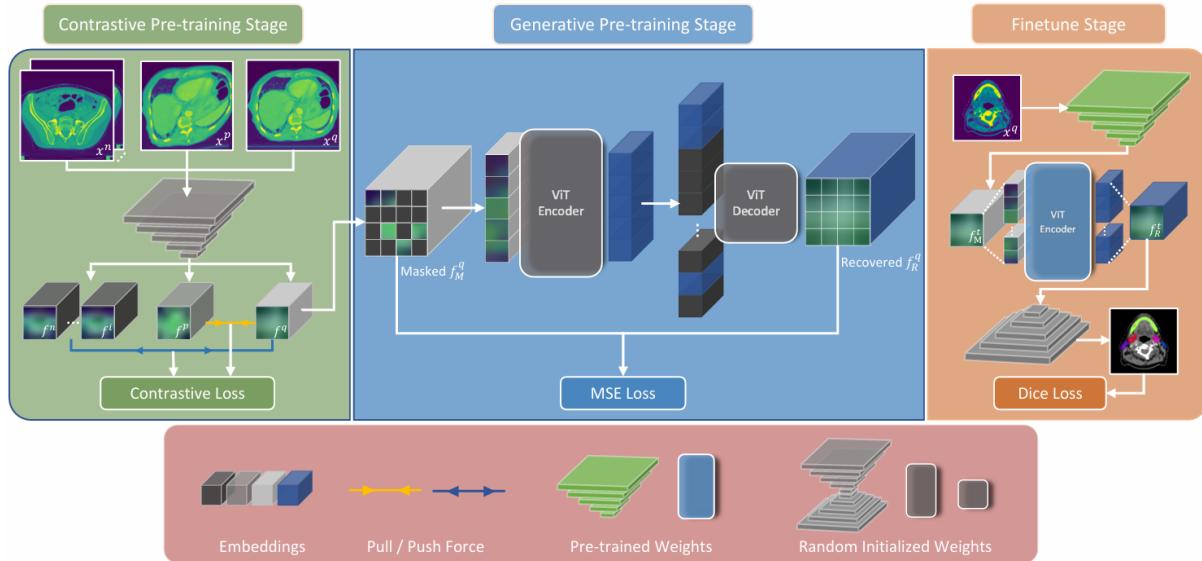


Figura 4.12. Arquitectura RepRec [119].

La estructura general del pipeline de Aprendizaje Auto Supervisado se presenta en la figura 4.12. Está basada en el artículo de conferencia **Representation Recovering for Self-Supervised Pre-training on Medical Images** publicado por Xiangyi Yan et. al en 2023 [119]. En él, se propone un framework llamado **RepRec**, formado por dos fases de pre-entrenamiento y una fase final de ajuste fino.

Su objetivo principal consiste en mejorar el análisis de imágenes médicas mediante la integración de técnicas de SSL discriminativas y generativas. La combinación de estos dos enfoques permite al modelo aprovechar tanto la extracción de características detalladas, (fase discriminativa) como la comprensión semántica global (fase generativa), para poder manejar grandes conjuntos de datos no etiquetados.

Una de las motivaciones principales para elegir esta arquitectura como base, han sido los prometedores resultados reportados en comparación con el resto de modelos SSL discriminativos y generativos (tabla 4.5). Algunos de ellos son considerados actualmente como *State Of The Art* (SOTA) en tareas de clasificación, pero no han obtenido tan buenos resultados en segmentación semántica.

Dataset	ABD-110			Thorax-85			HaN		
	T = 1	T = 10	T = 50	T = 1	T = 10	T = 50	T = 1	T = 10	T = 50
Baseline									
Inicialización Aleatoria	47.08	74.32	79.64	50.75	84.73	87.66	37.16	55.94	75.45
ImageNet	50.03	80.47	83.39	53.77	85.74	89.47	40.74	69.56	76.84
Pre-training Contrastivo									
MoCo.[94]	50.02	81.25	83.20	52.90	86.34	89.42	40.25	67.07	76.68
Wang et al.[126]	49.23	81.03	83.86	52.46	86.41	89.12	40.92	59.94	75.12
Chaitanya et al.[127]	49.60	81.43	84.23	53.04	87.04	89.61	41.12	65.24	76.75
Pre-training Generativo									
MAE[97]	47.84	77.61	80.70	50.91	84.87	88.78	37.54	64.10	75.04
Wei et al.[128]	47.17	76.34	80.94	51.63	84.83	88.99	37.15	67.82	75.70
Combinación de métodos Contrastivos y Generativos									
Tian et al.[81]	49.90	81.45	84.16	52.22	86.86	89.74	40.78	70.24	77.92
RepRec	50.31	81.89	84.67	53.97	87.01	90.37	41.99	71.71	77.31

Tabla 4.5. Resultados comparativos de diferentes métodos sobre distintos datasets[119].

4.3.3. Fase Discriminativa

En esta sección se describen de manera minuciosa las particularidades de la fase discriminativa, incluyendo todos los detalles sobre la arquitectura utilizada y la configuración del modelo.

Arquitectura

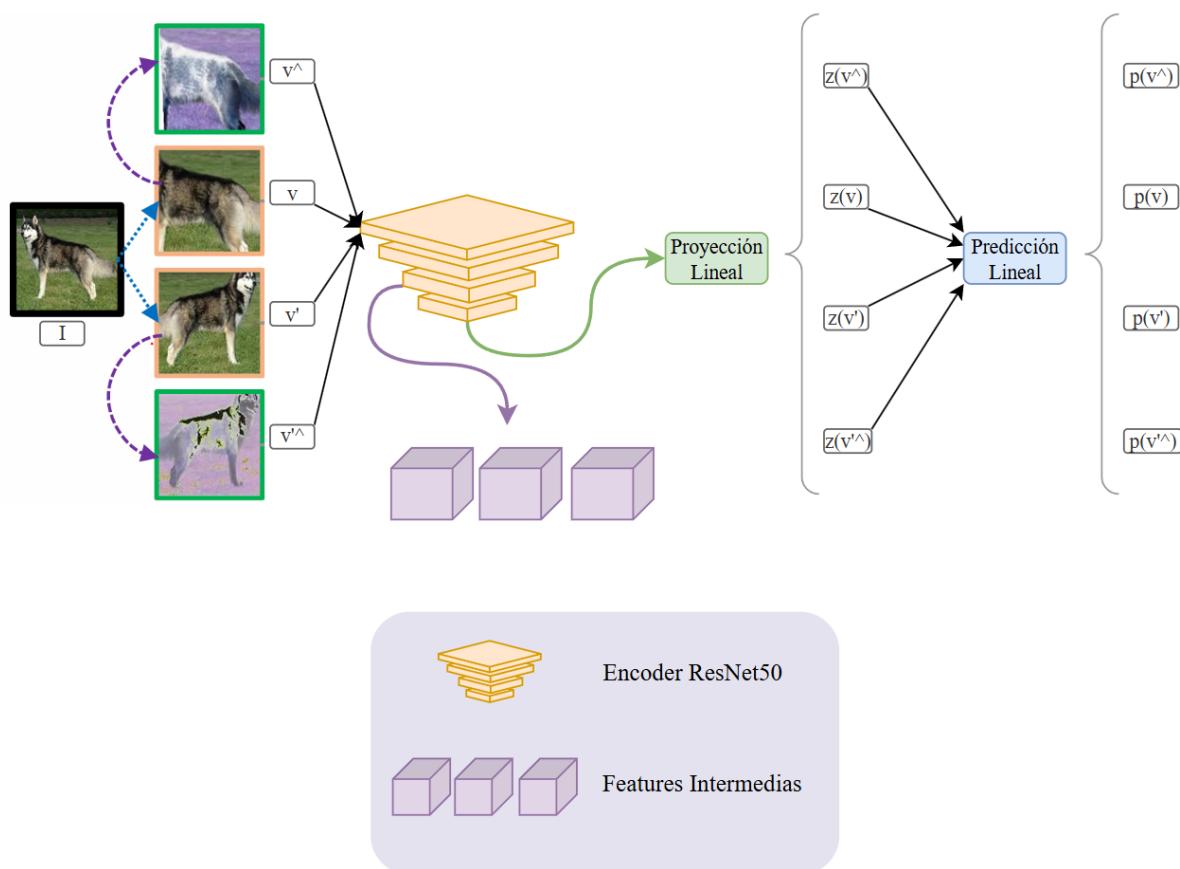


Figura 4.13. Arquitectura Discriminativa.

En esta primera fase se utiliza el paradigma **DSSL** (apartado 2.3.5) para desarrollar un entrenamiento contrastivo sobre la arquitectura **SimSiam** (apartado 2.3.4). Esto supone entrenar a un *encoder* convolucional, en este caso el de la **ResNet50** (apartado 2.2.2), para maximizar la similitud entre distintas vistas v de la misma imagen. Tal y como se muestra en la figura 4.13, por cada imagen de entrada se obtienen cuatro vistas: dos *standard* v, v' y dos *heavy*

\hat{v}, \hat{v}' .

Dichas vistas se introducen al *encoder* de la ResNet50, para ser posteriormente mapeadas a un espacio latente mediante una proyección lineal. De esta forma se obtienen las representaciones $z(v), z(v'), z(\hat{v}), z(\hat{v}')$, que son pasadas por un cabezal de predicción lineal, resultando en los vectores $p(v), p(v'), p(\hat{v}), p(\hat{v}')$.

Para calcular la pérdida contrastiva definida en la ecuación (4.7), se utiliza la similitud coseno negativa de manera simétrica entre v y v' , y de manera asimétrica entre \hat{v} y \hat{v}' .

$$\begin{aligned} negcos(p, z) &= -\frac{p \cdot z}{\|p\| \|z\|} & (4.6) \\ \mathcal{L}_{Simetrica} &= \frac{negcos(p(v), z(v'))}{2} + \frac{negcos(p(v'), z(v))}{2} \\ \mathcal{L}_{Asimetrica} &= \frac{negcos(p(\hat{v}), z(v))}{2} + \frac{negcos(p(\hat{v}'), z(v'))}{2} \\ \mathcal{L}_{Contrastiva} &= \mathcal{L}_{Simetrica} + \mathcal{L}_{Asimetrica} & (4.7) \end{aligned}$$

Con el fin de entrenar al modelo generativo en la siguiente fase (apartado 4.3.4), se necesita obtener una serie de **feature maps** de las imágenes de entrada (no de sus vistas). Para ello, se desvía la salida de una de las últimas capas del *encoder* convolucional y se persiste.

Seleccionar correctamente en qué capa extraer estas *features* intermedias tiene una importancia crítica, puesto que nos interesa que contengan la mayor información posible sin perder demasiado contexto espacial, ajustándose a los límites establecidos por los recursos computacionales disponibles. De esta manera, se deben sopesar estos tres aspectos con el objetivo de determinar las dimensiones adecuadas para esta tarea en concreto.

En este sentido, el proceso de búsqueda y optimización de este parámetro se ha visto simplificado en gran medida, gracias a que ha sido posible contactar con los autores del artículo [119]. De esta forma, he podido conocer de primera mano que en su implementación original, utilizaban *feature maps* de tamaño 16×16 . Siguiendo esta intuición, como las imágenes de nuestro dataset HyperKvasir están redimensionadas a una forma rectangular, me he decantado finalmente por elegir un tamaño de $32 \times 16 \times 512$ que supone un

equilibrio óptimo entre coste y rendimiento.

Configuración del Modelo

El modelo discriminativo se ha entrenado durante 250 **épocas** con el siguiente conjunto de hiperparámetros:

- El **optimizador** elegido ha sido el **Stochastic Gradient Descent** (SGD) con un **Momentum** de 0,9, respetando la configuración por defecto del DSSL.
- Pese a no incluir capas de *dropout*, se ha usado un **weight decay** de 0,0001. Además, tanto en la proyección como la predicción lineal se incluyen capas de **BatchNorm** inmediatamente después de las capas densas.
- El **learning rate** se ha fijado a 0,01, sin utilizar ningún tipo de planificador.
- Como por cada imagen se generan cuatro vistas mediante *data augmentation*, se ha reducido el número de los datos de entrada a 24000, ya que a partir de ellos se generarán 96000 imágenes artificiales.
- Se ha utilizado un **batch size** de 16.

Una vez entrenado el encoder convolucional , se han almacenado sus pesos con el objetivo de cargarlos en la fase final de *fine tuning*. Así se conseguirá transferir todo el conocimiento obtenido y se favorecerá la correcta segmentación de los datos de entrada.

4.3.4. Fase Generativa

En esta segunda fase del pre-entrenamiento, pese a ser independiente de la primera, se necesitan las *features* intermedias que esta produce. A continuación se describen todas la peculiaridades de la arquitectura utilizada y la configuración del modelo elegida para su entrenamiento.

Arquitectura

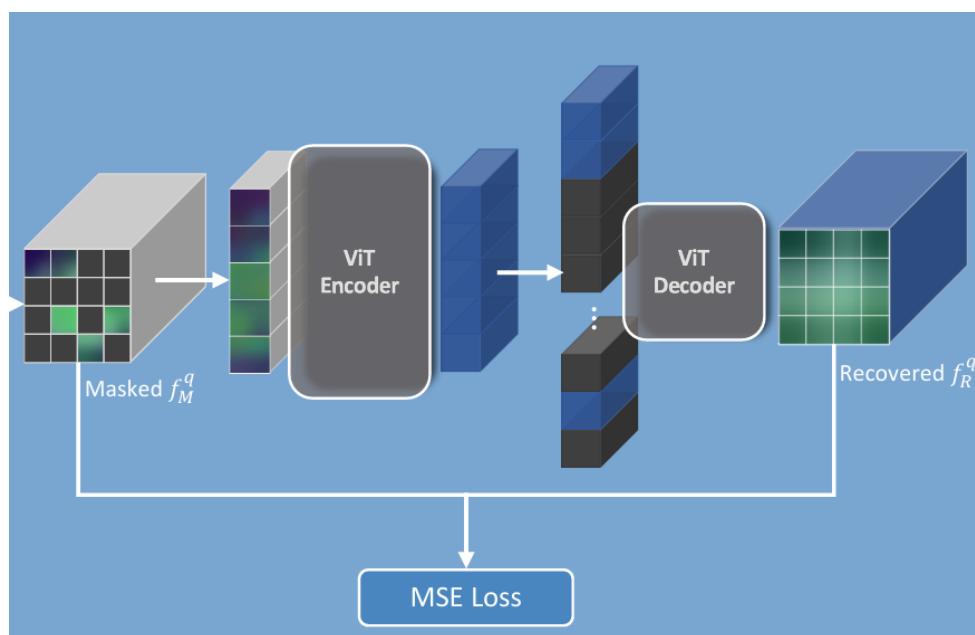


Figura 4.14. Arquitectura Generativa [119].

La arquitectura generativa se basa en una implementación de los **MAE** (apartado 2.3.4), como se muestra en la figura 4.14. Originalmente, los MAE se entrena para reconstruir un conjunto de parches enmascarados provenientes de una imagen. Este proceso les permite eliciar abundante información de alto nivel, lo cual resulta muy útil al transferirse a tareas de clasificación y detección. No obstante, este beneficio es marginal cuando son evaluados en tareas densas y complejas como la segmentación semántica.

Por este motivo, en lugar de alimentar a la red generativa con las imágenes originales, se utilizan los **feature maps** obtenidos de la primera etapa para su

pre-entrenamiento. Así, se consigue obtener información de bajo nivel muy relevante que servirá de gran ayuda en el contexto de la tarea objetivo.

Además, para preservar la información espacial contenida en estas *features*, los parches no enmascarados no se convierten en vectores, sino que se utilizan directamente como entrada para el *encoder generativo*. La figura 4.15 ilustra este proceso.

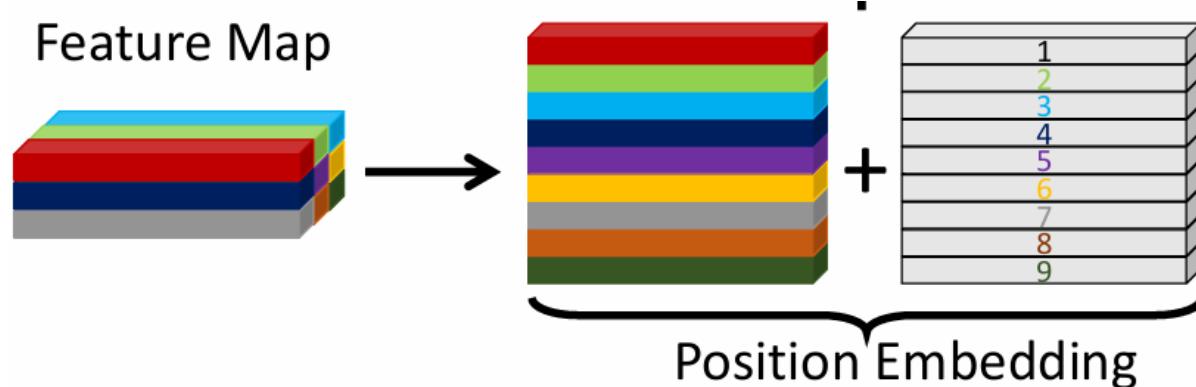


Figura 4.15. Entrada de los *feature maps* al *encoder generativo* [129].

La salida de dicho *encoder* profundiza la representación de los *feature maps* que toma como entrada, conservando sus dimensiones originales. En este sentido se ha optado por emplear **SimpleViT** en lugar del ViT propuesto en RepRec, debido a las mejoras de rendimiento reportadas en [130] y su menor complejidad.

Posteriormente, un pequeño *decoder* se encarga de reconstruir los parches enmascarados de las *features*, minimizando la pérdida MSE descrita en la ecuación (2.4).

Configuración del Modelo

Las adaptaciones al modelo base han ajustado el diseño para cumplir con las limitaciones de recursos disponibles, permitiendo el entrenamiento de la red durante 800 épocas con los siguientes hiperparámetros:

- El **optimizador** elegido ha sido el **AdamW** con un **weight decay** de 0,0001.

- No incluye capas de *dropout*, pero se ha utilizado normalización por capas (**LayerNorm**) a modo de regularización.
- El **learning rate** se ha fijado a 0,01, sin utilizar ningún tipo de planificador.
- Debido a las reducidas dimensiones de los *feature maps* que recibe como entrada, se ha podido utilizar un **batch size** de 64.

Dentro de toda esta estructura se aprovechará únicamente el *encoder* SimpleViT ya pre-entrenado, cuya integración dentro de la arquitectura final se expondrá a continuación. Los pesos del modelo se han almacenado para posibilitar transferir el conocimiento de la red mediante una carga sencilla en tiempo de ejecución.

4.3.5. Ajuste Fino

Este apartado define el proceso mediante el cual se ha ensamblado el *pipeline* de pre-entrenamiento auto supervisado, en un modelo final sobre el que se aplicará *fine tuning*. De este modo, se exponen las características de dicha red y su configuración durante el entrenamiento.

Ensamblado del Pipeline

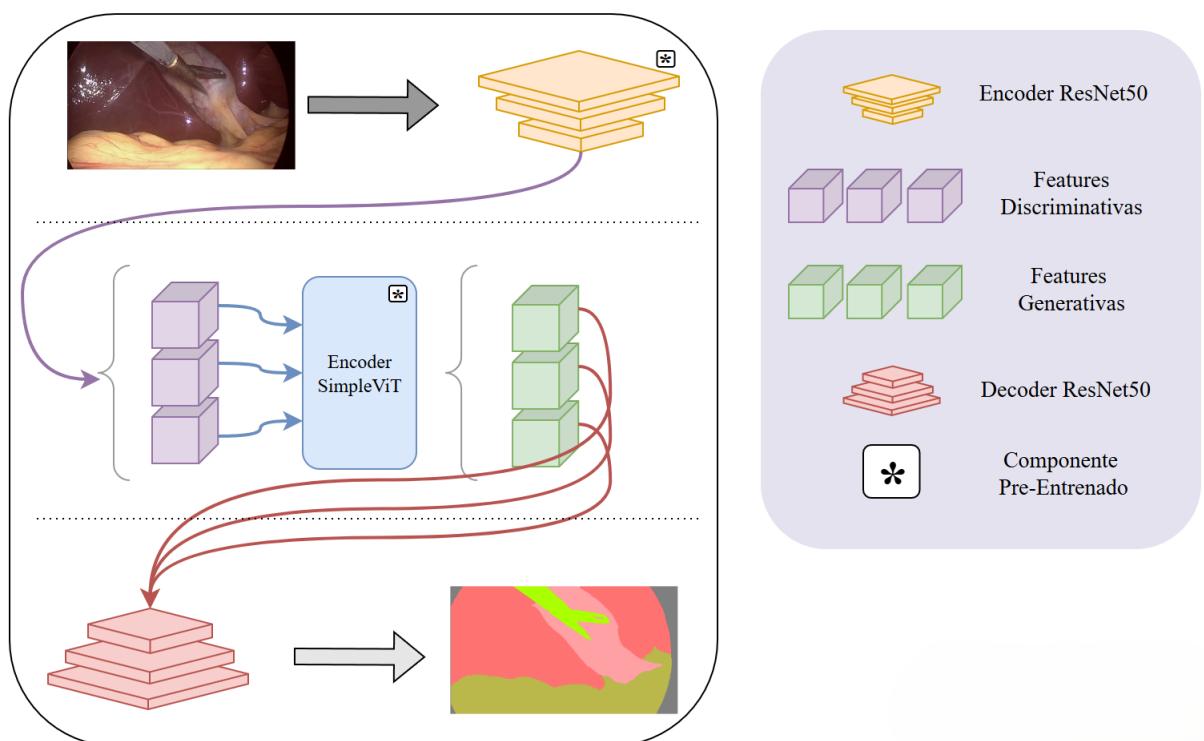


Figura 4.16. Arquitectura Modelo Final.

Una vez completadas las dos fases descritas previamente, se obtienen como resultado un **Encoder ResNet50** (Convolucional) y un **Encoder SimpleViT** (Transformer) ya pre-entrenados. Estos son utilizados en la arquitectura del modelo final junto a un **Decoder ResNet50** aleatoriamente inicializado, para segmentar las imágenes del dataset **CholecSeg8k**.

En la figura 4.16, las imágenes del conjunto de entrenamiento son procesadas por el encoder ResNet50, generando un conjunto de *feature maps* de

$32 \times 16 \times 512$. Estos se utilizan como entrada para el *encoder SimpleViT*, que produce representaciones más complejas y profundas debido a su capacidad para conseguir una comprensión semántica global más amplia. Los resultados del *encoder SimpleViT* constituyen el espacio latente del modelo, representando la información más comprimida y rica.

Posteriormente, estos *feature maps* se introducen en el *decoder ResNet50*, el cual está inicializado aleatoriamente. El *decoder* tiene la tarea de reconstruir la imagen de entrada a su tamaño original mientras realiza la segmentación por clases. Además, sus capas están conectadas mediante conexiones residuales a las capas correspondientes del *encoder ResNet50*, facilitando así el flujo de información y preservando las características relevantes durante la reconstrucción de la imagen.

Por último, se calcula el valor de la pérdida del mismo modo que durante el entrenamiento de los *baselines*, es decir, utilizando la suma ponderada entre la CE y la *Dice Loss* (4.2). De este modo, se garantiza que los tres modelos se pueden comparar de manera satisfactoria, permitiendo sacar conclusiones fehacientes.

Configuración del Modelo Final

El **Modelo Final** obtenido tras concluir las tres fases del *pipeline*, ha sido entrenado también durante 250 épocas, utilizando los siguientes hiperparámetros:

- Al realizar un proceso de **ajuste fino**, se ha permitido que el gradiente fluya por todos los parámetros del modelo. De este modo, los componentes pre-entrenados se han ajustado a esta nueva problemática, mientras que el *decoder ResNet50* ha sido entrenado de 0.
- El **learning rate** ha sido ajustado época a época por el planificador **PolyLRScheduler**, tomando un valor inicial de 0,001, ya que al igual que en el modelo **Uformer**, ocurría una explosión de gradiente con valores más altos.
- Se han utilizado capas de **BatchNorm** y **LayerNorm** para la regularización.

- El optimizador elegido ha sido el **AdamW**, con un **weight decay** de 0,0001.
- Se ha establecido el valor del **batch size** a 12, siendo el máximo valor posible dentro de las restricciones de memoria.

5.

Resultados

En este capítulo se muestran de manera conjunta los resultados obtenidos a partir de los distintos entrenamientos realizados. Además, se ilustran gráficos e imágenes que exponen de manera clara y concisa las características de cada modelo, permitiendo sacar conclusiones con ciertas garantías. En lugar de dividir este apartado por modelos, se ha optado por comparar categoría por categoría a las tres redes, comentando los puntos más importantes.

Para que dicha comparación entre los *baselines* y el modelo SSL sea justa, se ha utilizado la métrica del **Dice Coefficient**, que, como se explicó en el apartado [4.2.2](#), es crucial para controlar el rendimiento del modelo.

Además, con el objetivo de analizar profundamente la productividad de los modelos, se introduce la métrica **IoU** (*Intersection Over Union*), que mide la superposición de las etiquetas predichas y los *ground truths*, respecto al número total de instancias de ambas (figura [5.1](#)).

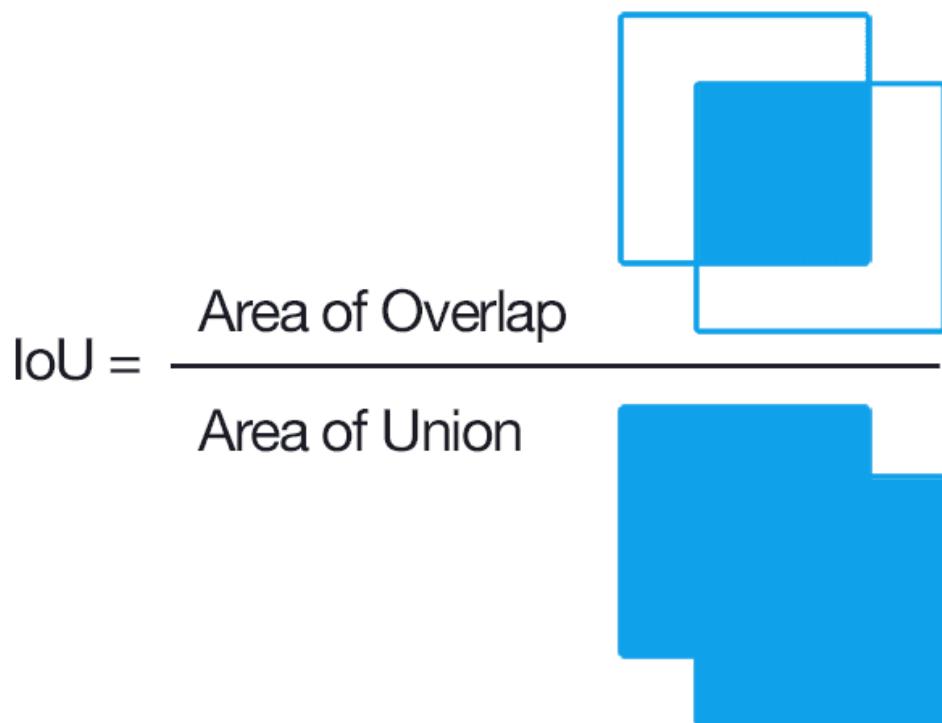


Figura 5.1. Intersección sobre la Unión (IoU) [131].

Estas métricas, pese a ser útiles para poder realizar un análisis objetivo, no siempre se corresponden al rendimiento del modelo, ya que no existe una métrica perfecta para calcular qué tan coherente es una imagen o su estructura. De esta manera, puede ocurrir que las etiquetas de la segmentación estén perfiladas de forma que en algunos puntos aparezcan *outlayers*, y que el modelo sea recompensado por dichas métricas si los memoriza.

Curvas de aprendizaje

Las curvas de aprendizaje de los modelos difieren ligeramente entre sí, pese a mostrar en todas ellas la misma tendencia (al alza, en el caso del DC, y a la baja en caso de la pérdida). La mejor manera de ilustrar la métrica *Dice* ha sido mostrando su valor absoluto (línea discontinua verde en las gráficas), y su media móvil (línea verde sólida), para permitir una visualización contrastada y tener perspectiva de los cambios.

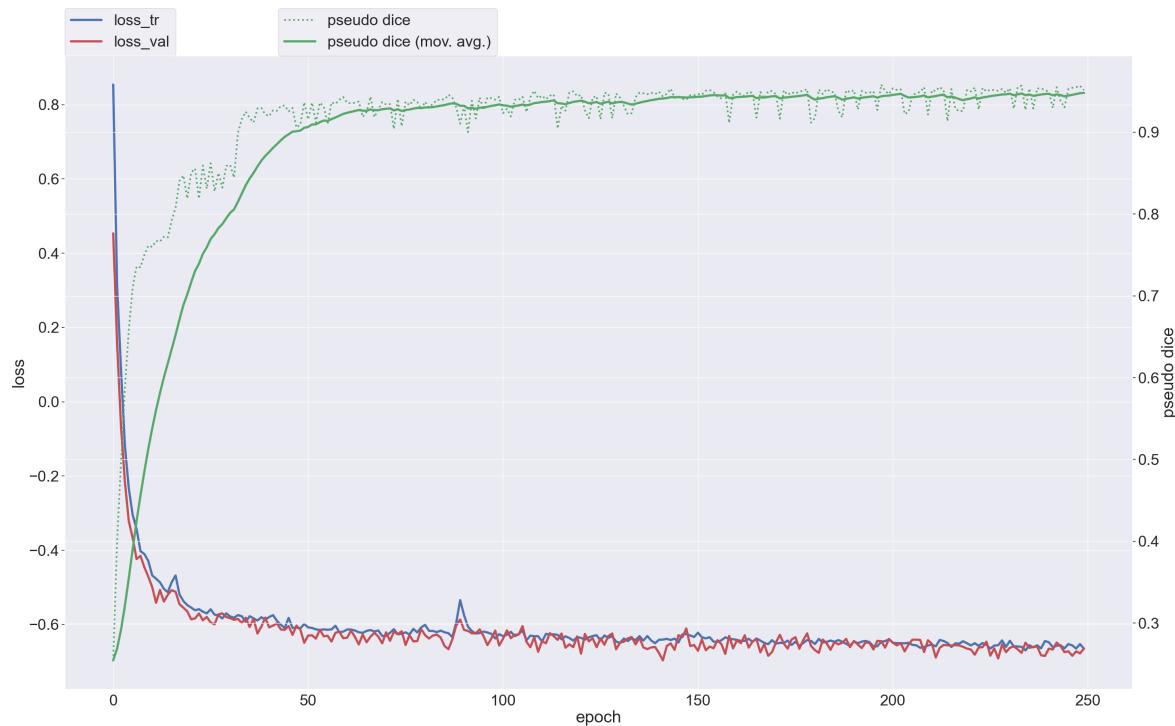


Figura 5.2. Curva de aprendizaje del nnUNet.

Quizá, de todas las gráficas, la que presenta la nnUNet (figura 4.8) es la que menos fluctuaciones sufre. Esto se debe a que es el único modelo completamente convolucional, que tiene de manera natural menor "hambre" de datos que sus competidores. Además, muestra una regularización más potente, que permite que las pérdidas de los conjuntos de entrenamiento (azul) y de validación (rojo) sean prácticamente idénticos.

No obstante, este modelo es el que más épocas ocupa para superar la barrera del DC del 0,9, tardando concretamente 36 épocas.

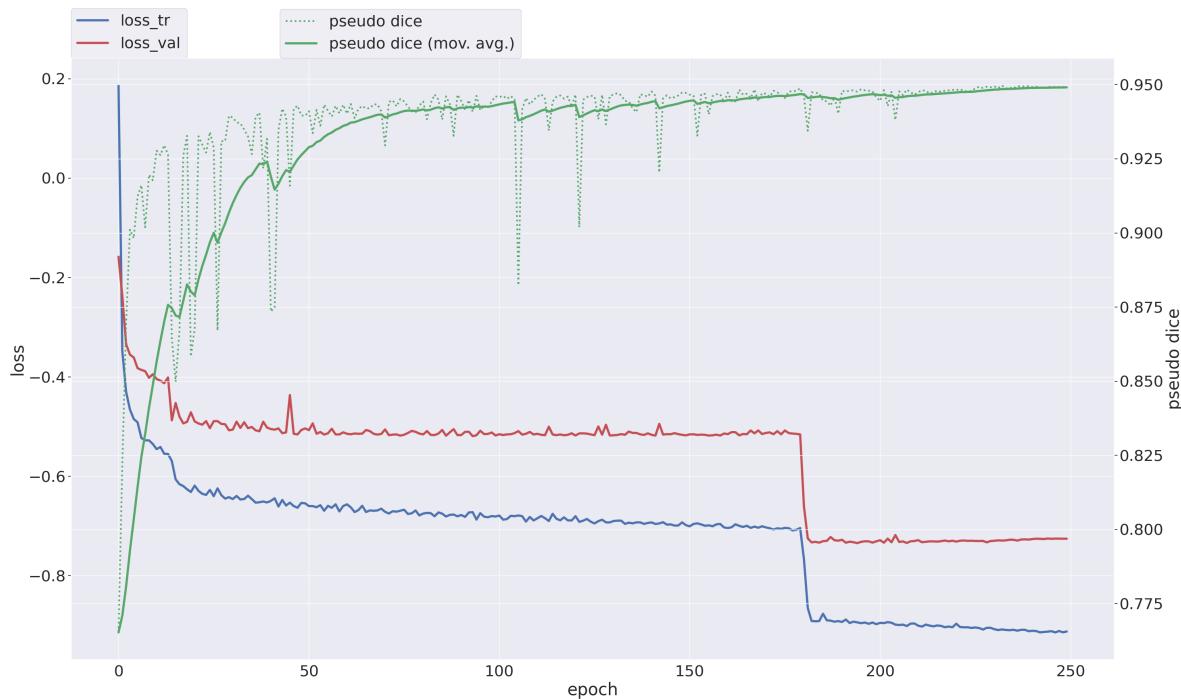


Figura 5.3. Curva de aprendizaje del **Uformer**.

Por otra parte, los pronunciados picos presentes en el DC durante el entrenamiento del Uformer (figura 5.3), denotan la necesidad que tienen los transformers de contar con un *dataset* grande. Además, se puede observar una ligera varianza entre la pérdida de entrenamiento y la de validación, que va aumentando a medida que avanzan las épocas. Esto es un primer indicio de *Overfitting*, que si bien ahora no es un problema, en el futuro habría que tomar medidas para subsanarlo.

Cabe destacar la caída de varias décimas de la pérdida, que ocurre alrededor de la época 175. Es un suceso curioso, puesto que no está relacionado con un aumento proporcional del DC, de hecho, este último se mantiene estable y no se ve inmutado. La razón detrás de este acontecimiento no está muy clara, pero podría haber sido provocada por la salida en ese punto de un mínimo local en la curva del gradiente.

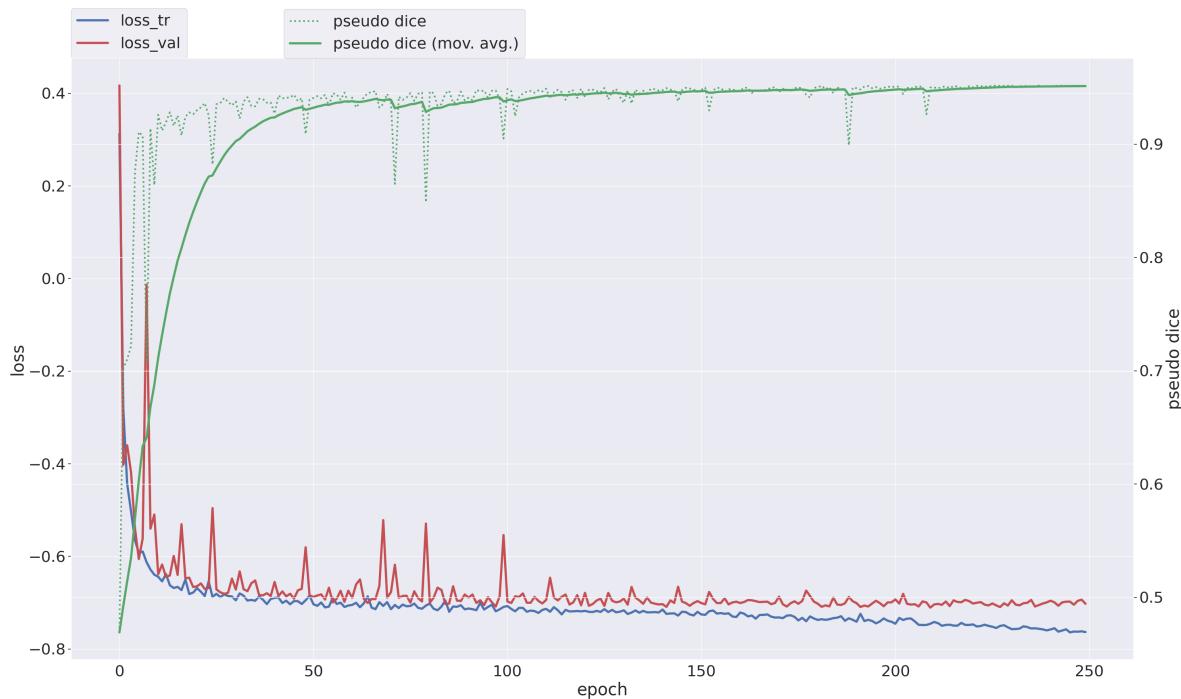


Figura 5.4. Curva de aprendizaje de RepRec.

Por último, la gráfica del modelo RepRec (figura 5.4) se encuentra a medio camino de los dos diagramas anteriores, lo que tiene sentido, al contar con partes convolucionales y partes de transformer. Pese a que también observamos picos en el DC y en la pérdida de validación, estos se van estabilizando a medida que avanza el entrenamiento. Presenta además poca varianza entre las pérdidas de entrenamiento y validación, a pesar de que en las últimas épocas se ve acrecentada por centésimas.

Este modelo (RepRec) es el que más rápidamente consigue mantenerse estable por encima de un valor de 0,9 del DC, necesitando menos de 25 épocas para lograrlo. Aquí tiene mucho peso la etapa de pre-entrenamiento, que le permite ganar cierta ventaja respecto al resto de competidores.

Evaluación del rendimiento

Para contrastar detalladamente los resultados y las producciones de los modelos, se considera importante proporcionar estadísticas globales y por clases, que permitan compararlos de manera exhaustiva.

En las siguientes figuras se muestran las **matrices de confusión normalizadas** de la nnUNet (5.5), el Uformer(5.6) y RepRec(5.7). El valor de cada celda representa el porcentaje de veces que las predicciones del modelo cayeron en la combinación específica de una clase verdadera (fila) y una clase predicha (columna), en relación con el total de instancias de la clase verdadera correspondiente.

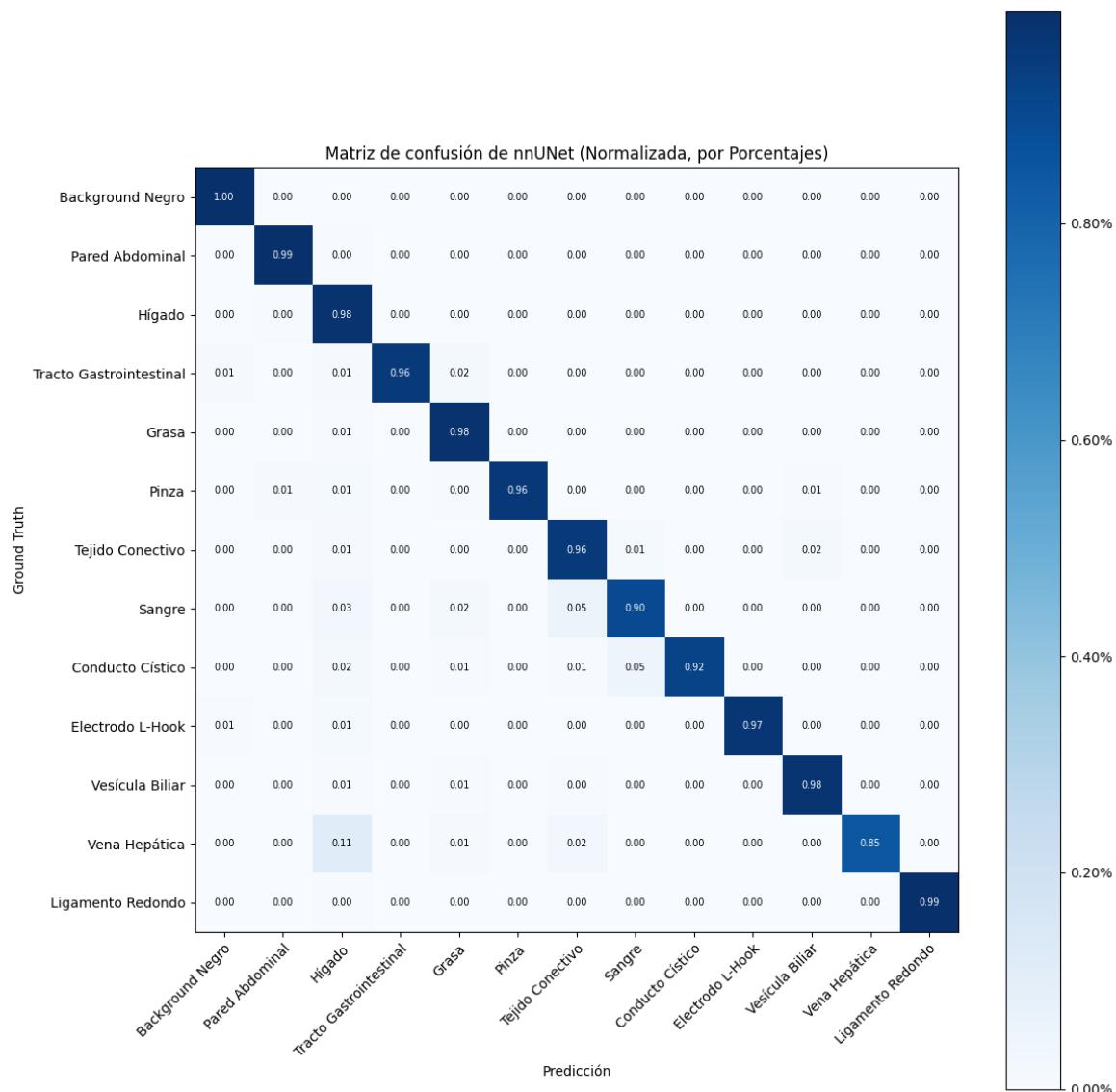


Figura 5.5. Matriz de Confusión Normalizada de la nnUNet.

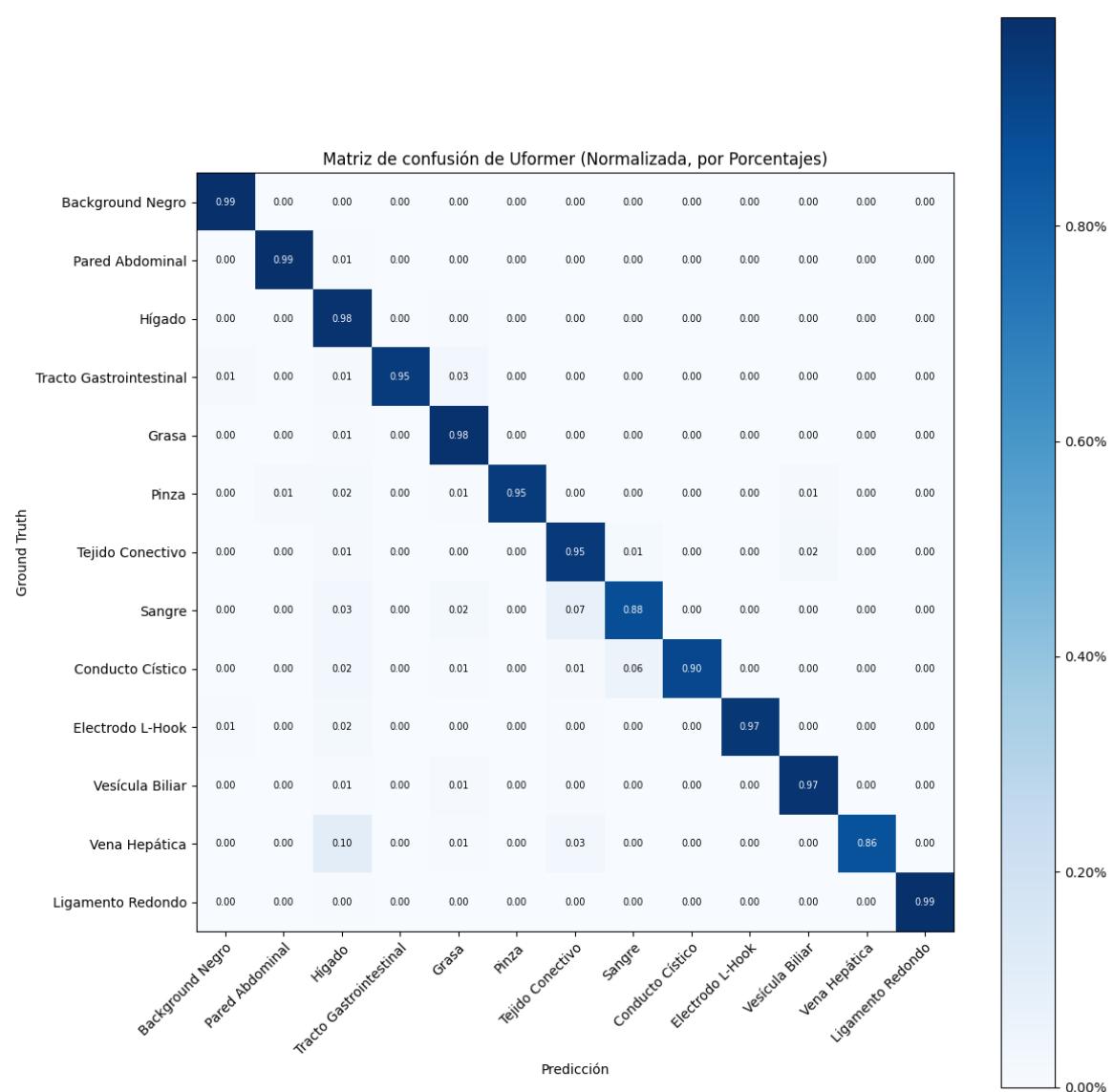


Figura 5.6. Matriz de Confusión Normalizada del **Uformer**.

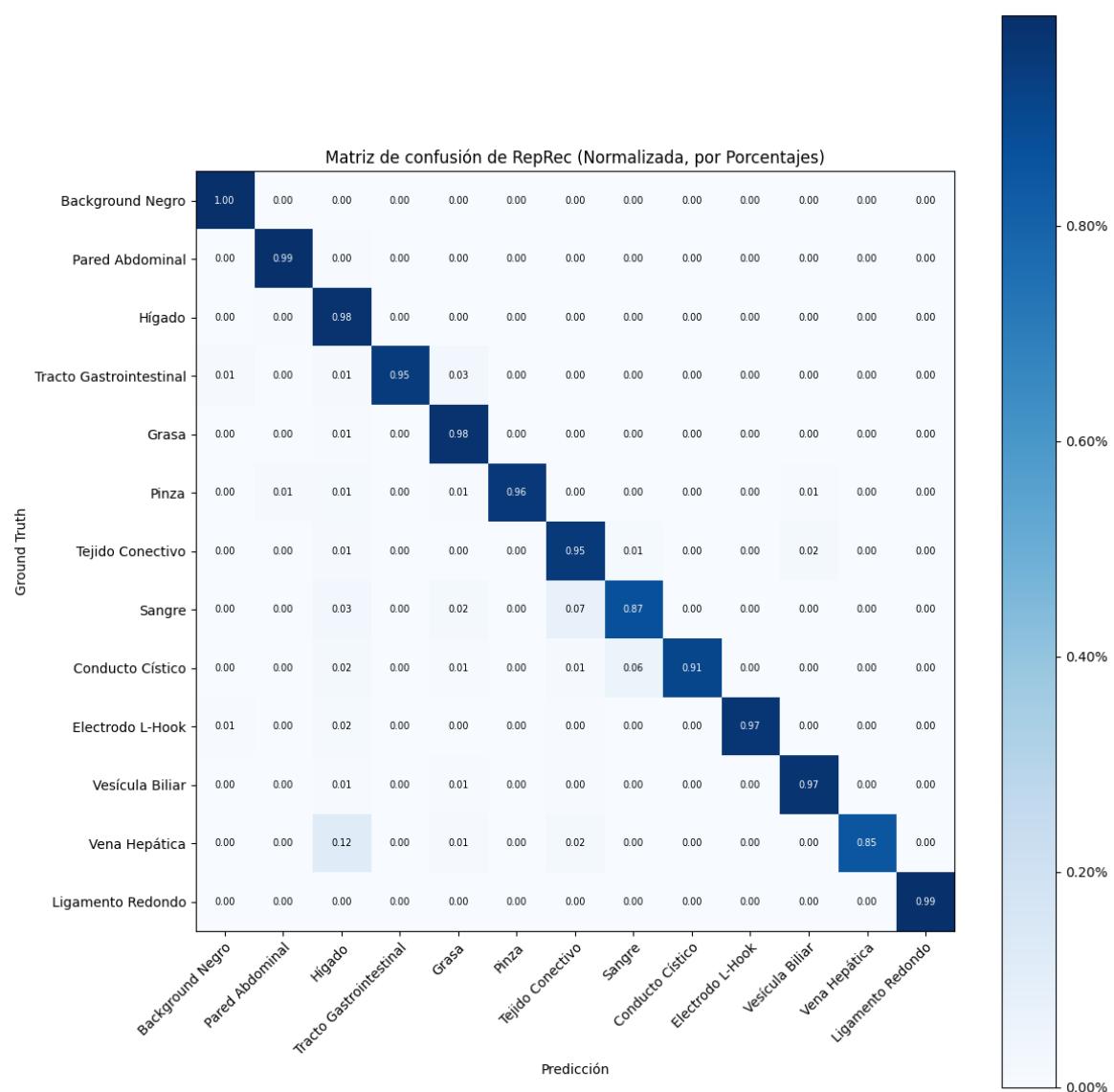


Figura 5.7. Matriz de Confusión Normalizada de RepRec.

Las diferencias entre todos ellos son mínimas, puesto que consiguen predecir correctamente y de forma consistente. Aunque esto es poco esclarecedor para decidir qué modelo es mejor, es un indicativo de que la planificación y la construcción de los modelos es la adecuada.

De esta forma, prácticamente la única zona coloreada en las matrices se corresponde con la diagonal principal. Quizá, la única ligera excepción se encuentra en el 0,11 % de los casos donde el *ground truth* es la Vena Hepática, que es confundida por el Hígado. Esto puede ocurrir porque esta vena, como

su propio nombre indica, se encuentra justamente en el hígado¹, y el modelo puede no estar de acuerdo con las etiquetas sobre qué pixel exacto establecer la frontera entre ellos.

Análisis de las Principales Características

A continuación, en la tabla 5.1, se exponen las **características globales** más destacables de estos modelos.

Modelo	Learning Rate	Épocas	Tiempo	Coeficiente Dice	IoU
nnUNet	0.01	250	~72 Horas	0.948	0.893
Uformer	0.001	250	~54 Horas	0.949	0.881
RepRec (SSL)	0.001	250	~43 Horas	0.951	0.861

Tabla 5.1. Características Globales.

Como aspecto destacable, llama mucho la atención la **velocidad** con la que la arquitectura RepRec completa el entrenamiento, necesitando únicamente de 43 horas para terminar las 250 épocas. Esto significa una **diferencia** del 20,37% respecto al Uformer, y del 40,28% en relación a la nnUNet. Además, pese a obtener un valor levemente inferior en la IoU, consigue mejorar ligeramente los resultados de sus competidores en la métrica del DC, situándose por encima del 0,95, siendo 1 el valor máximo.

Esta rapidez de la arquitectura RepRec se consigue gracias al pragmatismo de su diseño, el cual permite alimentar al *encoder* con representaciones de un tamaño muy reducido. Si tenemos en cuenta que dicho modelo fue el que menos épocas tardó en superar el 0,9 en DC, se puede calcular que, aproximadamente, necesitó solo 4 horas para ello (utilizando una Tesla T4).

Por otro lado, en la tabla 5.2 se incluye el valor del coeficiente *Dice* para las distintas clases del problema.

¹La vena hepática se encarga de drenar la sangre del hígado y transportarla hacia la vena cava inferior, que luego lleva la sangre de vuelta al corazón.

	nnUnet		Uformer		RepRec	
Clase	<i>DC</i>	<i>IoU</i>	<i>DC</i>	<i>IoU</i>	<i>DC</i>	<i>IoU</i>
Pared Abdominal	0.978	0.963	0.978	0.962	0.972	0.959
Hígado	0.982	0.965	0.977	0.956	0.980	0.961
Tracto Gastrointestinal	0.926	0.879	0.914	0.862	0.917	0.869
Grasa	0.978	0.959	0.973	0.951	0.971	0.951
Pinzas	0.948	0.905	0.931	0.878	0.924	0.881
Tejido Conectivo	0.958	0.920	0.944	0.905	0.906	0.868
Sangre	0.846	0.750	0.838	0.735	0.818	0.717
Conducto Cístico	0.889	0.827	0.805	0.739	0.817	0.762
Electrodo L-Hook	0.969	0.945	0.957	0.930	0.934	0.912
Vesícula Biliar	0.951	0.920	0.947	0.911	0.945	0.913
Vena Hepática	0.786	0.695	0.855	0.761	0.819	0.729
Ligamento Redondo	0.990	0.981	0.991	0.981	0.911	0.903

Tabla 5.2. Coeficiente Dice e IoU por clases.

Como se puede observar en la tabla anterior, todos los modelos consiguen reconocer todas las clases, obteniendo resultados satisfactorios en la mayoría de ellas, a excepción de casos puntuales. Las clases con menor representación presentan unos valores de DC e IoU ligeramente inferiores a las más abundantes, pero en ningún caso bajan de 0,78 y 0,69, respectivamente.

Ejemplos de Segmentación

Por último, como se muestra en la figura 5.8, podemos observar que, a simple vista, los resultados producidos por los modelos son prácticamente intercambiables. Se podría decir incluso, que en muchas ocasiones las predicciones tienen menos ruido que las propias etiquetas en los bordes que delimitan las clases.

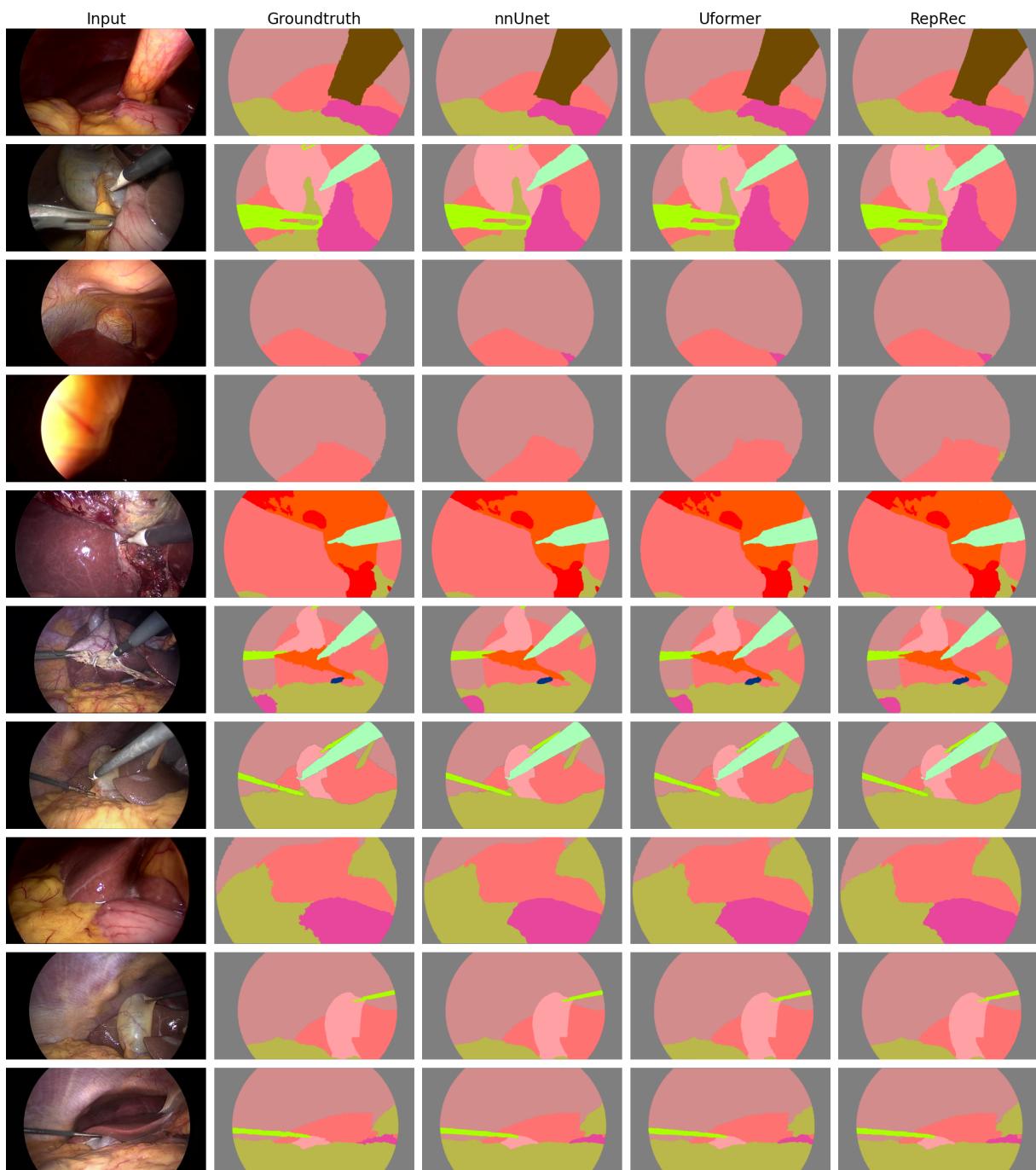


Figura 5.8. Predicciones de Segmentación sobre CholecSeg.

En el Anexo A se incluyen más ejemplos de las segmentaciones producidas por las distintas redes entrenadas en el transcurso de este PFG.

6.

Impacto del Proyecto

La integración del aprendizaje Auto Supervisado en la medicina, plantea una serie de consideraciones que trascienden el ámbito técnico y clínico. Este apartado analiza tanto las implicaciones sociales, éticas y medioambientales de la adopción de estas tecnologías avanzadas, como las consecuencias empresariales y económicas, evaluando los beneficios y los retos que presentan.

6.1. Impacto social, responsabilidad ética y medioambiental

El uso de la ayuda asistida de la segmentación de imágenes en las cirugías laparoscópicas tiene un impacto social significativo que incide de forma directa en el campo de la medicina. La implementación de métodos auto supervisados en este contexto permite una **precisión y eficiencia** sin precedentes, lo que se traduce en **mejores resultados para los pacientes y una optimización de los recursos médicos**. Este avance tecnológico no solo facilita los diagnósticos y tratamientos, sino que también **reduce la carga de trabajo** de los profesionales médicos, permitiéndoles centrarse en tareas más complejas y detalladas [132].

Por otro lado, la adopción de este método ayuda a **democratizar el acceso a cuidados de salud avanzados**. Herramientas como la desarrollada en este PFG pueden ser implementadas en áreas con recursos limitados, donde la presencia de especialistas en imagenología es escasa. Un estudio de la Organización Mundial de la Salud (OMS) [133] destacó que la implementación de tecnologías de IA en clínicas rurales de países en desarrollo ha permitido realizar diagnósticos precisos en tiempo real, mejorando significativamente la calidad de la atención sanitaria en estas regiones. Esto contribuye a una mayor equidad en la atención médica, asegurando que más personas puedan be-

neficiarse de diagnósticos precisos y tratamientos oportunos, independientemente de su ubicación geográfica [134, 135].

El despliegue de la IA en la medicina plantea también importantes consideraciones éticas. La precisión y el poder predictivo de estos modelos pueden llevar a decisiones automatizadas que afecten a la vida de los pacientes, por lo que se convierte en necesidad la adopción de una **estricta supervisión**. Es crucial asegurar que estos sistemas no perpetúen sesgos preexistentes en los datos de entrenamiento, lo que podría conducir a diagnósticos erróneos o tratamientos inadecuados para ciertos grupos poblacionales. El cumplimiento de normativas de privacidad, como el Reglamento General de Protección de Datos (RGPD) en la Unión Europea, es esencial para proteger la información sensible de los pacientes y asegurar que su uso se limita a fines médicos autorizados [136]. La implementación de estas regulaciones y la adopción de buenas prácticas en el manejo de datos son fundamentales para construir un entorno de confianza y seguridad en la adopción de esta tecnología.

En términos medioambientales, la segmentación automatizada **reduce drásticamente la generación de residuos hospitalarios hasta en un 50 %**, debido a la eliminación de procedimientos innecesarios o duplicados [132]. De esta manera, disminuye significativamente la cantidad de desechos médicos, reduciendo así el impacto ambiental de esta práctica.

La implementación de esta tecnología también plantea importantes consideraciones en términos de huella de carbono, debido al gran consumo de recursos computacionales necesarios para entrenar y operar estos sistemas. La infraestructura digital mundial, incluyendo servidores y centros de datos, consume una cantidad considerable de energía, contribuyendo aproximadamente al 1% de las emisiones globales de gases de efecto invernadero. El uso de técnicas de aprendizaje auto supervisado **optimiza el uso de recursos computacionales**, reduciendo el consumo de energía durante el entrenamiento de modelos. Al disminuir la demanda de recursos computacionales intensivos, el proyecto contribuye a una **menor huella de carbono** asociada con el procesamiento de datos médicos.

Por último, el proyecto apoya varios Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 [137]. En particular, contribuye al ODS 3 (Salud y Bienestar) al mejorar la calidad de los servicios de salud, y al ODS 9 (Industria, Innovación e Infraestructura) al impulsar la innovación tecnológica en el ámbito médico.

Además, al promover técnicas quirúrgicas menos invasivas y más eficientes, también contribuye al ODS 12 (Producción y Consumo Responsables).

6.2. Impacto económico y empresarial

Desde el punto de vista económico, la implementación de técnicas de aprendizaje auto supervisado en la atención médica está respaldado por una serie de beneficios tangibles.

En primer lugar, la IA ayuda a **reducir los costos operativos hospitalarios** al automatizar tareas rutinarias y administrativas. Esto no solo disminuye la carga de trabajo del personal médico, sino que también minimiza errores y optimiza el uso de recursos, aspecto que ya se ha tratado en el impacto social y medioambiental.

Por otro lado, la adopción de IA también impulsa la **innovación empresarial** y la **creación de empleo**. La demanda creciente de especialistas en esta tecnología está promoviendo el desarrollo de nuevas startups y tecnologías médicas, fortaleciendo así la economía y generando nuevas oportunidades profesionales [138].

En conclusión, la inteligencia artificial no solo está mejorando la calidad del cuidado de los pacientes y la eficiencia operativa en la atención médica, sino que también está transformando positivamente el panorama económico y empresarial del sector, promoviendo la innovación y el desarrollo sostenible en la industria de la salud.

7. Conclusiones y Proyección a futuro

En base a los resultados obtenidos, se puede afirmar que **se han cumplido todos los objetivos** propuestos al comienzo de este proyecto. El profundo análisis realizado sobre la naturaleza de la técnica laparoscópica, ha sido crucial a la hora de especializar diversas arquitecturas en la tarea de segmentación semántica. Además, se ha llevado a cabo una extensa labor de investigación sobre los principales modelos supervisados y los fundamentos del paradigma de aprendizaje auto supervisado, que ha permitido obtener resultados excepcionales en tres arquitecturas distintas: dos *baselines* supervisados (**nnUNet** y **Uformer**) y un pipeline de pre-entrenamiento auto supervisado (basado en **RepRec**). En concreto, el rendimiento de todas estas redes ha alcanzado valores del Coeficiente *Dice* en torno al 0,95, mostrando segmentaciones muy precisas que reflejan correctamente las clases de entrada.

Por otro lado, los factores diferenciales entre dichas arquitecturas han sido la velocidad del entrenamiento y el número de épocas necesarias para alcanzar una buena efectividad. En este sentido, el modelo auto supervisado ha demostrado ser superior a sus competidores supervisados, reduciendo estos tiempos hasta en un 40 % y necesitando un número de iteraciones menor para aprender a clasificar cada píxel correctamente. De esta forma, la **hipótesis principal** de este proyecto es **aceptada**, justificando el desarrollo de una arquitectura que combina ideas de distintos ámbitos, para posibilitar la integración del paradigma auto supervisado en tareas complejas como la segmentación de imágenes médicas.

Sin embargo, la infrarrepresentación de algunas clases en las imágenes de entrenamiento, ha provocado ciertas discrepancias entre sus valores reales y las predicciones de los modelos, que no son capaces de mantener consistentemente una precisión muy alta al segmentarlas (a diferencia de las clases con multitud de ejemplos). Esto ocurre debido al limitado tamaño de los

conjuntos de datos de entrenamiento, que no son comparables con los utilizados por las grandes empresas tecnológicas, pese a exprimir al máximo los **recursos disponibles** actualmente. En concreto, la alta escalabilidad del modelo auto supervisado permitiría aprovechar nuevas imágenes de entrada para mejorar su comprensión de la naturaleza de las imágenes extraídas por un endoscopio. Además, las limitaciones en cuanto a **capacidad de cómputo** han sido latentes en todas las etapas del proyecto, ralentizando los entrenamientos y afectando negativamente a la velocidad para iterar entre distintas versiones.

Adicionalmente, se presentan una serie de **posibilidades a futuro** que complementarían el trabajo realizado en este PFG:

- En primer lugar, sería interesante la implementación de distintas metodologías auto supervisadas, con el fin de obtener una arquitectura óptima para la tarea de segmentación. Esto nos permitiría descubrir qué componentes concretos son más adecuados y qué nuevos conceptos se podrían aplicar.
- Continuando el camino marcado, sería de provecho contrastar el rendimiento de estas arquitecturas sobre imágenes provenientes de otras partes del cuerpo, extraídas mediante escáneres o radiografías. De esta manera, se podría comprobar la capacidad de generalización de cada una de ellas a distribuciones de datos que nunca han visto.
- Para contrarrestar el **desbalanceo** de las clases, en una versión posterior se podrían aplicar técnicas de preprocesamiento de las imágenes que las equilibren de manera artificial. También se podrían endurecer las políticas de **regularización** para disminuir las posibilidades de *overfitting* y mejorar su generalización.
- En última instancia, se desearía aplicar este sistema auto supervisado en hospitales y centros quirújicos, para aportar una herramienta fiable y sólida que ayude a los profesionales a preparar, realizar y evaluar las distintas operaciones que se llevan a cabo. Esto potencialmente se traduciría en salvar vidas y favorecer la recuperación de los pacientes, evitando heridas en zonas críticas y un mayor control del entorno.

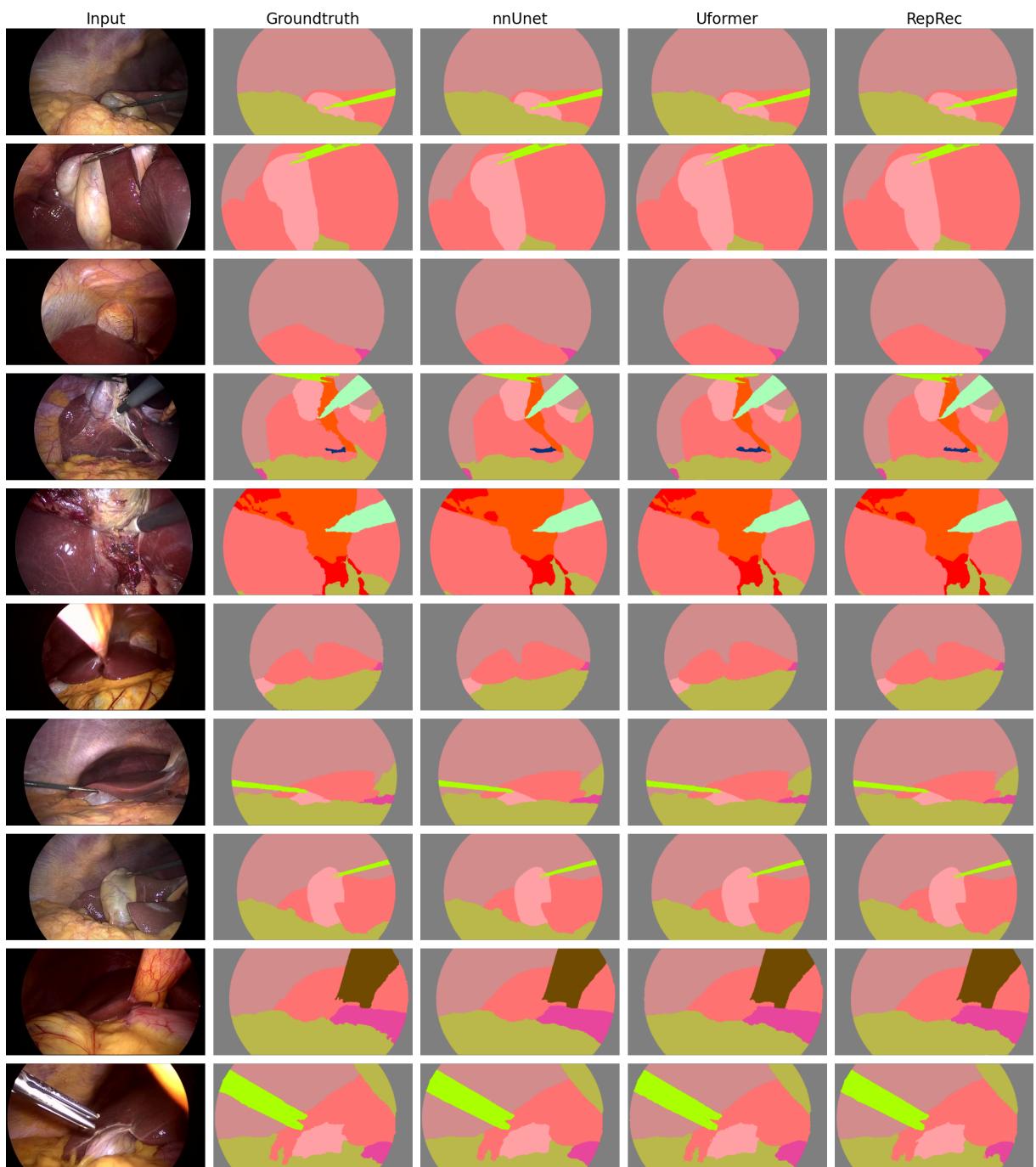
Todos estos factores sirven de gran motivación para seguir con este trabajo de investigación de cara al futuro, una vez completado el desarrollo del

PFG. De esta manera, se expresa la intención de utilizar la arquitectura auto supervisada desarrollada, como base para futuros experimentos y proyectos personales y académicos.

A. Ejemplos de Segmentaciones



Figura A.1. Predicciones de Segmentación sobre CholecSeg (A).



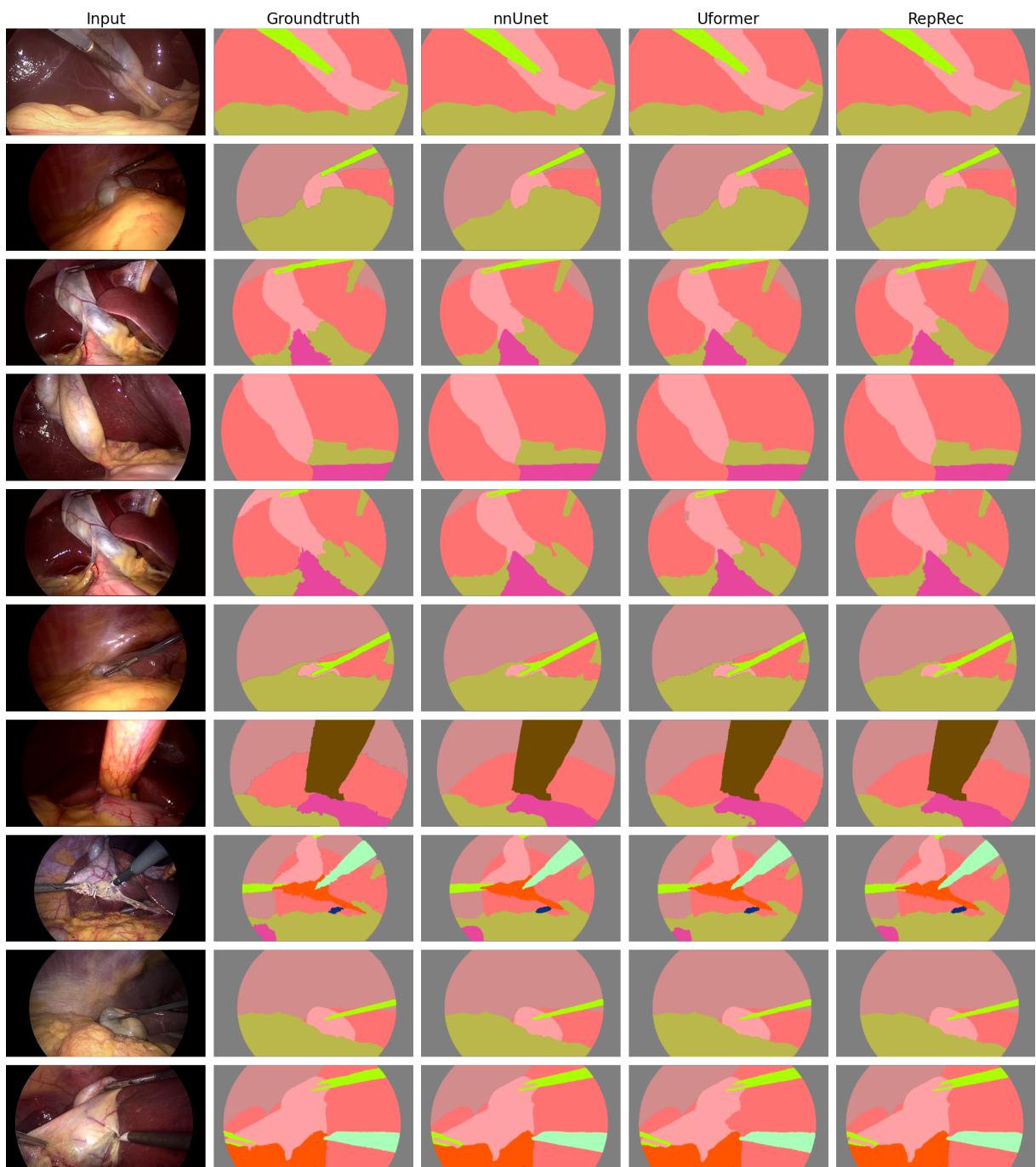


Figura A.3. Predicciones de Segmentación sobre CholecSeg (C).

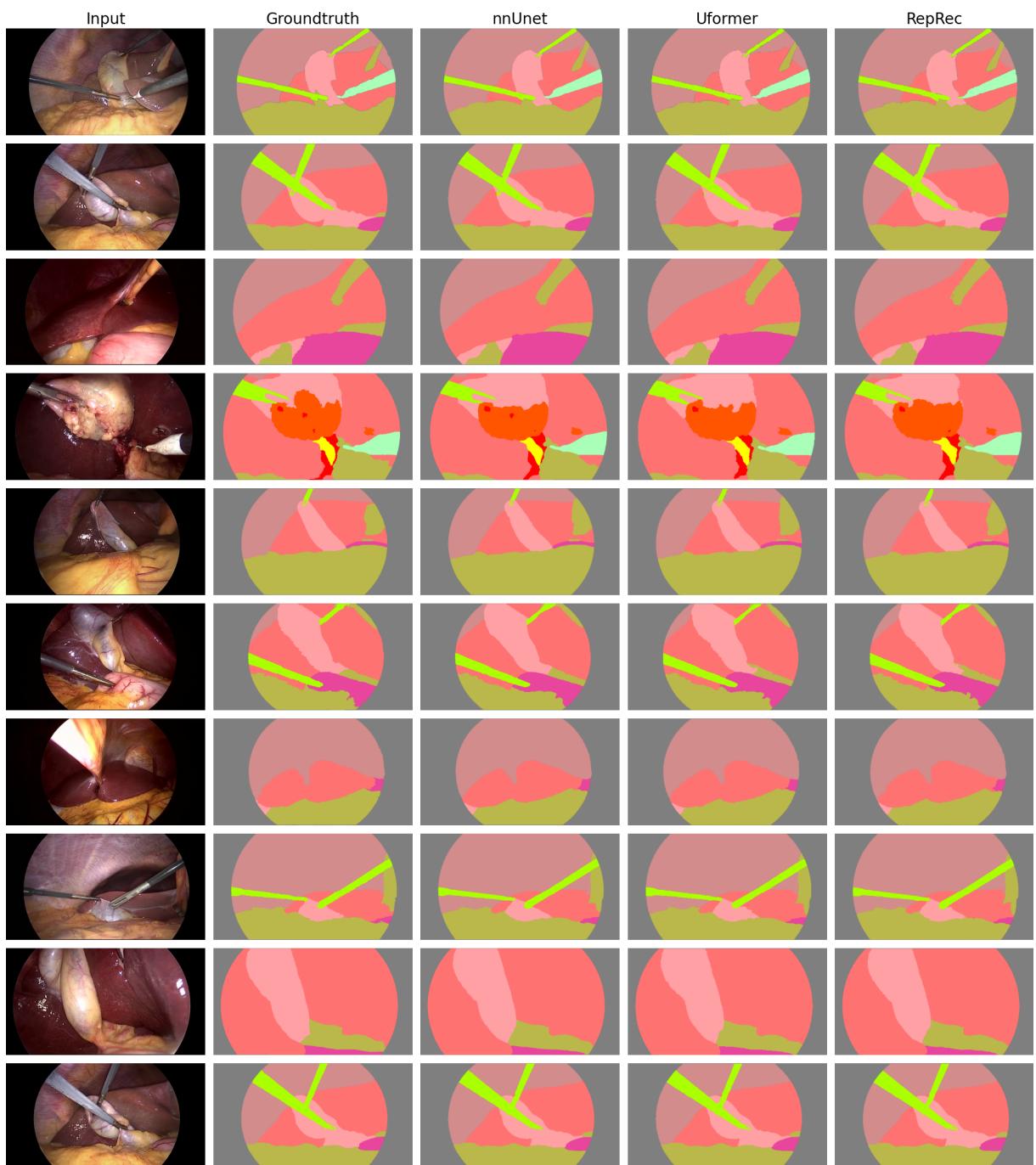


Figura A.4. Predicciones de Segmentación sobre CholecSeg (D).

B.

Repositorio

El repositorio de Github final se encuentra en:

https://github.com/Ismatse/TFG_Ismael_Tse_Perdomo_Rodriguez.git.

Índice de términos

PFG: Proyecto de Fin de Grado.

DC: Coeficiente Dice *Dice Coefficient*.

MIS: Cirugía Mínimamente Invasiva (*Minimally Invasive Surgery*).

CV: Visión por Computador (*Computer Vision*).

IA: Inteligencia Artificial.

LLM: Gran Modelo de Lenguaje (*Large Language Model*).

SSL: Aprendizaje Auto Supervisado (*Self Supervised Learning*).

DL: Aprendizaje Profundo (*Deep Learning*).

NCI: Instituto Nacional del Cáncer (*National Cancer Institute*).

NOTES: Cirugía Endoscópica Transluminal a través de Orificios Naturales (*Natural orifice translumenal endoscopic surgery*).

ML: Aprendizaje Automático (*Machine Learning*).

NN: Red Neuronal (*Neural Network*).

ReLU: Unidad Linear Rectificada (*Rectified Linear Unit*).

MAE: Error Absoluto Medio (*Mean Absolute Error*).

MSE: Error Cuadrático Medio (*Mean Squared Error*).

BCE: Entropía Cruzada Binaria (*Binary CrossEntropy*).

ADAM: Estimación de Momento Adaptativo (*Adaptive Moment Estimation*).

SGD: Descenso de Gradiente Estocástico (*Stochastic Gradient Descent*).

BN: Normalización por Lotes (*Batch Normalization*).

CNN: Red Neuronal Convolucional (*Convolutional Neural Network*).

FC: Red Densa (*Fully Connected network*).

NLP: Procesamiento del Lenguaje Natural (*Natural Language Processing*).

FF: Capa de Propagación Directa (*FeedForward layer*).

MSA: Auto Atención Multi Cabezal (*Multi-head Self-Attention*).

ViT: Vision Transformer.

MLP: Perceptrón Multi Capa (*Multi Layer Perceptron*).

FCN: Red Completamente Convolucional (*Fully Convolutional Network*).

AE: *AutoEncoder*.

SL: Aprendizaje Supervisado (*Supervised Learning*).

UL: Aprendizaje No Supervisado (*Unsupervised Learning*).

RL: Aprendizaje por Refuerzo (*Reinforcement Learning*).

MAE: *Masked AutoEncoders*.

DSSL: Aprendizaje Auto Supervisado Direccional (*Directional Self Supervised Learning*).

W-MSA: Auto Atención Multi Cabezal por Ventanas (*Window-based Multi-head Self-Attention*).

MW-MSA: Auto Atención Multi Cabezal por Ventanas Modularizada(*Modularized Window-based Multi-head Self-Attention*).

LeWin: Ventanas con enfoque Local *Locally-Enhanced Window*.

CE: Entropía Cruzada Categórica (*Categorical CrossEntropy*).

RepRec: *Representation Recovering*.

SOTA: Estado del Arte (*State Of The Art*).

IoU: Intersección sobre la Unión (*Intersection Over Union*).

OMS: Organización Mundial de la Salud.

RGPD Reglamento General de Protección de Datos.

ODS Objetivos de Desarrollo Sostenible.

Bibliografía

- [1] Sachan PK Sahu SK Agrawal A. «Intraoperative difficulties in laparoscopic cholecystectomy». En: *Jurnalul de chirurgie (Iași)* 9.2 (2013), págs. 149-155. DOI: [10.7438/1584-9341-9-2-5](https://doi.org/10.7438/1584-9341-9-2-5).
- [2] S. Bodenstedt et al. «Comparative evaluation of instrument segmentation and tracking methods in minimally invasive surgery». En: *arXiv preprint arXiv:1805.02475* (2018).
- [3] Adrien Bartoli et al. «Computer assisted Minimally Invasive Surgery: Is medical Computer Vision the answer to improving laparosurgery?» En: *Medical Hypotheses* 79.6 (2012), págs. 858-863. DOI: [10.1016/j.mehy.2012.09.007](https://doi.org/10.1016/j.mehy.2012.09.007).
- [4] Debesh Jha et al. «Exploring Deep Learning Methods for Real-Time Surgical Instrument Segmentation in Laparoscopy». En: *2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)* (2021). DOI: [10.1109/BHI50953.2021.9508610](https://doi.org/10.1109/BHI50953.2021.9508610).
- [5] Instituto Nacional del Cáncer (NCI). *Diccionario de cáncer del NCI*. 2011. URL: <https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/laparoscopia>.
- [6] Ray Garry. «Laparoscopic surgery». En: *Best Practice & Research Clinical Obstetrics & Gynaecology* 20.1 (2006), págs. 89-104. DOI: [10.1016/j.bpobgyn.2005.10.003](https://doi.org/10.1016/j.bpobgyn.2005.10.003).
- [7] R. Bittner. «Laparoscopic Surgery—15 Years After Clinical Introduction». En: *World Journal of Surgery* 30 (2006), págs. 1190-1203. DOI: [10.1007/s00268-005-0644-2](https://doi.org/10.1007/s00268-005-0644-2).
- [8] Cirugía Laparoscópica. ENDOMÉXICO. 2019. URL: <https://endomexico.com.mx/laparoscopia/cirugia-laparoscopica/>.
- [9] Semm K. *Atlas of gynecologic laparoscopy and hysteroscopy*. Philadelphia: W.B. Sanders, 1977.

- [10] GARTH LOREN SPANER SHELLEY JANEand WARNOCK. «A Brief History of Endoscopy, Laparoscopy, and Laparoscopic Surgery». En: *Journal of Laparoendoscopic & Advanced Surgical Techniques* 7.6 (1997), págs. 369-373. doi: [10.1089/lap.1997.7.369](https://doi.org/10.1089/lap.1997.7.369).
- [11] M. et al. Schollmeyer Schollmeyer. «Georg Kelling (1866–1945): the root of modern day minimal invasive surgery. A forgotten legend?». En: *Arch Gynecol Obstet* 276 (2007), págs. 505-509. doi: [10.1007/s00404-007-0372-y](https://doi.org/10.1007/s00404-007-0372-y).
- [12] Martin Hatzinger et al. «Hans Christian Jacobaeus: Inventor of Human Laparoscopy and Thoracoscopy». En: *Journal of Endourology* 20.11 (2006), págs. 848-850. doi: [10.1089/end.2006.20.848](https://doi.org/10.1089/end.2006.20.848).
- [13] *I Am Number 3 or 4?: Presidential Lecture at KSELS. Scientific Figure on ResearchGate*. 2019. URL: https://www.researchgate.net/figure/German-surgeon-Erich-M-Muehe-performed-the-first-laparoscopic-cholecystectomy-in-a-human_fig1_333824336.
- [14] Periodista de Medical Tribune. «Blinddarm muss raus durchs Endoskop Wie kompliziert das wirklich ist.» En: *Medical Tribune* (1983).
- [15] Ibrahim Alkatout et al. «The Development of Laparoscopy—A Historical Overview». En: *Frontiers in Surgery* 8 (2021). doi: [10.3389/fsurg.2021.799442](https://doi.org/10.3389/fsurg.2021.799442).
- [16] Zheng Li et al. «Design of a Novel Flexible Endoscope—Cardioscope». En: *Journal of Mechanisms and Robotics* 8.5 (2016), pág. 051014. doi: [10.1115/1.4032272](https://doi.org/10.1115/1.4032272).
- [17] Brunt Pucher, Davies y N. et al. «Outcome trends and safety measures after 30 years of laparoscopic cholecystectomy: a systematic review and pooled data analysis.» En: *Surgical Endoscopy* 32 (2018), págs. 2175-2183. doi: [10.1007/s00464-017-5974-2](https://doi.org/10.1007/s00464-017-5974-2).
- [18] Blackmore, Wong y C. L. «Evolution of laparoscopy in colorectal surgery: an evidence-based review.» En: *World journal of gastroenterology* 20.17 (2014), págs. 4926-4933. doi: [10.3748/wjg.v20.i17.4926](https://doi.org/10.3748/wjg.v20.i17.4926).
- [19] *Scientific Figure on ResearchGate*. 2021. URL: https://www.researchgate.net/figure/Laparoscopic-appendectomy-past-and-present-A-B-are-original-pictures-of-the-first_fig11_357061725.

- [20] Jianmin Li et al. «Application of Improved Robot-assisted Laparoscopic Telesurgery with 5G Technology in Urology». En: *European Urology* 83.1 (2023), págs. 41-44. doi: [10.1016/j.eururo.2022.06.018](https://doi.org/10.1016/j.eururo.2022.06.018).
- [21] Constantin Virgil Negoita. *Expert systems and fuzzy systems*. Benjamin-Cummings Publishing Co., Inc., 1984.
- [22] *Machine Learning: What is ML and how does it work?* 2022. URL: <https://www.algotive.ai/blog/machine-learning-what-is-ml-and-how-does-it-work>.
- [23] Warren S McCulloch y Walter Pitts. «A logical calculus of the ideas immanent in nervous activity.» En: *The bulletin of mathematical biophysics* 5.4 (1943), págs. 115-133.
- [24] R. Pramoditha. *The concept of artificial neurons (perceptrons) in neural networks. Towards Data Science*. 2021. URL: <https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc>.
- [25] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [26] IBM. *AI vs. Machine learning vs. Deep learning vs. Neural networks*. 2024. URL: <https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>.
- [27] Datademia. ¿Qué es Deep Learning y qué es una red neuronal? Datademia. 2022. URL: <https://datademia.es/blog/que-es-deep-learning-y-que-es-una-red-neuronal>.
- [28] S. Jadon. *Introduction to different activation functions for deep learning*. Medium. 2018. URL: <https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092>.
- [29] E. Zvornicanin. *Bias update in neural network backpropagation*. Baeldung on Computer Science. 2022. URL: <https://www.baeldung.com/cs/deep-learning-bias-backpropagation>.
- [30] Meenal V. Narkhede, Prashant P. Bartakke y Mukul S. Sutaone. «A review on weight initialization strategies for neural networks». En: *Artificial Intelligence Review* 55 (2022), págs. 291-322. doi: [10.1007/s10462-021-10033-z](https://doi.org/10.1007/s10462-021-10033-z).
- [31] Diederik P. Kingma y Jimmy Ba. «Adam: A Method for Stochastic Optimization». En: *arXiv preprint arXiv:1412.6980* (2015).

- [32] James M. Lucas y Michael S. Saccucci. «Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements». En: *Technometrics* 32.1 (1990), págs. 1-12. DOI: [10.1080/00401706.1990.10484583](https://doi.org/10.1080/00401706.1990.10484583).
- [33] Moacir Antonelli Ponti et al. «Training Deep Networks from Zero to Hero: avoiding pitfalls and going beyond». En: *arXiv preprint arXiv:2109.02752* (2021).
- [34] *Bias and Variance in Machine Learning*. [www.javatpoint.com](http://www.javatpoint.com/bias-and-variance-in-machine-learning). 2024. URL: <https://www.javatpoint.com/bias-and-variance-in-machine-learning>.
- [35] Sambhav Bhandari. *Overfitting and Underfitting. The Correlation*. 2022. URL: <https://thecorrelation.in/overfitting-and-underfitting/>.
- [36] Ivet Rafegas et al. «Understanding trained CNNs by indexing neuron selectivity». En: *Pattern Recognition Letters* 136 (2020), págs. 318-325. DOI: <https://doi.org/10.1016/j.patrec.2019.10.013>.
- [37] J. Shubham. *What exactly does CNN see? - Becoming Human: Artificial Intelligence Magazine*. 2018. URL: <https://becominghuman.ai/what-exactly-does-cnn-see-4d436d8e6e52>.
- [38] Hugo Andrade. *Modelo para detectar el uso correcto de mascarillas en tiempo real utilizando redes neuronales convolucionales - Scientific Figure on ResearchGate*. 2021. URL: https://www.researchgate.net/figure/Figura-1-Descripcion-del-funcionamiento-de-una-red-neuronal-convolucional-CNN-10_fig1_348825166.
- [39] David Batista. *Convolutional Neural Networks for Text Classification*. (s/f). Davidsbatista.net. 2018. URL: <https://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/>.
- [40] Hossein Gholamalinezhad y Hossein Khosravi. «Pooling Methods in Deep Neural Networks, a Review». En: *arXiv preprint arXiv:2009.07485* (2020).
- [41] A. Mandour. *Downsampling and upsampling in CNN*. OpenGenus IQ: Learn Algorithms, DL, System Design. 2022. URL: <https://iq.opengenus.org/downsampling-and-upsampling-in-cnn/>.
- [42] Kaiming He et al. «Deep Residual Learning for Image Recognition». En: *arXiv preprint arXiv:1512.03385* (2015).

- [43] A. Kouidri. *Mastering ResNet: Deep learning breakthrough in image recognition*. Ikomia.Ai. 2023. URL: <https://www.ikomia.ai/blog/mastering-resnet-deep-learning-image-recognition>.
- [44] Razvan Pascanu, Tomas Mikolov y Yoshua Bengio. «On the difficulty of training Recurrent Neural Networks». En: *arXiv preprint arXiv:1211.5063* (2013).
- [45] Ashish Vaswani et al. «Attention Is All You Need». En: *arXiv preprint arXiv:1706.03762* (2017).
- [46] Tomas Mikolov et al. «Efficient Estimation of Word Representations in Vector Space». En: *arXiv preprint arXiv:1301.3781* (2013).
- [47] M. Schuster y K.K. Paliwal. «Bidirectional recurrent neural networks». En: *IEEE Transactions on Signal Processing* 45.11 (1997), págs. 2673-2681. DOI: [10.1109/78.650093](https://doi.org/10.1109/78.650093).
- [48] Junyoung Chung et al. «Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling». En: *arXiv preprint arXiv:1412.3555* (2014).
- [49] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-Term Memory». En: *Neural Computation* 9.8 (1997), págs. 1735-1780. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [50] J. Alammar. *The Illustrated Transformer*. Github.io. 2018. URL: <https://jalammar.github.io/illustrated-transformer/>.
- [51] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». En: *arXiv preprint arXiv:1810.04805* (2018).
- [52] OpenAI. «GPT-4 Technical Report». En: *arXiv preprint arXiv:2303.08774* (2023).
- [53] Marcin Junczys-Dowmunt et al. «Marian: Fast Neural Machine Translation in C++». En: *arXiv preprint arXiv:1804.00344* (2018).
- [54] Mike Lewis et al. «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension». En: *arXiv preprint arXiv:1910.13461* (2019).
- [55] J. Alammar. *The Illustrated GPT-2 (Visualizing Transformer Language Models)*. Github.io. 2019. URL: <https://jalammar.github.io/illustrated-gpt2/>.

- [56] Alexey Dosovitskiy et al. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». En: *arXiv preprint arXiv:2010.11929* (2021).
- [57] H. Taud y J.F. Mas. «Geomatic Approaches for Modeling Land Change Scenarios». En: Springer International Publishing, 2018. Cap. Multilayer Perceptron (MLP), págs. 451-455.
- [58] Zihao Wang y Lei Wu. «Theoretical Analysis of the Inductive Biases in Deep Convolutional Networks». En: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., 2023, págs. 74289-74338.
- [59] Ze Liu et al. «Swin Transformer: Hierarchical Vision Transformer using Shifted Windows». En: *arXiv preprint arXiv:2103.14030* (2021).
- [60] Prajit Ramachandran et al. «Stand-Alone Self-Attention in Vision Models». En: *arXiv preprint arXiv:1906.05909* (2019).
- [61] Han Hu et al. «Local Relation Networks for Image Recognition». En: *arXiv preprint arXiv:1904.11491* (2019).
- [62] Xiao Xiao et al. «A Swin Transformer-Based Encoding Booster Integrated in U-Shaped Network for Building Extraction». En: *Remote Sensing* 14 (2022). DOI: [10.3390/rs14112611](https://doi.org/10.3390/rs14112611).
- [63] Gaurav Sharma. *Semantic segmentation: A complete guide. Towards AI*. 2021. URL: <https://towardsai.net/p/l/machine-learning-7>.
- [64] Pedro H. O. Pinheiro y Ronan Collobert. «Recurrent Convolutional Neural Networks for Scene Parsing». En: *arXiv preprint arXiv:1306.2795* (2013). URL: <https://arxiv.org/abs/1306.2795>.
- [65] Jonathan Long, Evan Shelhamer y Trevor Darrell. «Fully Convolutional Networks for Semantic Segmentation». En: *arXiv preprint arXiv:1411.4038* (2015). URL: <https://arxiv.org/abs/1411.4038>.
- [66] Autocodificadores. (s/f). Mathworks.com. 2024. URL: <https://la.mathworks.com/discovery/autoencoder.html>.
- [67] Thomas Schlegl et al. «f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks». En: *Medical Image Analysis* 54 (2019), págs. 30-44. DOI: [10.1016/j.media.2019.01.010](https://doi.org/10.1016/j.media.2019.01.010).
- [68] Yu Sun et al. «A Multi-Attention UNet for Semantic Segmentation in Remote Sensing Images». En: *Symmetry* 14.5 (2022). DOI: [10.3390/sym14050906](https://doi.org/10.3390/sym14050906).

- [69] Olaf Ronneberger, Philipp Fischer y Thomas Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». En: *arXiv preprint arXiv:1505.04597* (2015).
- [70] Jason Kugelman et al. «A comparison of deep learning U-Net architectures for posterior segment OCT retinal layer segmentation». En: *Scientific Reports* 12 (2022). doi: [10.1038/s41598-022-18646-2](https://doi.org/10.1038/s41598-022-18646-2).
- [71] Bo Pang et al. «GA-UNet: A Lightweight Ghost and Attention U-Net for Medical Image Segmentation». En: *Journal of Imaging Informatics in Medicine* (2024). doi: [10.1007/s10278-024-01070-5](https://doi.org/10.1007/s10278-024-01070-5).
- [72] Rasha Alshawi, Md Tamjidul Hoque y Maik C. Flanagin. «A Depth-Wise Separable U-Net Architecture with Multiscale Filters to Detect Sinkholes». En: *Remote Sensing* 15.5 (2023). doi: [10.3390/rs15051384](https://doi.org/10.3390/rs15051384).
- [73] Debesh Jha et al. «DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation». En: *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*. 2020, págs. 558-564. doi: [10.1109/CBMS49503.2020.00111](https://doi.org/10.1109/CBMS49503.2020.00111).
- [74] Debesh Jha et al. «ResUNet++: An Advanced Architecture for Medical Image Segmentation». En: *2019 IEEE International Symposium on Multimedia (ISM)*. 2019, págs. 225-2255. doi: [10.1109/ISM46123.2019.00049](https://doi.org/10.1109/ISM46123.2019.00049).
- [75] Huggingface.co. URL: <https://huggingface.co/>.
- [76] R. Gupta. *Transfer learning*. BotPenguin. 2023. URL: <https://botpenguin.com/glossary/transfer-learning>.
- [77] S. Weisberg. *Applied Linear Regression*. Wiley, 2005.
- [78] Trevor Hastie, Robert Tibshirani y Jerome Friedman. «Unsupervised Learning». En: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer New York, 2009, págs. 485-585. doi: [10.1007/978-0-387-84858-7_14](https://doi.org/10.1007/978-0-387-84858-7_14).
- [79] Yuxi Li. «Deep Reinforcement Learning: An Overview». En: *arXiv preprint arXiv:1701.07274* (2018).
- [80] Key differences of machine learning vs. Deep learning. Pipedrive. 2024. URL: <https://www.pipedrive.com/en/blog/machine-learning-vs-deep-learning>.
- [81] Yunjie Tian et al. «Semantic-Aware Generation for Self-Supervised Visual Representation Learning». En: *arXiv preprint arXiv:2111.13163* (2021).

- [82] Synced. *RI seminar: Yann LeCun: The next frontier in AI: Unsupervised learning*. Youtube. 2019. URL: <https://medium.com/syncedreview/yann-lecun-cake-analogy-2-0-a361da560dae>.
- [83] W. L Taylor. «Çloze procedure”: a new tool for measuring readability.» En: *Journalism Quarterly* 30 (2022), págs. 415-433. doi: [10.1038/s41598-022-18646-2](https://doi.org/10.1038/s41598-022-18646-2).
- [84] Yann LeCun. *RI seminar: Yann LeCun: The next frontier in AI: Unsupervised learning*. Youtube. 2016. URL: <https://www.youtube.com/watch?v=IbjF5VjniVE>.
- [85] Gemini Team Google. «Gemini: A Family of Highly Capable Multimodal Models». En: *arXiv preprint arXiv:2312.11805* (2023).
- [86] Meta AI. «LLaMA: Open and Efficient Foundation Language Models». En: *arXiv preprint arXiv:2302.13971* (2023).
- [87] B. Liu. *NLP pretraining - from BERT to XLNet*. 2019. URL: <https://bangliu.github.io/survey/2019/07/01/NLP-Pretraining/>.
- [88] Junlin Han et al. «You Only Cut Once: Boosting Data Augmentation with a Single Cut». En: *arXiv preprint arXiv:2201.12078* (2022).
- [89] Humza Naveed et al. «Survey: Image Mixing and Deleting for Data Augmentation». En: *arXiv preprint arXiv:2106.07085* (2023).
- [90] Aekam Parmar. *The what, why, and how of data augmentation in Machine Learning*. VOLANSYS. 2021. URL: <https://www.volansys.com/blog/data-augmentation-in-ml/>.
- [91] Spyros Gidaris, Praveer Singh y Nikos Komodakis. «Unsupervised Representation Learning by Predicting Image Rotations». En: *arXiv preprint arXiv:1803.07728* (2018).
- [92] Carl Doersch, Abhinav Gupta y Alexei A. Efros. «Unsupervised Visual Representation Learning by Context Prediction». En: *arXiv preprint arXiv:1505.05192* (2016).
- [93] Mehdi Noroozi y Paolo Favaro. «Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles». En: *arXiv preprint arXiv:1603.09246* (2017).
- [94] Kaiming He et al. «Momentum Contrast for Unsupervised Visual Representation Learning». En: *arXiv preprint arXiv:1911.05722* (2020).
- [95] Ting Chen et al. «A Simple Framework for Contrastive Learning of Visual Representations». En: *arXiv preprint arXiv:2002.05709* (2020).

- [96] Xinlei Chen y Kaiming He. «Exploring Simple Siamese Representation Learning». En: *arXiv preprint arXiv:2011.10566* (2020).
- [97] Kaiming He et al. «Masked Autoencoders Are Scalable Vision Learners». En: *arXiv preprint arXiv:2111.06377* (2021).
- [98] Yalong Bai et al. «Directional Self-supervised Learning for Heavy Image Augmentations». En: *arXiv preprint arXiv:2110.13555* (2021).
- [99] Ekin D. Cubuk et al. «RandAugment: Practical automated data augmentation with a reduced search space». En: *arXiv preprint arXiv:1909.13719* (2019).
- [100] Pengguang Chen, Shu Liu y Jiaya Jia. «Jigsaw Clustering for Unsupervised Visual Representation Learning». En: *arXiv preprint arXiv:2104.00323* (2021).
- [101] Benyamin Hosseiny et al. «Beyond Supervised Learning in Remote Sensing: A Systematic Review of Deep Learning Approaches». En: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* PP (2023), págs. 1-22. doi: [10.1109/JSTARS.2023.3316733](https://doi.org/10.1109/JSTARS.2023.3316733).
- [102] Huimin Huang et al. «UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation». En: *arXiv preprint arXiv:2004.08790* (2020).
- [103] Enze Xie et al. «SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers». En: *arXiv preprint arXiv:2105.15203* (2021).
- [104] Hu Cao et al. «Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation». En: *arXiv preprint arXiv:2105.05537* (2021).
- [105] Yiwen Ye et al. «DeSD: Self-Supervised Learning with Deep Self-Distillation for 3D Medical Image Segmentation». En: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022*. Springer Nature Switzerland, 2022, págs. 545-555.
- [106] Lei Zhou et al. «Self Pre-training with Masked Autoencoders for Medical Image Classification and Segmentation». En: *arXiv preprint arXiv:2203.05573* (2023).
- [107] Yuan Wang et al. «Fully Self-Supervised Learning for Semantic Segmentation». En: *arXiv preprint arXiv:2202.11981* (2022).
- [108] Y. Li. *Colab Pro vs. Free — AI computing performance. Towards Data Science*. 2021. URL: <https://towardsdatascience.com/colab-pro-vs-free-ai-computing-performance-4e983d578fb2>.

- [109] R. Rahmat. *Tensorflow vs PyTorch vs keras*. GoPenAI. 2022. URL: <https://blog.gopenai.com/tensorflow-vs-pytorch-vs-keras-9161988c19b9>.
- [110] Pytorch.org. 2024. URL: <https://pytorch.org/>.
- [111] TensorFlow. 2024. URL: https://www.tensorflow.org/api_docs/python/tf.
- [112] Keras 3 API documentation. Keras.io. 2024. URL: <https://keras.io/api/>.
- [113] W-Y Hong et al. «Cholecseg8k: a semantic segmentation dataset for laparoscopic cholecystectomy based on cholec80». En: *arXiv preprint arXiv:2012.12453* (2020).
- [114] S. Saxena. *cholec80 [Data set]*. Kaggle. 2023. URL: <https://www.kaggle.com/datasets/swamysaxena09/cholec80>.
- [115] Shervin Minaee et al. «Image Segmentation Using Deep Learning: A Survey». En: *arXiv preprint arXiv:2001.05566* (2020).
- [116] Narinder Singh Punn y Sonali Agarwal. «Modality specific U-Net variants for biomedical image segmentation: a survey». En: *Artificial Intelligence Review* 55.7 (2022), págs. 5845-5889. DOI: [10.1007/s10462-022-10152-1](https://doi.org/10.1007/s10462-022-10152-1).
- [117] F. Isensee et al. «nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation.» En: *Nature methods* 18.2 (2021), págs. 203-211.
- [118] Zhendong Wang et al. «Uformer: A General U-Shaped Transformer for Image Restoration». En: *arXiv preprint arXiv:2106.03106* (2021).
- [119] Xiangyi Yan et al. «Representation Recovering for Self-Supervised Pre-training on Medical Images». En: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, págs. 2684-2694.
- [120] Hanna Borgli et al. «HyperKvasir, a comprehensive multi-class image and video dataset for gastrointestinal endoscopy». En: *Scientific Data* 7.1 (2020). DOI: [10.1038/s41597-020-00622-y](https://doi.org/10.1038/s41597-020-00622-y).
- [121] P. Gupta. *nnU-Net: The no-new-UNet for automatic segmentation*. MICCAI Educational Initiative. 2020. URL: <https://medium.com/miccai-educational-initiative/nnu-net-the-no-new-unet-for-automatic-segmentation-8d655f3f6d2a>.
- [122] Hong-Yu Zhou et al. «nnFormer: Interleaved Transformer for Volumetric Segmentation». En: *arXiv preprint arXiv:2109.03201* (2022).

- [123] Sheng He et al. «U-Netmer: U-Net meets Transformer for medical image segmentation». En: *arXiv preprint arXiv:2304.01401* (2023).
- [124] Ali Hatamizadeh et al. «UNETR: Transformers for 3D Medical Image Segmentation». En: *arXiv preprint arXiv:2103.10504* (2021).
- [125] A. Pesce, N. Fabbri y C. V. Feo. «Vascular injury during laparoscopic cholecystectomy: An often-overlooked complication.» En: *World Journal of Gastrointestinal Surgery* 15.3 (2023), págs. 338-345. DOI: [10.4240/wjgs.v15.i3.338](https://doi.org/10.4240/wjgs.v15.i3.338).
- [126] Xinlong Wang et al. «Dense Contrastive Learning for Self-Supervised Visual Pre-Training». En: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, págs. 3023-3032. DOI: [10.1109/CVPR46437.2021.00304](https://doi.org/10.1109/CVPR46437.2021.00304).
- [127] Krishna Chaitanya et al. «Contrastive learning of global and local features for medical image segmentation with limited annotations». En: *arXiv preprint arXiv:2006.10511* (2020).
- [128] Chen Wei et al. «Masked Feature Prediction for Self-Supervised Visual Pre-Training». En: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, págs. 14648-14658. DOI: [10.1109/CVPR52688.2022.01426](https://doi.org/10.1109/CVPR52688.2022.01426).
- [129] Seyed Rohollah Hosseyni, Sanaz Seyedin y Hasan Taheri. «Human Action Recognition in Still Images Using ConViT». En: *arXiv preprint arXiv:2307.08994* (2024).
- [130] Lucas Beyer, Xiaohua Zhai y Alexander Kolesnikov. «Better plain ViT baselines for ImageNet-1k». En: *arXiv preprint arXiv:2205.01580* (2022).
- [131] Rafael Padilla, Sergio Netto y Eduardo da Silva. «A Survey on Performance Metrics for Object-Detection Algorithms». En: 2020. DOI: [10.1109/IWSSIP48289.2020](https://doi.org/10.1109/IWSSIP48289.2020).
- [132] Forética. *Inteligencia artificial: hacia un impacto económico, social y medioambiental positivo* - Forética. Forética. 2018. URL: <https://foretica.org/2018/11/inteligencia-artificial-hacia-un-impacto-economico-social-y-medioambiental-positivo/>.

- [133] La OMS publica el primer informe mundial sobre inteligencia artificial (IA) aplicada a la salud y seis principios rectores relativos a su concepción y utilización. Who.int. 2021. URL: <https://www.who.int/es/news/item/28-06-2021-who-issues-first-global-report-on-ai-in-health-and-six-guiding-principles-for-its-design-and-use>.
- [134] Unesco.org. 2024. URL: https://unesdoc.unesco.org/ark:/48223/pf000386510_spa.
- [135] OdiseIA. 2024. URL: <https://www.odiseia.org/>.
- [136] Juan Gustavo Corvalán. «nteligencia artificial: retos, desafíos y oportunidades - Prometea: la primera inteligencia artificial de Latinoamérica al servicio de la Justicia». En: *Revista de Investigações Constitucionais* 5.1 (2018), págs. 295-316. DOI: [10.5380/rinc.v5i1.55334](https://doi.org/10.5380/rinc.v5i1.55334).
- [137] Agenda 2030. Gob.es. 2024. URL: <https://www.mdsocialesa2030.gob.es/agenda2030/index.htm>.
- [138] El impacto económico de la inteligencia artificial en la atención médica: Ventajas y desafíos económicos. Realidad Económica. 2024. URL: <https://www.realidadeconomica.es/el-impacto-economico-de-la-inteligencia-artificial-en-la-atencion-medica-27/34932>.



Universidad
Politécnica
de Madrid

ETSI SISTEMAS
INFORMÁTICOS