TESINA FINAL DE GRADO

Licenciatura en Ciencias de la Computación

Análisis e Implementación de Herramientas Inteligentes para la Detección de Cáncer en Imágenes Médicas

Autora: Mariel Denise Volman Stern

Tutor: Dr. Ing. Pablo Javier Vidal

Octubre 2023

Facultad de Ingeniería
UNIVERSIDAD NACIONAL DE CUYO

Agradecimientos

Quiero expresar mi profundo agradecimiento a todas las personas que contribuyeron de manera significativa a la realización de esta tesina. Este trabajo no habría sido posible sin su apoyo y colaboración.

En primer lugar, deseo agradecer a mi director de tesis, Dr. Pablo Javier Vidal, por su orientación constante, paciencia y experiencia. Sus conocimientos y consejos fueron fundamentales para dar forma a este trabajo. También quiero agradecer a la Dra. Ana Carolina Olivera que estuvo siempre disponible para resolver cualquier inquietud que surgiera en el proceso de elaboración del presente trabajo.

También agradezco sinceramente a todos los profesores de la carrera, que me han brindado los conocimientos necesarios durante estos años para poder llegar hasta acá.

A mis amigos y familiares, especialmente a mis padres, les agradezco por su inquebrantable apoyo emocional y comprensión durante todo este proceso. Su aliento fue un pilar fundamental para mantenerme motivada.

Por último, pero no menos importante, quiero dedicar este trabajo a mi hermano, cuyas contribuciones han sido una fuente constante de inspiración en mi vida académica y profesional.

¡Muchas gracias!

Mariel Denise Volman Stern
Mendoza, Octubre 2023

Resumen

La inteligencia artificial, en particular, las redes neuronales, son herramientas poderosas que han permitido mejorar la detección y el diagnóstico de enfermedades analizando y detectando patrones. Por ejemplo, en imágenes médicas, posibilitan la identificación temprana de enfermedades y el tratamiento rápido de las mismas.

El cáncer de colon es un tipo de cáncer que comienza en el intestino grueso (colon). Este suele afectar a los adultos mayores, aunque puede aparecer a cualquier edad. Por lo general, comienza como grupos pequeños y no cancerosos (benignos) de células llamados pólipos que se forman en el interior del colon. La colonoscopia se usa para buscar la presencia de cambios (pólipos) en el intestino grueso. El video resultante de este procedimiento puede generar un centenar de imágenes que pueden requerir varias horas de análisis por parte del experto, lo que lo vuelve una tarea monótona y plausible de errores humanos.

En este trabajo, se presenta un sistema de detección de cáncer de colon utilizando un algoritmo basado en la arquitectura de una Red Neuronal Profunda (DNN, *Deep Neural Network*). En particular, se utiliza la versión 8 de la red neuronal *You Only Look Once* (YOLO). Se evalúa el rendimiento de este algoritmo a partir de diferentes conjuntos de bases de datos de imágenes de colonoscopias para entrenar y probar el rendimiento del sistema. Los resultados muestran que el modelo seleccionado es capaz de detectar con precisión la presencia de pólipos y es significativamente más rápido que otros modelos de la literatura. Por otro lado, el sistema de detección de cáncer de colon permite visualizar la identificación producida por la red neuronal, resultando una herramienta valiosa como sistema auxiliar para ayudar a los médicos a tomar decisiones sobre el diagnóstico y tratamiento de la enfermedad.

Abstract

Artificial intelligence, in particular neural networks, are powerful tools that have allowed us to improve the detection and diagnosis of diseases by analysing and detecting patterns. For example, in medical images, they enable the early identification of diseases and their rapid treatment.

Colon cancer is a type of cancer that begins in the large intestine (colon). This usually affects older adults, although it can appear at any age. It usually starts as small, non-cancerous (benign) clumps of cells called polyps that form inside the colon. A colonoscopy is used to look for the presence of changes (polyps) in the large intestine. The video resulting from this procedure can generate a hundred images that are difficult to analyse appropriately by the expert, in addition to having to analyse hundreds of images from different videos, which sometimes makes the task too heavy for the technician in charge.

In this work, a colon cancer detection system is presented using a *Deep Neural Network*-based algorithm. Specifically, version 8 of the *You Only Look Once* (YOLO) neural network is employed. The performance of this algorithm is evaluated using several datasets of colonoscopy images for training and testing the system. The results demonstrate that the selected model can accurately detect the presence of polyps and is significantly faster than other models in the literature. Additionally, the colon cancer detection system allows visualizing the identification produced by the neural network, making it a valuable auxiliary tool to assist medical professionals in making decisions regarding the diagnosis and treatment of the disease.

Tabla de Contenidos

Índice de Tablas

Tabla **Página**

Índice de Figuras

Figura **Página**

Capítulo 1

Introducción

1.1. Motivación

En la actualidad, el cáncer es una de las principales enfermedades que afecta la salud humana y tiene una alta tasa de mortalidad. Según las estimaciones realizadas por el Observatorio Global del Cáncer (GCO, *Global Cancer Observatory*) de la Agencia Internacional de Investigación del Cáncer (IARC, *International Agency for Research on Cancer*) [15, 14] en Argentina, ocurrieron 130.878 casos nuevos de cáncer en ambos sexos en el año 2020. Respecto de los países de América Latina, nuestro país se ubicó en quinto lugar en términos de frecuencia, subiendo dos posiciones en relación al año 2018. La distribución de los casos según los principales sitios tumorales muestra que el cáncer de mama, con 22.024 casos, fue el de mayor magnitud en el año 2020, representando el 16,8 % de todos los casos nuevos y es el más frecuente en mujeres. En segundo lugar, se ubicó el cáncer colorrectal, con 15.895 casos nuevos, representando el 12,1 % del total y, en tercer lugar, el cáncer de pulmón, con 12.110 casos nuevos, que concentran el 9,3 % del total de tumores.

En particular, el Cáncer Colorrectal (CRC, *Colorectal Cancer*) es el tercero con mayor incidencia a nivel mundial y el segundo con mayor tasa de mortalidad [32]. Este cáncer se desarrolla en el colon y el recto (partes del intestino grueso), teniendo la particularidad de que en más del 80 % de los casos, se genera primero un pólipo (crecimiento anormal de las células) denominado adenoma, que puede crecer lentamente durante más de 10 años y transformarse en CRC si no se detecta y extirpa a tiempo. La progresión lenta de los pólipos o adenomas hace posible la realización de exámenes

periódicos para detectarlos y extirparlos, permitiendo prevenir la enfermedad. Además, este tipo de cáncer tiene alta tasa de supervivencia, considerando que si se detecta tempranamente, las posibilidades de curación son superiores al 90 % y los tratamientos a etapas tempranas son menos invasivos [29].

Los exámenes de detección verifican la presencia del cáncer antes de que la persona tenga síntomas. Si en un examen preventivo se descubre el tejido anormal, las probabilidades de un tratamiento favorable aumentan, si no, recién cuando aparecen los primeros síntomas, es probable que el cáncer se haya empezado a diseminar [58]. Actualmente, la colonoscopia es el examen más utilizado para detectar el cáncer de colon [46]. Este procedimiento se utiliza para observar el interior del recto y el colon para determinar si hay pólipos, áreas anormales o cáncer. Se introduce un colonoscopio a través del ano hasta llegar al colon. Un colonoscopio es un instrumento delgado en forma de tubo con una luz y una lente para observar. En la extremidad del colonoscopio hay una microcámara que transmite las imágenes a un monitor, permitiendo que el médico vea y grabe lo que ocurre dentro del intestino grueso. Los últimos centímetros del colonoscopio son articulados y pueden girar en varios ángulos, a fin de facilitar la visualización de todo el interior del colon.

La interpretación de las imágenes obtenidas de una colonoscopia a veces puede ser parcial, produciendo una variabilidad en la precisión de la detección de pólipos [48]. Algunos estudios sugieren que más de la mitad de los casos de cáncer de colon detectados por una colonoscopia surgen de lesiones que no se vieron en las colonoscopias anteriores a las que se sometió el paciente [10]. Uno de los últimos hitos tecnológicos como parte de la reducción del margen de error en estos procesos es la aplicación de la Inteligencia Artificial (AI, *Artificial Intelligence*) al campo de la colonoscopia.

Diferentes técnicas de AI han comenzado a ser ampliamente utilizadas como parte de los procedimientos de detección de enfermedades [25]. A través de un proceso de aprendizaje previo, la AI es capaz de dar respuestas similares a las que daría un humano en la misma situación pero con tiempos de respuesta mucho más rápidos. A las técnicas que engloban este aprendizaje se les denomina Aprendizaje Automático (ML, *Machine Learning*). Los modelos de ML utilizan un conjunto de datos de entrada para entrenar el sistema y reconocer patrones, de manera tal que permita al sistema "aprender" y aplicar su conocimiento a nuevos datos.

El Aprendizaje Profundo (DL, *Deep Learning*) es un subcampo del ML. En el ML hay que guiar al sistema en cada una de las fases del proceso para que, a través de la práctica, aprenda a identificar lo que se desea, de manera automática. Contrariamente, en el DL el sistema aprende por sí solo con cada nueva entrada de información que recibe. Los recientes avances en ML [58] permiten reconocer y clasificar patrones complejos a

partir de diferentes modalidades de imágenes médicas. En muchas aplicaciones, estos sistemas han demostrado un rendimiento comparable al de la toma de decisiones humana y son los ingredientes claves de los futuros sistemas de control y toma de decisiones clínicas.

Con este trabajo se busca dar soporte a los especialistas que trabajan con imágenes médicas de colonoscopias utilizando técnicas de AI, en particular, DL, para ayudar a mejorar la detección y el diagnóstico de enfermedades de colon proporcionando una herramienta objetiva y precisa para la interpretación de imágenes médicas.

1.2. Objetivos

El objetivo principal de esta Tesina Final de Grado consiste en aplicar herramientas de Aprendizaje Profundo (DL, *Deep Learning*) para el análisis de conjuntos de imágenes médicas en la detección temprana del cáncer.

En base a lo anterior, como objetivos específicos se pueden mencionar:

- **Objetivo 1:** Comprender el funcionamiento de las técnicas comúnmente aplicadas a la detección de cáncer utilizando imágenes médicas, en particular, aquellas que corresponden al DL.
- **Objetivo 2:** Diseñar y trabajar sobre un *pipeline* que permita evaluar el comportamiento de técnicas de AI para la detección del cáncer de colon.
- **Objetivo 3:** Implementar un prototipo que permita evaluar diferentes imágenes y visualizar la detección del algoritmo seleccionado, ayudando a los médicos a detectar rápidamente los pólipos que pueden generar el cáncer de colon.

1.3. Estructura general del documento

El presente documento se encuentra organizado de la siguiente manera:

- **Capítulo 2:** Brinda un marco teórico sobre conceptos de Inteligencia Artificial y Redes Neuronales Artificiales, centrándose en las Redes Neuronales Convolucionales utilizadas para los problemas de detección, aplicadas en imágenes médicas de colonoscopias. A su vez, este capítulo describe los trabajos relacionados en esta área.
- **Capítulo 3:** Detalla el diseño y desarrollo del trabajo, explicando la metodología llevada a cabo para la implementación del sistema de detección de cáncer de colon.

Además, presenta el algoritmo seleccionado y define las diferentes instancias de prueba, mencionando las tecnologías y herramientas utilizadas.

- **Capítulo 4:** Expone los principales resultados obtenidos a partir de los diferentes experimentos realizados y el diseño e implementación de la herramienta de visualización.
- **Capítulo 5:** Presenta las conclusiones del trabajo, dificultades encontradas durante el transcurso del mismo y trabajos futuros.

Capítulo 2

Marco Teórico y Antecedentes

Este capítulo tiene el propósito de introducir las nociones básicas de la AI y cómo se aplica al procesamiento de imágenes digitales. El objetivo es describir el funcionamiento de las Redes Neuronales Artificiales (ANNs, *Artificial Neural Network*s) que se utilizan para la detección de objetos en imágenes haciendo hincapié en las imágenes médicas. En la Sección 2.1 se definen los conceptos de Inteligencia Artificial (AI, *Artificial Intelligence*), Aprendizaje Automático (ML, *Machine Learning*) y Aprendizaje Profundo (DL, *Deep Learning*). Seguidamente, en la Sección 2.2, se brinda una descripción de las ANNs dando detalles de sus modos de operación en la Sección 2.3 y de las métricas que se utilizan para evaluar los modelos en la Sección 2.4. Luego, se profundiza en las Redes Neuronales Convolucionales (CNNs, *Convolutional Neural Network*s) en la Sección 2.5, especialmente en las arquitecturas más utilizadas para la detección de objetos (Sección 2.5.1). Posteriormente, en la Sección 2.6, se explica en qué consiste el análisis de imágenes digitales mediante Redes Neuronales Profundas (DNNs, *Deep Neural Network*s); se introduce el concepto de imágenes digitales (Sección 2.6.1), su aplicación en la detección (Sección 2.6.2) y los trabajos realizados sobre detección de cáncer de colon en imágenes médicas (Sección 2.6.3). Para concluir el capítulo, se da una breve explicación de cómo es el desarrollo de prototipos mediante una metodología ágil en la Sección 2.7.

2.1. Inteligencia Artificial

La Inteligencia Artificial (AI, *Artificial Intelligence*) es un campo de la informática que se enfoca en el desarrollo de sistemas y algoritmos que puedan realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, la toma de decisiones, la resolución de problemas y la comprensión del lenguaje natural [42]. La AI se puede aplicar en diferentes campos, como la robótica, la visión por computadora, la atención médica, el transporte, la educación, la agricultura, entre otros [6, 22, 2].

En la Figura 2.1 se muestra de manera general la agrupación de las diversas áreas que forman la AI. Como se mencionó anteriormente, la AI es un conjunto de técnicas que permiten a las computadoras imitar la inteligencia humana. El Aprendizaje Automático (ML, *Machine Learning*) es un subcampo de la AI que se centra en el diseño y desarrollo de algoritmos y sistemas que pueden aprender y mejorar a partir de la experiencia y los datos. Estos algoritmos utilizan modelos matemáticos y estadísticos para analizar grandes conjuntos de datos, identificar patrones y relaciones, y extraer información para tomar decisiones y hacer predicciones precisas [30]. Tiene dos fases: una primera llamada "entrenamiento", en la cual trabaja con una serie de datos de entrada previamente etiquetados para que el modelo aprenda; y una segunda llamada "predicción", en la cual el modelo ya entrenado realiza la predicción con nuevos datos no etiquetados.

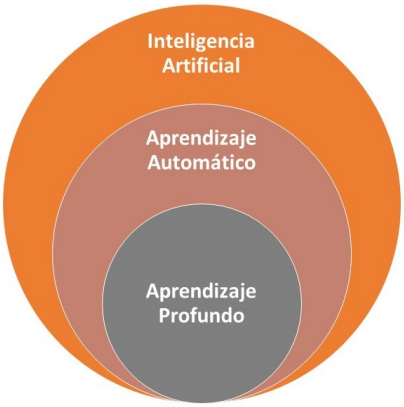

Figura 2.1: Relación entre AI, ML y DL. Inspirada en [2].

Hay varios tipos de ML, como el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo. En el aprendizaje supervisado, el sistema aprende a partir de ejemplos etiquetados, es decir, datos que ya están clasificados o etiquetados por humanos. En el aprendizaje no supervisado, el sistema aprende a partir de datos sin etiquetar, buscando patrones y relaciones por sí mismo. En el aprendizaje por refuerzo,

el sistema aprende a partir de la retroalimentación y el refuerzo o recompensa que recibe de su entorno.

Por otro lado, el Aprendizaje Profundo (DL, *Deep Learning*) es un subconjunto del ML que se enfoca en el diseño y entrenamiento de redes neuronales artificiales muy profundas y complejas que pueden aprender y mejorar a partir de grandes conjuntos de datos. Estas redes neuronales están compuestas por múltiples capas de neuronas interconectadas, donde cada capa procesa y extrae características de la información de entrada. El DL ha demostrado ser muy efectivo en una amplia variedad de aplicaciones, especialmente en aquellas que involucran grandes cantidades de datos no estructurados, como imágenes y texto [12]. Al entrenar redes neuronales profundas en grandes conjuntos de datos, el DL puede aprender características y patrones complejos y sutiles que serían difíciles de detectar mediante otros métodos. El *Deep Learning* se ha utilizado en muchas aplicaciones, entre las que se encuentran la clasificación de imágenes y el diagnóstico médico [28]. La capacidad del aprendizaje profundo para aprender automáticamente de grandes conjuntos de datos ha llevado a avances significativos en muchas áreas y ha cambiado la forma en que se interactúa con la tecnología [52].

2.2. Redes Neuronales Artificiales

Una Red Neuronal Artificial (ANN, *Artificial Neural Network*) es un sistema de procesamiento de información inspirada en el cerebro humano. Está diseñada para aprender y realizar tareas complejas a partir de datos de entrada. Al igual que el cerebro humano, las ANNs están compuestas por un gran número de unidades de procesamiento, llamadas neuronas artificiales, que están interconectadas mediante conexiones ponderadas, llamadas sinapsis [1, 31].

Cada neurona artificial es una unidad o nodo que recibe datos del exterior o de otras neuronas, simulando los impulsos nerviosos que reciben las neuronas del cerebro humano. Cada neurona procesa y genera un valor de salida que alimenta a otras neuronas de la red o son la salida hacia el exterior de la red [42].

Una neurona artificial está conformada por:

- Un *conjunto de datos de entrada* que son los enlaces o interconexiones por donde reciben información del exterior de la red o de otras neuronas. Cada una de estas entradas está *ponderada* por un *peso* que determina la importancia del dato que recibe por esa interconexión y que es un valor que se va ajustando durante el

entrenamiento de la red neuronal para minimizar el error de la red y que los resultados sean más fiables.

- Un conjunto de *funciones*:
  - Función de propagación: permite relacionar matemáticamente los valores de las entradas, sus pesos y un *sesgo (*bias*)* para el cálculo del valor de salida de la neurona.
  - Función de activación: determina si la neurona está activa o no, es decir, si produce un valor de salida o no. Se utiliza para introducir la *no linealidad* en la salida de la neurona, ya que sin funciones de activación, una red neuronal sería simplemente una combinación lineal de sus entradas, lo que limitaría su capacidad para aprender y representar relaciones no lineales en los datos. Existen varias funciones de activación y su elección depende de la naturaleza de la tarea.

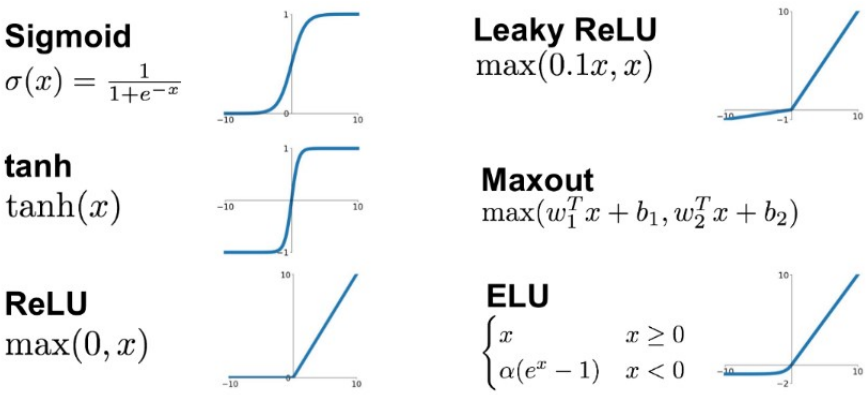

Figura 2.2: Ejemplos de funciones de activación. Extraída de [21].

En la Figura 2.2 se muestran algunas de las más comunes. Una de las funciones más comunes es la Sigmoide, la cual lleva todos los valores entre (0, 1), por lo que se usa especialmente para modelos en los que se tiene que predecir la probabilidad de algo, ya que se utiliza 0 como 0 % y 1 como 100 %. Otra función comúnmente usada como se ve en el gráfico es la Función ReLU (ReLU, *Rectified Lineal Unit*), la cual se mantiene constante al principio: $f(x)$ es cero cuando x es menor que cero y luego se dispara hacia arriba en forma lineal donde $f(x)$ es igual a x cuando x es superior o igual a cero. La clave está en que todos los valores negativos se vuelven cero, eso significa que cualquier entrada negativa dada a la función de activación de ReLU convierte el valor en cero inmediatamente. Esto puede ayudar mucho en la simplificación computacional, ya que todos los valores iguales

a 0 son inmediatamente descartados (dichas neuronas son irrelevantes). A su vez, esto puede disminuir la capacidad del modelo para ajustarse o entrenarse a partir de los datos correctamente. En general, la función ReLU tiene un buen rendimiento en las CNN.

- *Salida de la neurona*, que es el enlace o interconexión por donde entrega el resultado al exterior de la red neuronal u otras neuronas.

En la Figura 2.3 se puede observar la arquitectura de una ANN, donde los valores x_i representan los datos de entrada y los valores w_i los pesos de cada interconexión. La función de propagación consiste en realizar la sumatoria de los valores de entrada multiplicados por los pesos correspondientes que sirven de entrada para la función de activación. Luego, se puede utilizar diferentes funciones de activación tales como las mencionadas anteriormente para modificar ese valor y producir la salida de la red.

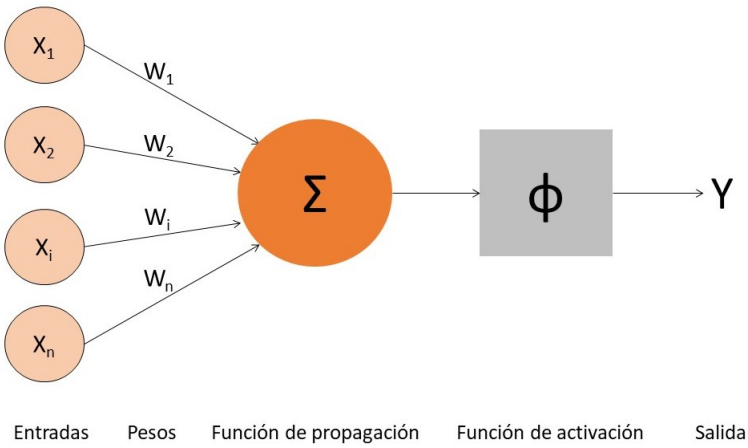

Figura 2.3: Arquitectura de una Red Neuronal Artificial. Inspirada en [42].

Las ANNs pueden ser clasificadas de acuerdo a la arquitectura que adopta cada modelo. Usualmente, se pueden clasificar según su topología [39]:

- Perceptrón Simple (SLP, *Single Layer Perceptron*) [20]: se corresponde con la red neuronal más simple, está formada por una única capa de neuronas que recibe valores del exterior de la red, las procesa y genera unos valores de salida que entrega como resultado. Las neuronas de la capa no se interconectan entre ellas.

- Perceptrón Multicapa (MLP, *Multi Layer Perceptron*) [13]: es el tipo de red neuronal más extendido. Consta de una capa de entrada que recibe valores del exterior, una serie de capas intermedias u ocultas que van procesando la información y una capa de salida que entrega los resultados de la red neuronal. Las neuronas

de una capa se interconectan con todas las neuronas de la capa siguiente, pero no entre ellas.

- Red Neuronal con Función de Base Radial (RBFNN, *Radial Basis Function Neural Network*) [50]: estas redes están compuestas únicamente de tres capas, la de entrada, una capa oculta y la de salida. Presentan un comportamiento similar al perceptrón simple. Su nombre surge debido a que la función de activación de las neuronas de la capa oculta es de base radial, generalmente de tipo Gaussiano.
- Red Neuronal Convolucional (CNN, *Convolutional Neural Network*) [24]: son similares a las redes neuronales multicapa, pero la diferencia radica en que las neuronas de cada capa no se interconectan con todas las de la capa siguiente, sino con un subconjunto de estas, y tampoco se interconectan entre las neuronas de la misma capa. En este caso se habla de especialización de las neuronas (grupos) y se reduce la cantidad de unidades necesarias y la complejidad de los sistemas que utilizan este tipo de redes.
- Redes Neuronales Recurrentes (RNN, *Recurrent Neural Networks*) [54]: en estas redes las neuronas no se organizan en capas, sino que están interconectadas entre ellas de manera arbitraria, incluso pudiendo crear ciclos. Esto permite que estas redes tengan memoria, es decir, que la información generada en iteraciones anteriores afecte al resultado del procesamiento en un tiempo futuro.

En este trabajo se profundiza en las redes convolucionales, las cuales se han especializado en el análisis de imágenes [44].

2.3. Modos de operación

El proceso de desarrollo de las redes neuronales consiste en tres fases: fase de entrenamiento, fase de validación y fase de prueba. A continuación se explica cada una de ellas.

2.3.1. Fase de entrenamiento

Durante la fase de entrenamiento, la red neuronal aprende a partir de un conjunto de datos de entrenamiento para luego poder realizar predicciones precisas en función de las características proporcionadas. Para ello, en el aprendizaje supervisado, se alimenta a la red con ejemplos de entrada junto con las etiquetas o salidas deseadas correspondientes.

El proceso de aprendizaje comienza desde cero (*from scratch*) partiendo de un conjunto de pesos sinápticos aleatorios. El objetivo es buscar un conjunto de pesos que permitan a la red desarrollar correctamente una determinada tarea. Durante el proceso se va refinando iterativamente la solución hasta alcanzar un nivel de operación suficientemente aceptable. Sin embargo, existe una técnica denominada Aprendizaje por Transferencia (TL, *Transfer Learning*) que permite transferir el conocimiento de un modelo ya entrenado a otro con el fin de proporcionar un entrenamiento más rápido y exitoso.

Es importante durante el entrenamiento monitorear el riesgo de sobreajuste (*overfitting*), donde el modelo se adapta demasiado a los datos de entrenamiento y no generaliza bien a datos no vistos. Esto se controla mediante el uso de datos de validación.

2.3.2. Fase de validación

La fase de validación se utiliza para evaluar el rendimiento del modelo durante el entrenamiento y ajustar los hiperparámetros para obtener un modelo más generalizado y evitar el sobreajuste.

Los hiperparámetros son variables de configuración externa que se seleccionan de manera manual antes de entrenar un modelo. Algunas configuraciones comunes incluyen: el número de épocas (*epochs*), el tamaño del lote (*batch size*), la elección del optimizador (*optimizer*) y la tasa de aprendizaje (*learning rate*).

Los *epochs* son el número de veces que el conjunto de datos de entrenamiento se muestra por completo a la red neuronal durante el entrenamiento. Cada *epoch* se divide en lotes, el *batch size* indica la cantidad de datos que se entrenarán en una iteración dentro de un *epoch*. Se realizarán tantas iteraciones hasta entrenar todo el conjunto. El *optimizer* busca optimizar los valores de los parámetros internos para reducir el error cometido por la red. Para ello, se basa en el *learning rate*, que indica la frecuencia con la que un algoritmo actualiza las estimaciones. Un *learning rate* muy chico puede hacer que la red demore demasiado en aprender, mientras que uno muy grande puede generar un comportamiento errático donde la red no aprenderá nada.

El ajuste de hiperparámetros consiste en la realización de múltiples experimentos con un conjunto de hiperparámetros para elegir adecuadamente los que mejor se ajustan al modelo y poder mejorar el rendimiento de la red neuronal.

2.3.3. Fase de prueba

Esta fase se utiliza para evaluar el rendimiento final del modelo después de que se haya completado el entrenamiento y se hayan ajustado los hiperparámetros.

Se utiliza un conjunto de datos completamente separado, llamado conjunto de datos de prueba, que no se ha utilizado en ninguna etapa anterior (entrenamiento ni validación). El modelo hace predicciones en este conjunto de datos y se evalúa su rendimiento en datos completamente nuevos.

2.4. Evaluación de los modelos

Para evaluar el rendimiento de un modelo de ML se utilizan métricas. Estas son medidas cuantitativas que proporcionan una forma objetiva de medir qué tan bien funciona un modelo frente a datos nunca observados o bien, se utilizan para comparar el desempeño de diferentes modelos. De acuerdo al tipo de algoritmo a evaluar, se utilizan diferentes métricas.

La métrica más usada para los algoritmos de detección es la Intersección Sobre Unión (IOU, *Intersection Over Union*), que mide la superposición entre dos límites. Muestra cuánto se superpone el límite predicho con el límite del objeto real. Luego de obtener este valor, las predicciones se clasifican en Verdadero Positivo (TP, *True Positive*), Falso Positivo (FP, *False Positive*), Verdadero Negativo (TN, *True Negative*) y Falso Negativo (FN, *False Negative*) en base a un umbral predefinido de IOU. Por lo tanto, el resultado será:

- TP: cuando el valor real es positivo y la predicción también es positiva, es decir, había un pólipo y se detectó como pólipo.
- FP: cuando el valor real es negativo y se predice como positivo, es decir, cuando se detecta un pólipo incorrectamente.
- TN: cuando el valor real es negativo y la predicción también es negativa, es decir, no había ningún pólipo y tampoco se detectó que hubiera. En los problemas de detección este valor no se evalúa ya que existen infinitas cajas delimitadoras en cada imagen que no contienen la clase que se desea detectar, por lo que se considera como fondo y no deben ser detectadas.
- FN: cuando el valor real es positivo y se predice como negativo, es decir, había un pólipo, pero no se detectó.

A partir de estas opciones surgen las siguientes métricas: precisión (*precision*), sensibilidad (*recall*), puntuación F1 (*F1 Score*) y la precisión media (AP, *Average Precision*).

La *precision* indica cuántos de los valores estimados como positivos son realmente positivos. Por lo tanto, es una métrica importante para restar el costo del número de FP. Se utiliza para saber si el modelo realiza una clasificación o predicción acertada capaz de identificar la clase buscada o excluir el resto de clases. Se calcula en la Ecuación 2.1.

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

El *recall* indica la proporción de casos positivos que son correctamente detectados. Por lo tanto, se utiliza cuando hay un alto costo asociado con el FN. En medicina es muy utilizada ya que el costo de no detectar una enfermedad puede ser grave. También se la denomina Tasa de Verdaderos Positivos. Se calcula en la Ecuación 2.2.

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

El *F1 Score* resume la precisión y la sensibilidad en una sola métrica, su fórmula se muestra en la Ecuación 2.3. Es útil cuando se desea buscar un equilibrio entre la precisión y la sensibilidad y hay una distribución de clases desigual.

$$F1\ Score = \frac{2 * precision * recall}{precision + recall} \tag{2.3}$$

Otra métrica que permite relacionar la precisión con la sensibilidad es el AP. Esta métrica se utiliza comúnmente para analizar el rendimiento de los sistemas de detección de objetos. Se calcula en la Ecuación 2.4.

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.4}$$

donde R_n y P_n son la *precision* y el *recall* en el umbral n.

AP también puede considerarse como el área bajo la curva de precisión sobre sensibilidad, que es trazada para diferentes valores de umbral de confianza. Y mAP es la media de AP sobre todas las clases, que en este caso coinciden ya que hay una sola clase. Por otro lado, el número que se añade entre paréntesis indica el valor del umbral utilizado para el cálculo. Por ejemplo, *mAP50* es la precisión promedio media calculada para el umbral de confianza 0.5 (50 %).

2.5. Redes Neuronales Convolucionales

Las CNNs son un tipo especializado de red neuronal que se utiliza principalmente para el procesamiento de imágenes y datos con estructura espacial; utilizan una serie de operaciones de convolución y agrupamiento para extraer características y patrones relevantes de los datos de entrada [12].

Como se muestra en la Figura 2.4, las CNNs están compuestas por capas de neuronas llamadas capas convolucionales, seguidas de capas de submuestreo o agrupamiento y capas completamente conectadas. A continuación se describe cada una de ellas:

- Capas convolucionales (*convolutional layer*): estas capas aplican filtros convolucionales a la imagen de entrada para detectar características locales, como bordes, texturas o formas. A la salida de esta capa se obtiene un conjunto de mapas de características (*features maps*) de la imagen analizada.

- Capas de submuestreo (*subsampling layer*): estas capas reducen la dimensionalidad de las características y resumen la información importante. Se aplican por lo general entre dos capas de convolución, reciben los *features maps* de la capa anterior y reducen su tamaño preservando las características más esenciales.

- Capas completamente conectadas (*fully connected layer*): estas capas realizan la clasificación o predicción basándose en las características extraídas. Se colocan al final de la arquitectura de la red y se conectan por completo a todas las neuronas de salida. Devuelven un vector de tamaño correspondiente al número de clases en el que cada componente representa la probabilidad de que la imagen de entrada pertenezca a una clase.

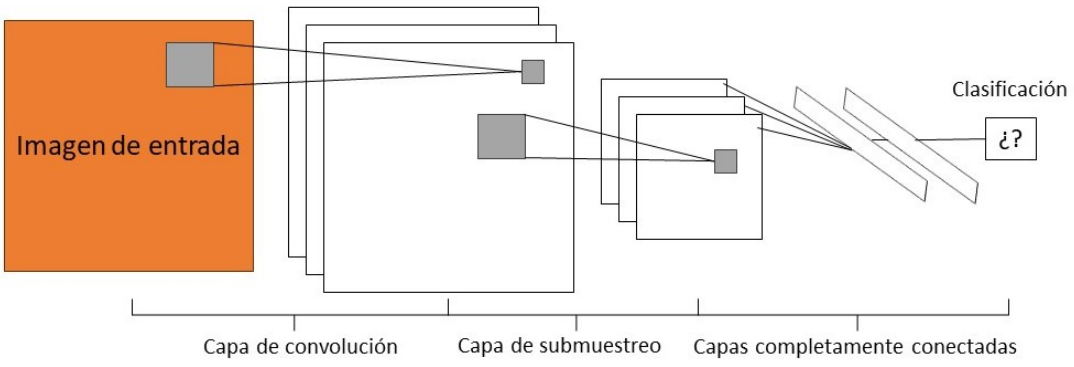

Figura 2.4: Arquitectura de una Red Neuronal Convolucional. Inspirada en [44].

Las CNNs han demostrado ser muy efectivas en tareas de visión por computadora, como la detección de objetos y la segmentación de imágenes [28]. Su capacidad para aprender características automáticamente a partir de los datos de entrada ha sido fundamental para el avance en estas actividades.

2.5.1. Arquitecturas para Detección

Existen varias arquitecturas de redes neuronales que han sido utilizadas para la detección de objetos en imágenes. De todas ellas, las más destacadas son las CNNs. En particular, debido a su gran popularidad y desempeño, se han utilizado arquitecturas como *Regions with Convolutional Neural Networks* (R-CNN) [11], Faster R-CNN [41] y *You Only Look Once* (YOLO) [40] para la detección de cáncer en imágenes médicas. A continuación, se describe la arquitectura de cada una de ellas con ejemplos de imágenes extraídas de videos de colonoscopias que sirven para detectar pólipos y otros tipos de lesiones precursoras del cáncer de colon.

2.5.1.1. R-CNN

R-CNN [11] es una técnica de detección de objetos en imágenes que se compone de tres etapas principales las cuales pueden ser observadas en la Figura 2.5:

a) Propuesta de regiones: Se utilizan técnicas de selección de regiones, como *Selective Search*, para generar varias regiones candidatas en la imagen que podrían contener objetos.

b) Extracción de características: Se extraen características de las regiones candidatas utilizando una CNN pre-entrenada.

c) Clasificación: Se utiliza una Máquina de Soporte Vectorial (SVM, *Support Vector Machine*) para clasificar las regiones candidatas como objeto o no objeto.

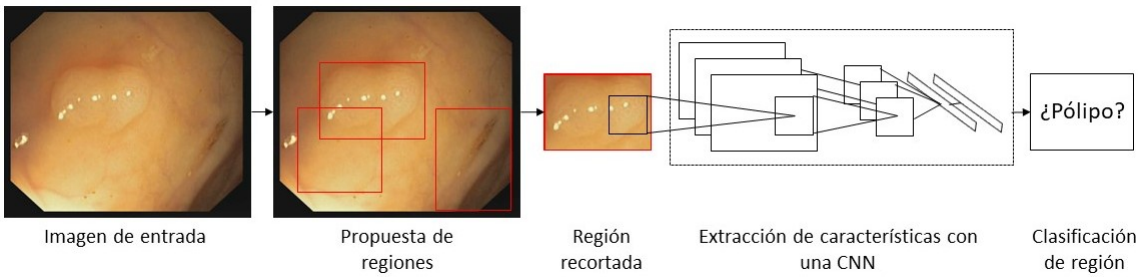

Figura 2.5: Arquitectura de R-CNN. Inspirada en [11].

R-CNN introdujo la idea de dividir una imagen en regiones propuestas antes de procesarlas para la detección de objetos con CNN. Esto permitió una detección más precisa y mejoró el rendimiento en comparación con enfoques anteriores que no utilizaban propuestas de regiones [11]. Aunque esta técnica fue revolucionaria en su momento, ha sido superada por arquitecturas más modernas, como Faster R-CNN y YOLO, que utilizan técnicas de detección de objetos más eficientes y precisas [41, 40].

2.5.1.2. Faster R-CNN

La arquitectura de Faster R-CNN [41] consta de dos componentes principales: una red de extracción de características y una red de propuestas de región.

La red de extracción de características se entrena en un conjunto de datos grande y diverso, como ImageNet [8], y se utiliza para extraer características significativas de la imagen de entrada. Luego, la red de propuestas de región utiliza estas características para generar propuestas de regiones de interés en la imagen, que son las áreas donde es más probable que se encuentren objetos de interés. Finalmente, estas propuestas se pasan a una red de detección, que se encarga de clasificar las propuestas de región como objetos o no objetos, y de estimar las coordenadas de la caja delimitadora que encierra el objeto en la imagen, como se muestra en la Figura 2.6.

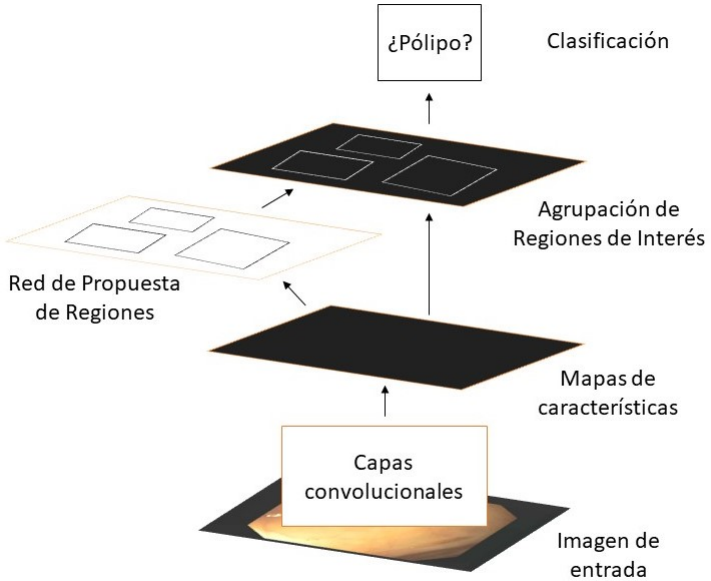

Figura 2.6: Arquitectura de las Faster R-CNN. Inspirada en [41].

La principal diferencia entre R-CNN y Faster R-CNN es que R-CNN utiliza una estrategia de "dos disparos" (*two-shot*) para generar las propuestas de región, mientras

que Faster R-CNN utiliza una red neuronal adicional llamada Red de Propuestas (RPN, *Region Proposal Network*) para generar las propuestas de región de manera más rápida y eficiente.

En resumen, Faster R-CNN es una mejora sobre R-CNN que utiliza una estrategia más eficiente para la generación de propuestas de región. Esto permite una detección de objetos más rápida y precisa, ya que la RPN se entrena simultáneamente con la red neuronal de detección de objetos.

2.5.1.3. YOLO

YOLO [40] es otra arquitectura popular de detección de objetos en imágenes, que se basa en redes neuronales convolucionales. La principal característica que distingue a YOLO de otras arquitecturas de detección de objetos es su enfoque de "único disparo" (*single-shot*), lo que significa que la red neuronal toma una imagen como entrada y produce las predicciones de detección de objetos en una sola pasada. A continuación, se explica su funcionamiento (Figura 2.7):

- División de la imagen de entrada: La imagen de entrada se divide en una cuadrícula de celdas. Cada celda es responsable de predecir la presencia o ausencia de objetos, así como las coordenadas del cuadro delimitador y las clases de los objetos dentro de esa celda.

- Predicciones del cuadro delimitador (*bounding box*): En cada celda de la cuadrícula, YOLO predice un cuadro delimitador que describe la ubicación y el tamaño de un objeto dentro de esa celda. Cada cuadro delimitador generalmente consta de cuatro coordenadas: coordenadas x e y del centro del cuadro y su *ancho* y *alto*. Además, YOLO predice la confianza de que un objeto esté presente en esa celda.

- Predicciones de clases: YOLO también realiza predicciones de clases para cada cuadro delimitador. Esto implica predecir la probabilidad de que el objeto dentro del cuadro delimitador pertenezca a cada clase de objeto conocida.

- Confianza y supresión no máxima: Para cada cuadro delimitador predicho, se calcula una puntuación de confianza que refleja la probabilidad de que un objeto esté presente en ese cuadro. Se aplica un umbral de confianza para eliminar cuadros con puntuaciones bajas. Luego, se aplica la Supresión No Máxima (NMS, *Non-maximum Suppression*) para eliminar cuadros duplicados y seleccionar el cuadro con la puntuación de confianza más alta.

- Combinación de predicciones: Las predicciones de todas las celdas se combinan para obtener la salida final de la red. Esto da como resultado una lista de cuadros delimitadores con sus clases y puntuaciones de confianza.

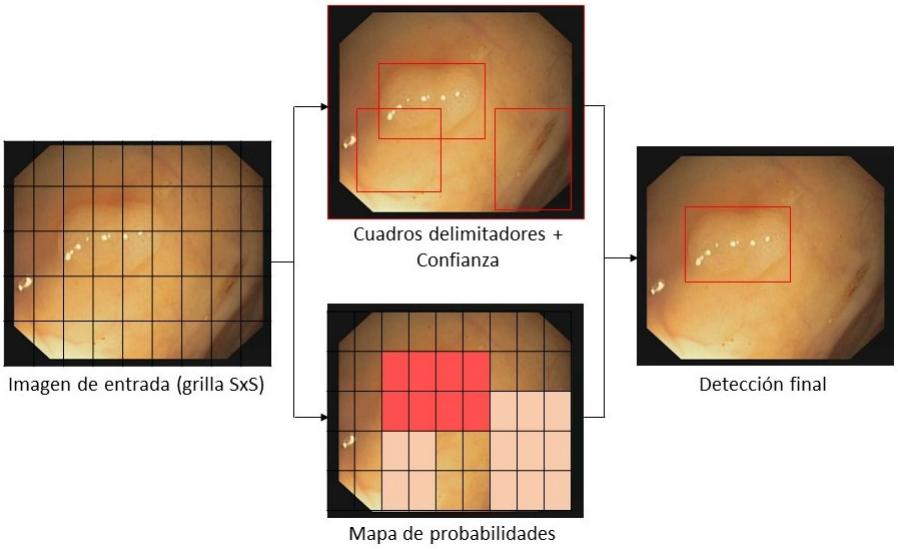

Figura 2.7: Arquitectura de YOLO. Inspirada en [40].

En resumen, YOLO es un modelo de detección de objetos que divide la imagen de entrada en celdas y realiza predicciones en cada celda para detectar objetos y clasificarlos en una sola pasada por la red. Esto lo hace rápido y eficiente, y ha llevado a su popularidad en aplicaciones de detección de objetos en tiempo real.

2.6. Análisis de imágenes digitales mediante redes profundas

2.6.1. ¿Qué son las imágenes digitales?

Las imágenes digitales son representaciones visuales de objetos o escenas que se capturan mediante dispositivos como cámaras digitales, escáneres o equipos médicos de diagnóstico por imagen como Radiografía (X-Ray, *X-Ray*), Resonancia Magnética (MRI, *Magnetic Resonance Imaging*) o Tomografía Computada (CT, *Computed Tomography*). Estas imágenes se almacenan en formato digital y se pueden procesar, analizar y compartir fácilmente utilizando herramientas de software [36].

Las imágenes médicas desempeñan un papel fundamental en el diagnóstico de enfermedades, ya que permiten a los profesionales de la salud observar el interior del cuerpo para buscar indicios de una afección médica. En el caso específico del cáncer,

los estudios por imágenes se pueden usar para detectar la enfermedad, saber cuán lejos se ha propagado o hasta observar si el tratamiento está surtiendo efecto [45]. En la detección del cáncer de colon, el estudio más común es la colonoscopia [53].

Una colonoscopia es un procedimiento que usa un médico para observar el interior del colon y del recto con un instrumento denominado colonoscopio, que consiste en un tubo flexible del grosor de un dedo que tiene una luz y una pequeña cámara de video en uno de sus extremos. Se introduce por el ano y se lleva hasta el recto y el colon [46]. En la Figura 2.8 se puede observar un ejemplo de este estudio.

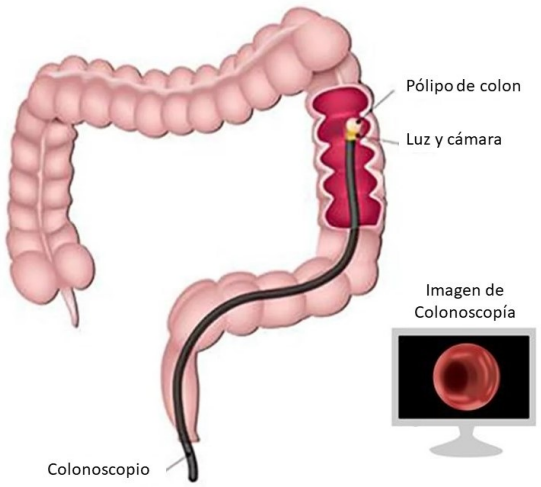

Figura 2.8: Ejemplo de Colonoscopia. Inspirada en [7].

Estas imágenes colonoscopias digitales se pueden utilizar para identificar lesiones, pólipos o masas sospechosas en el colon. Los médicos examinan las imágenes y buscan características específicas, como la forma, el tamaño y los patrones de color, que pueden indicar la presencia de tumores benignos o precursores del cáncer de colon. El análisis y la interpretación de estas imágenes pueden ser parciales y dependen en gran medida de la experiencia del médico.

2.6.2. Detección en imágenes médicas

La detección tiene un papel crucial en el análisis de imágenes médicas para diagnosticar enfermedades. En los estudios realizados, el DL ayuda al diagnóstico de cáncer de colon mediante la detección de núcleos en imágenes con patologías de dicho cáncer y la posterior clasificación en células malignas o benignas. Especialmente en las imágenes de colonoscopias, la detección de pólipos es el paso más importante en la

etapa temprana del cáncer o en la detección sin cáncer. También es la técnica elegida para el diagnóstico precoz y la prevención del cáncer de colon [34].

El propósito de una colonoscopia es encontrar y eliminar los pólipos de colon que son los precursores de los cánceres de colon [46]. Las formas, los colores y los tamaños de estos pólipos pueden variar. Asimismo, la calidad del dispositivo de colonoscopia utilizado y las imágenes de los pólipos a obtener pueden variar. Con todos estos factores, problemas como los pólipos que son difíciles detectar puede hacer que los profesionales médicos duden y, a veces, obtengan resultados incorrectos. En general, la tasa de pérdida de pólipos varía del 6 % al 27 % [34]. Obtener resultados incorrectos puede causar que el cáncer de colon se propague y se vuelva metastásico, lo cual disminuye la tasa de supervivencia de un paciente con dicho cáncer.

Por todo lo mencionado anteriormente, el DL tiene un lugar importante en la detección de pólipos. Especialmente con los estudios recientes, el éxito en las tasas de detección de pólipos ha aumentado y se han llevado a cabo más estudios.

2.6.3. Trabajos relacionados

El campo de la detección de cáncer de colon a partir de imágenes de colonoscopias ha experimentado un gran avance en las últimas décadas gracias a los progresos en el procesamiento de imágenes y la inteligencia artificial.

En el año 2015, se organizó entre laboratorios y universidades de España, Francia y Estados Unidos una competencia internacional denominada "Endoscopic Vision Challenge" como parte de *Medical Image Computing and Computer Assisted Intervention* (MICCAI) para evaluar los algoritmos de detección de pólipos y cáncer de colon a partir de imágenes de colonoscopias.[1] La misma marcó un antes y un después en las investigaciones del área, ya que permitió evaluar y comparar diferentes métodos de detección de cáncer de colon de forma más justa, ya que todos debían utilizar los mismos conjuntos de datos y métricas. A continuación, se examinan algunos de los estudios realizados a partir de dicha competencia.

En el año 2018, Shin et al. [43] propusieron un enfoque de detección de pólipos de colon basado en la red Faster R-CNN. Aplicaron una CNN profunda basada en regiones para la detección automática de pólipos en videos de colonoscopias, utilizando la Faster R-CNN con un modelo *Inception-Resnet* como esquema para la transferencia de aprendizaje. Además, usaron diferentes estrategias de aumento de imágenes para superar los obstáculos de detección de pólipos y la pequeña cantidad de imágenes

[1] Automatic Polyp Detection in Colonoscopy Videos, Endoscopic Vision Challenge, 2015

para entrenar las redes neuronales. Su modelo obtuvo mejores resultados que otras aproximaciones.

En el 2018, Zheng et al. [59] presentaron un modelo de CNN profunda basado en la red YOLO. En comparación con los detectores típicos basados en clasificación, como R-CNN, YOLO evita generar las regiones de interés mediante métodos de propuesta adicionales, lo que la hace mucho más rápida. Además, utilizaron transferencia de aprendizaje y técnicas de aumento de imágenes para mejorar el rendimiento de la red.

En el año 2019, Liu et al. [27] utilizaron un detector de un solo golpe (SSD, *Single Shot Detector*) para detectar los pólipos en videos de colonoscopias. Evaluaron tres arquitecturas de CNN diferentes que son comúnmente utilizadas para extraer las características de las imágenes: *ResNet50*, *VGG16* e *InceptionV3*. Esta última fue la que obtuvo mejores resultados en precisión y sensibilidad.

Sornapudi et al. [49] propusieron un algoritmo basado en R-CNN modificado para formar máscaras alrededor de los pólipos detectados. Utilizaron las arquitecturas *ResNet50* y *ResNet101* para la extracción de características. Además, los modelos fueron pre entrenados con tres conjuntos de datos diferentes: COCO, ImageNet y Ballon. Los mejores resultados los obtuvieron con la red *ResNet101* pre entrenada con las imágenes de Ballon.

Wang et al. [55] propusieron un nuevo método de una sola etapa para la detección de pólipos en tiempo real denominado *AFPNet*, por las palabras *Anchor Free Polyp Network*, que hace referencia a una red de detección "sin anclaje". En este contexto, lograron algunas mejoras con una arquitectura basada en la red *CenterNet* para realizar detecciones en tiempo real. También evaluaron diferentes arquitecturas para la extracción de características, consiguiendo mejores resultados con la red *VGG16* en lugar de *ResNet50* y *ResNet101*.

En el año 2020, Jia et al. [18] presentaron un enfoque en dos etapas llamado *PLPNet*, donde la abreviatura "PLP" hacía referencia a la palabra "pólipo". En primer lugar, construyeron una etapa de propuesta de pólipos como una extensión de Faster R-CNN, que funciona como un detector de pólipos a nivel de región para reconocer el área de la lesión como un todo y constituye la primera etapa de *PLPNet*. La segunda etapa se basó en una red convolucional para la segmentación a nivel de píxel. Comparado con las otras arquitecturas, la *PLPNet* mejora la precisión de reconocimiento, agregando una etapa que predice el cuadro de localización con presencia de pólipos.

En el año 2021, Xu et al. [56] propusieron una YOLO versión 3 modificada, donde agregaron un módulo denominado FPRM (*False Positive relearning module*) para reducir la presencia de falsos positivos, reaprendiendo de sus características, y otro módulo ISTM (*Image Style transfer module*) que renderiza una imagen en otro estilo ayudando

a aprender mejor la forma de los pólipos.

Pacal et al. [33] hicieron un estudio profundo sobre la YOLO en su versión 4, la cual en ese momento era la más reciente, y obtuvieron los mejores resultados optimizándola con la utilización de la red neuronal CSPNet (*Cross Stage Partial Network*) en toda la arquitectura. Luego, en 2022, Pacal junto con otros colaboradores [35] analizaron diferentes arquitecturas de las redes neuronales YOLO en sus versiones 3 y 4 incorporando dos conjuntos de datos nuevos (SUN y PICCOLO). Una vez más, compararon sus resultados con los del estado del arte, con el objetivo de examinar los efectos de agregar más imágenes de entrenamiento, y lograron obtener mejores resultados.

Recientemente, en el año 2023, Krenzer et al. [23] implementaron un sistema de detección en tiempo real, de acceso libre (*open-source*) y listo para ser usado en aplicaciones clínicas. El mismo está basado en la red neuronal YOLO versión 5 de *Ultralytics* y con una etapa de post-procesamiento a la que denominaron *RT-REPP*, por sus siglas en inglés *Real-Time Robust and Efficient Post-Processing*. Esta etapa se añadió para mejorar la detección en tiempo real una vez que la red había sido entrenada y evaluada. Utilizaron una combinación de conjuntos de datos públicos y privados para obtener la mayor cantidad de imágenes posibles y lo testearon en el conjunto de datos *CVC-VideoClinicDB*, obteniendo uno de los resultados con mayor precisión de detección de la literatura.

A modo de resumen, se muestra en la Tabla 2.1 una comparación de los diferentes modelos de la literatura aplicados sobre el conjunto de datos de prueba *ETIS-Larib* de la competencia para la detección de pólipos organizada en 2015.

Tabla 2.1: Comparación de resultados de diferentes modelos del estado del arte sobre el conjunto de datos ETIS-Larib.

Autor	Modelo	Prec.	Recall	F1	F2
Shin et al., 2018 [43]	Faster R-CNN	86.50	80.30	83.30	81.50
Zheng et al., 2018 [57]	YOLO	76.00	66.80	71.10	68.50
Liu et al., 2019 [27]	SSD	73.60	80.30	76.80	78.90
Sornapudi et al., 2019 [49]	R-CNN	72.93	80.29	76.43	78.70
Wang et al., 2020 [55]	AFPNet	88.89	80.77	84.63	82.27
Jia et al., 2020 [18]	PLPNet	63.90	81.70	71.70	77.40
Xu et al., 2021 [56]	YOLOv3 Modificada	83.24	71.63	77.00	73.69
Pacal et al., 2021 [33]	YOLOv4-CSP	91.62	82.55	86.85	84.00

La tabla se divide de la siguiente manera: La primer columna indica el autor junto con el año de publicación del trabajo. En la segunda columna se detalla el modelo

utilizado. Y, en las columnas restantes, se muestran los resultados de las métricas de evaluación seleccionadas para la competencia.

Como se puede observar, el mejor resultado lo obtiene Pacal utilizando la versión 4 de la red neuronal YOLO. Luego, en segundo lugar, se encuentra la *AFPNet*, que es un detector de una sola etapa similar a la arquitectura de YOLO. Y, en tercer lugar, la Faster R-CNN, que como se vio anteriormente, mejora el rendimiento de la R-CNN.

2.7. Desarrollo ágil de prototipos mediante Programación Extrema

El proceso para el desarrollo de software contempla una serie de actividades relacionadas que conduce a la elaboración de un producto de software. Existen varios modelos a seguir, cada uno de los cuales describe un enfoque diferente para llevar a cabo las actividades que ocurren durante el proceso [37].

El desarrollo de software ágil es un enfoque de gestión de proyectos y desarrollo de software que se centra en la colaboración, la adaptabilidad y la entrega iterativa de productos de software de alta calidad. Se desarrolló como una respuesta a los enfoques tradicionales de desarrollo de software, como el modelo en cascada, que a menudo resultaban en proyectos largos, costosos y propensos a cambios de requisitos [47].

Existen varios marcos de trabajo ágil populares, como Scrum [51], Kanban [9] y XP [3], que proporcionan estructuras y prácticas específicas para implementar estos principios en el desarrollo de software. Dentro de los diferentes modelos de desarrollo ágil, la metodología XP presenta diferentes características que la hacen especial para el desarrollo de prototipos en grupos pequeños o bien personales, ya que se centra en la implementación rápida y flexible y es adecuada para proyectos a corto plazo.

La Programación Extrema (XP, *Extreme Programming*) fue formulada por Kent Beck en el año 2000 como una metodología ligera para pequeños y medianos equipos de desarrollo de software en la cara de los requerimientos imprecisos o rápidamente cambiantes, llevando a niveles “extremos” las prácticas reconocidas de ingeniería de software [3].

Es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad, con ciclos de desarrollo cortos y con menos documentación. En la Figura 2.9 se muestra las diferentes fases que incluye esta metodología.

Este proceso es iterativo e incremental, por lo tanto, el ciclo se repite cada un tiempo determinado (aproximadamente dos semanas) liberando incrementos del sistema cada vez que se llega a la última fase.

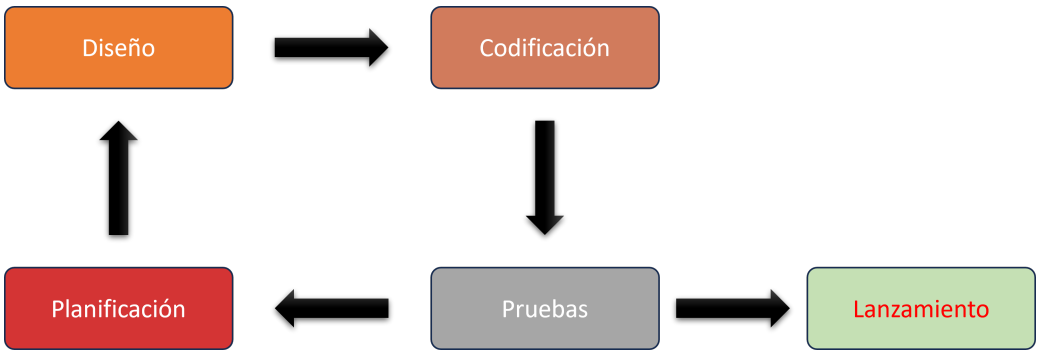

Figura 2.9: Fases de la metodología XP

El objetivo de cada fase es el siguiente:

- **Planificación:** En esta fase se identifican las historias de usuario, que son tarjetas simples donde se describen los requerimientos, y se descomponen en tareas a realizar para obtener una miniversión del sistema.
- **Diseño:** En este paso se dedica al diseño de un código sencillo, haciendo lo mínimo imprescindible para que funcione, y se obtiene el prototipo de la versión en la que se está trabajando.
- **Codificación:** En esta etapa se procede a la programación del código. Debe realizarse en parejas, sin embargo, en este caso se realizó de manera individual.
- **Pruebas:** En este último paso se realizan pruebas para testear el funcionamiento de la versión alcanzada.
- **Lanzamiento:** Esta fase consiste en la liberación del producto.

Capítulo 3

Diseño y desarrollo del trabajo

Para llevar a cabo los objetivos propuestos en este trabajo final es necesario definir una metodología consistente que permita planificar y gestionar las diferentes tareas y recursos de manera ordenada.

Este capítulo detalla el diseño de la metodología de trabajo utilizada junto los métodos a través de los cuales se pretende llegar a obtener resultados (Sección 3.1) y las tecnologías de hardware y software utilizadas (Sección 3.2).

En la Sección 3.1.1 se describe el algoritmo seleccionado para realizar los experimentos sobre los diferentes conjuntos de datos. La Sección 3.1.2 describe detalladamente las instancias de datos utilizadas y en la Sección 3.1.3 se detalla el preprocesamiento de dichos datos. Luego, en la Sección 3.1.4 se presentan los experimentos realizados para la evaluación del modelo seleccionado y se brinda una breve descripción de la interpretación de los resultados en la Sección 3.1.5. Finalmente, se describe el desarrollo del prototipo de la herramienta de visualización en la Sección 3.1.6.

3.1. Metodología general de trabajo

En esta sección se describen todos los pasos establecidos como parte del proceso llevado a cabo para desarrollar el sistema de detección en imágenes de cáncer de colon.

La secuencia de pasos definida en este trabajo final se puede observar en la Figura 3.1. En primer lugar, a partir de lo analizado en el capítulo anterior, se hace la selección del modelo algorítmico a evaluar, en este caso la ultima versión de YOLO. Posteriormente, se realiza la adquisición y preprocesamiento a los conjuntos de imáge-

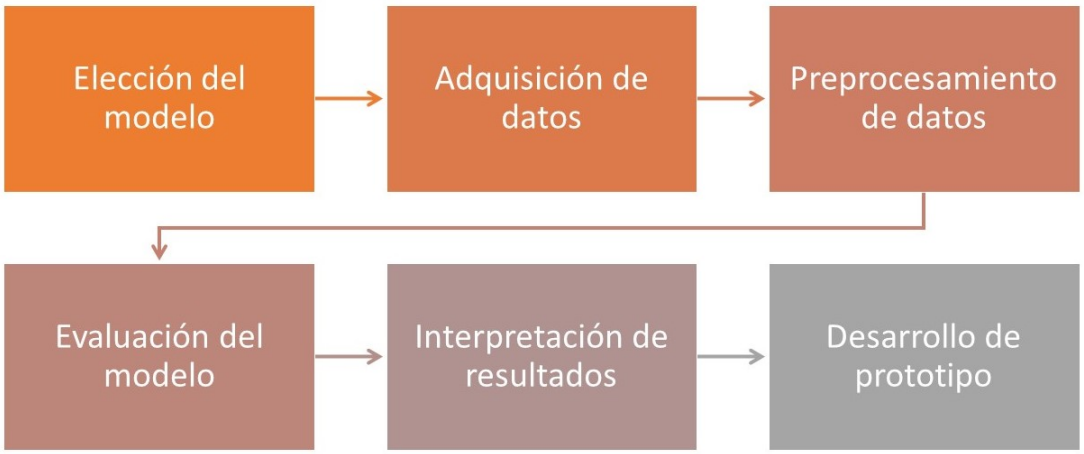

Figura 3.1: Metodología propuesta

nes seleccionados para adaptarlos al formato requerido por YOLO y que puedan servir como datos de entrada a la red. A continuación, se definen diferentes evaluaciones en las cuales se entrena la red en diferentes aspectos de rendimiento, teniendo en cuenta la configuración de los conjuntos de datos y los diversos hiperparámetros que presenta YOLO. Estas pruebas tienen en cuenta configuraciones previas obtenidas a partir del estado del arte. Seguidamente, se interpretan los resultados obtenidos mediante la utilización de las métricas de rendimiento previamente explicadas en el capítulo anterior. Se busca encontrar una configuración o un conjunto de ellas que representen de mejor manera la capacidad de detección y el rendimiento numérico de la red. Además, se prueba el uso de técnicas de transferencia de aprendizaje. Finalmente, a partir de la configuración seleccionada como la más robusta y confiable, se realiza la implementación de una herramienta prototipo para visualizar la detección producida por la red. En las siguientes secciones se explica cada unos de los pasos con mayor detalle.

3.1.1. Elección del modelo algorítmico

En la Sección 2.5.1 se describieron algunas de las arquitecturas más utilizadas para la detección de objetos en imágenes, haciendo hincapié principalmente en su uso para la detección del cáncer de colon en imágenes médicas. Entre ellas, se destaca YOLO debido a su enfoque de "único disparo" y su capacidad de detección en tiempo real. Luego, en la Sección 2.6.3 se presentaron los trabajos relacionados con este campo. En los últimos trabajos se utilizó la arquitectura YOLO en su versión 4 y 5, que era la más reciente hasta el momento, y se obtuvieron muy buenos resultados en comparación a otros modelos de la literatura. A principios de 2023 se diseñó y publicó la YOLO en

su versión 8 (*YOLOv8*), la cual está basada en las versiones anteriores, introduciendo nuevas características y mejoras para aumentar el rendimiento, la flexibilidad y la eficiencia [19].

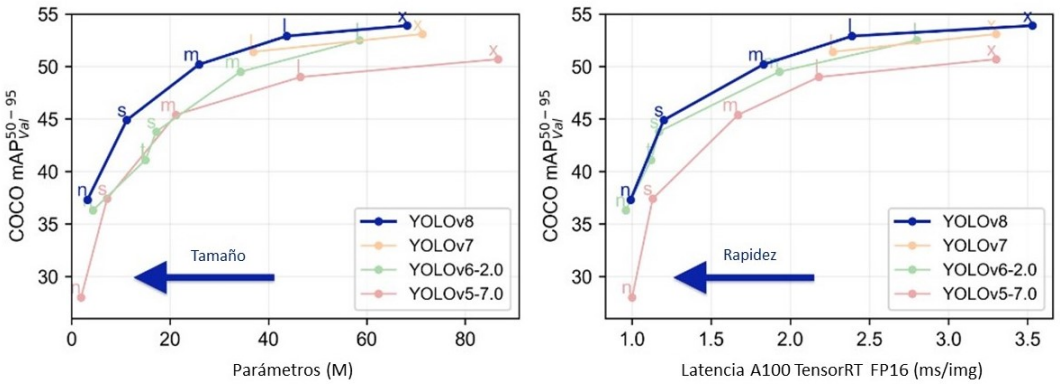

Figura 3.2: Comparación de las últimas versiones de YOLO sobre el conjunto de datos COCO. Extraída de [19].

En la Figura 3.2 se muestran dos gráficos comparativos de las últimas versiones de YOLO donde se evalúa su rendimiento en precisión media (*mAP*) sobre el conjunto de datos COCO [26], en base al tamaño y la rapidez de la red. Se puede observar cómo la *YOLOv8* (marcada en línea color azul) supera a las versiones anteriores en términos de precisión en ambos gráficos. En el lado izquierdo se muestra como varía la precisión basándose en el tamaño de la red, mostrando claramente que el modelo YOLOv8 escala de mejor forma sin perder la capacidad de precisión. Por otro lado, en el lado derecho se puede observar la variación de la precisión con base en la latencia de la Unidad de Procesamiento Gráfico (GPU, *Graphics Processing Unit*) NVIDIA 100. Estas mejoras a nivel algorítmico permiten considerar la red *YOLOv8* como un excelente algoritmo de partida para entrenar y evaluar en el problema de la detección del cáncer de colon.

Cada versión de YOLO se divide a su vez en *n* (*nano*), *s* (*small*), *m* (*medium*), *l* (*large*) y *x* (*extra large*) según el tamaño de la red. Los modelos más pequeños tienen menor cantidad de parámetros y capas en comparación con los modelos más grandes, por lo que son más rápidos de entrenar y ejecutar. Sin embargo, pueden ser menos precisos, ya que tienen menor capacidad para aprender características complejas. Además, el tamaño del modelo influye en los recursos de *hardware* necesarios para entrenar la red neuronal, ya que mientras más grande sea la red requiere más memoria y potencia de cálculo. En la Tabla 3.1 se puede observar mejor las diferencias de las distintas versiones de YOLOv8 mencionadas anteriormente en cuanto a precisión media (*mAP*), rapidez (latencia) y tamaño (cantidad de parámetros).

Modelo	mAP50-95	Latencia (ms)	Parámetros (M)
YOLOv8n (*nano*)	37.3	0.99	3.2
YOLOv8s (*small*)	44.9	1.20	11.2
YOLOv8m (*medium*)	50.2	1.83	25.9
YOLOv8l (*large*)	52.9	2.39	43.7
YOLOv8x (*extra large*)	53.9	3.53	68.2

Tabla 3.1: Comparación de las distintas versiones de YOLOv8. Extraída de [19].

En este trabajo se utiliza la versión *nano*, ya que contiene la menor cantidad de parámetros y capas, lo que hace que sea mucho más rápida que las demás y que consuma menos recursos. Asimismo, al evaluar la capacidad innata de la red más pequeña en sistemas de cómputos hogareños, se puede brindar una idea general del funcionamiento del nuevo modelo y cómo puede impactar su utilización en problemas de detección de imágenes.

3.1.2. Adquisición de datos

Para evaluar el funcionamiento del algoritmo seleccionado es necesario contar con diferentes conjuntos de datos de imágenes que permitan una evaluación completa. Al utilizar distintos conjuntos de datos y mayor cantidad de imágenes, se obtiene mayor variabilidad, lo cual influye en la capacidad de generalización del algoritmo y en el aprendizaje de la red. A continuación, se describen los conjuntos de datos seleccionados para realizar las tareas de detección de cáncer.

a) CVC-ClinicDB [5]: es el conjunto de datos de entrenamiento que se usó en la competencia MICCAI de 2015 sobre detección automática de pólipos en videos de colonoscopias, por lo que es uno de los más utilizados en el estado del arte. Contiene 612 imágenes de pólipos extraídas de 31 secuencias diferentes de videos de colonoscopias. Cada imagen cuenta con una máscara que corresponde a la región donde se encuentra el pólipo. La resolución de las imágenes es de 384 × 288 píxeles. El conjunto de datos es de acceso abierto.

b) CVC-ColonDB [4]: es el primer conjunto de datos de acceso abierto creado con el objetivo de facilitar la comparación entre diferentes métodos, por lo que contiene una gran variabilidad en tipos de pólipos. Se publicó en 2012. Contiene 300 imágenes de 574 × 500 píxeles de resolución extraídas de 15 secuencias cortas de videos de colonoscopias. Cada imagen cuenta con su máscara correspondiente. Si bien es de acceso abierto el enlace original de descarga se encuentra deshabilitado

por lo que se utilizó un enlace alternativo que cuenta con 380 imágenes en lugar de 300.

c) Kvasir-SEG [17]: es un conjunto de datos más nuevo, publicado en el 2020. Contiene 1000 imágenes de pólipos con sus correspondientes máscaras. La resolución de las imágenes varía de 332 × 487 a 1920 × 1072 píxeles. Además, son más brillantes y de mayor contraste. Este también es de acceso abierto.

El propósito y la información principal de cada conjunto de datos se encuentra resumida en la Tabla 3.2.

Conjunto de datos	Propósito	Total de imágenes	Resolución	Enlace de acceso
CVC-ClinicDB [5]	Entrenamiento/ Validación	612	384 × 288	CVC-ClinicDB
CVC-ColonDB [4]	Prueba	300	574 × 500	CVC-ColonDB*
Kvasir-SEG [17]	Entrenamiento/ Validación	1000	de 332 × 487 a 1920 × 1072	Kvasir-SEG

**Enlace alternativo a CVC-ColonDB*

Tabla 3.2: Información de los conjuntos de datos seleccionados

3.1.3. Preprocesamiento de los datos

Para entrenar la red neuronal YOLOv8 con los conjuntos de datos seleccionados se requiere adaptar los mismos al formato de configuración especificado en la documentación de la herramienta utilizada para trabajar con YOLO[1]. Para que la red pueda trabajar con los conjuntos de imágenes se necesita seguir los siguientes pasos.

Generación de cuadros delimitadores (*bounding boxes*): Para llevar a cabo la detección de un objeto en una imagen es necesario indicar si dicho objeto se encuentra en la imagen y su posición. El *bounding box* es un cuadro formado por cuatro valores que sirve para señalar los límites del objeto dentro de una imagen. En YOLO se utiliza el siguiente formato: (*x_center y_center width height*), donde *x_center* e *y_center* indican las coordenadas del centro del objeto y *width* y *height* indican el ancho y alto respectivamente.

Las coordenadas del *bounding box* deben estar normalizadas, es decir, con valores entre 0 y 1. Para normailzar se debe dividir *x_center* y *width* por el ancho de la imagen e *y_center* y *height* por la altura de la imagen.

[1]Documentación de Ultralytics

Los *labels* indican la clase a la que pertenece el objeto. En YOLO se debe crear un archivo *.txt por imagen que contenga una fila por cada objeto indicando el número de clase a la que pertenece dicho objeto (comenzando con 0), seguido de las coordenadas de su *bounding box*. Si no hay objetos en una imagen, no se requiere ningún archivo.

Como los conjuntos de datos seleccionados contienen las imágenes con sus respectivas máscaras, se creó un *script* de Python (Anexo A.2.1) para extraer los bordes de esas máscaras, y una vez obtenido el *bounding box*, se le da el formato especificado anteriormente y se crea un archivo *.txt por imagen. En la Figura 3.3 se muestra la primera imagen de cada conjunto de datos con su correspondiente máscara y el cuadro delimitador generado.

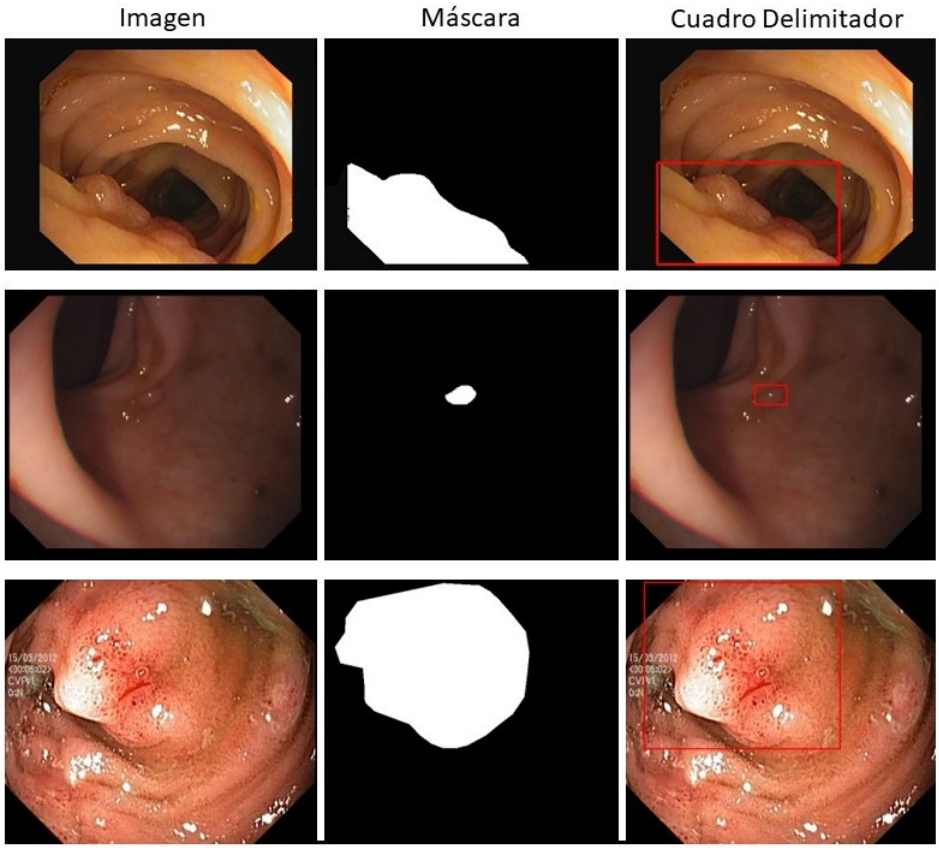

Figura 3.3: Primera imagen de cada conjunto de datos con su correspondiente máscara y cuadro delimitador, de arriba hacia abajo: CVC-ClinicDB, CVC-ColonDB y Kvasir-SEG.

División de conjuntos de datos: En todo proceso de ML los datos se dividen en conjuntos de entrenamiento (*training*), validación (*validation*) y prueba (*testing*).

Los datos de entrenamiento se utilizan para actualizar los pesos de cada *batch*, los de validación comprueban la capacidad de generalización de la red en cada *epoch* y los de evaluación dan una intuición de lo buena que es la red al generalizar con un conjunto nunca visto. Hacer una buena división de los conjuntos de datos es fundamental para obtener un modelo que se ajuste bien a dichos datos.

Sobre la base de los diferentes experimentos llevados a cabo en el estado del arte, se decidió utilizar los conjuntos de datos CVC-ClinicDB y Kvasir-SEG para el entrenamiento y validación de la red. El conjunto de imágenes CVC-ColonDB se utilizó para la prueba del modelo debido a que es uno de los más usados en la literatura para dicha tarea. CVC-ClinicDB se dividió en 536 imágenes de entrenamiento y 76 de validación [49] y Kvasir-SEG se dividió en 880 imágenes para entrenamiento y 120 para validación [16]. Si bien una de las reglas más comunes es dividir los conjuntos de datos en un 80/20, en este caso para ambos conjuntos se utilizó un 88 % de imágenes para el entrenamiento y un 12 % para la validación. La elección de realizar dicha división para los distintos conjuntos de datos se basó en lo observado en el estado del arte, ya que los trabajos que utilizan estos conjuntos los dividen de esa manera. Esto se debe a que no presentan una gran cantidad de imágenes, por lo que se decide aumentar un poco el conjunto de entrenamiento para que la red aprenda mayor variedad de datos.

Organización de directorios: Una vez divididos los conjuntos, se organizaron las imágenes en diferentes carpetas y se creó un archivo de configuración YAML (Código 1) donde se especificaron las rutas relativas a los directorios que contenían las imágenes de entrenamiento, validación y prueba para cada experimento junto con el número y nombre de clases a detectar, en este caso: *pólipo*. A continuación, se puede ver un ejemplo de este archivo para los experimentos realizados con el conjunto de datos CVC-ClinicDB.

Código 1 Ejemplo de archivo de configuración YAML

```yaml
# Train/val/test sets
path: ../datasets/CVC-ClinicDB  # dataset root dir
train: images/train  # 536 images
val: images/val  # 76 images
test: images/test # optional
# Classes (1 class)
names:
  0: polyp
```

3.1.4. Evaluación del modelo algorítmico

Una vez realizado el preprocesamiento de las imágenes y preparado todo el entorno de trabajo, se procede a evaluar la red neuronal. Esta evaluación consiste en una serie de pruebas diseñadas para valorar el comportamiento y el rendimiento numérico del modelo seleccionado.

1. **Evaluación numérica para CVC-ClinicDB y Kvasir-SEG:** En primer lugar, se llevaron a cabo experimentos iniciales sobre estos conjuntos de datos para probar adecuadamente los hiperparámetros utilizados en la literatura, los cuales mostraban un comportamiento más eficiente y posibilitaban funcionar de manera más robusta los modelos, permitiendo mejorar el rendimiento de la red neuronal. Por ello, se seleccionaron diferentes valores para el número de *epochs*, el *batch size*, el *optimizer* y el *learning rate* y se utilizó la búsqueda de cuadrícula (*grid search*), que consiste en construir un modelo por cada posible combinación de todos los valores de hiperparámetros proporcionados. A continuación, se muestra la tabla con los valores elegidos finalmente, los cuales fueron cuidadosamente seleccionados con base en los resultados obtenidos en los trabajos mencionados en la Sección 2.6.3:

epochs	{100,200}
batch size	{16,32}
optimizer	{SGD,ADAM}
learning rate	{0.001,0.0001}

Tabla 3.3: Selección de hiperparámetros

   Además, las pruebas sobre la YOLOv8 se realizaron entrenando la red desde cero (*from scratch*) y utilizando la técnica de transferencia de aprendizaje (*transfer learning*) para evaluar la influencia de las mismas en el rendimiento y eficiencia de la red. En el caso de *transfer learning*, se usaron los pesos de una red previamente entrenada con el conjunto de datos COCO de Microsoft [26].

2. **Evaluación con respecto al estado del arte:** Luego de entrenar y validar la red con los conjuntos de datos previamente mencionados, se seleccionaron las mejores configuraciones obtenidas para compararlas con otros modelos algorítmicos del estado del arte. Para ello, se evaluó el modelo sobre un nuevo conjunto de datos (CVC-ColonDB), el cual fue utilizado exclusivamente para probar la red neuronal, tal como se hizo en los otros trabajos con los que se realiza la comparación.

3. **Evaluación a partir de combinación de conjuntos de datos:** Finalmente, se realizaron otros experimentos donde se evaluó la red combinando los conjuntos de datos que se utilizaron para el entrenamiento (CVC-ClinicDB y Kvasir-SEG) con el fin de examinar los efectos de ampliar el conjunto de datos de entrenamiento y evaluar los mejores modelos obtenidos sobre cada conjunto por separado y sobre el conjunto de datos que surge de la unión de ambos.

3.1.5. Interpretación de los resultados

Los resultados surgen a partir de las métricas utilizadas para evaluar los modelos. Las métricas como *precision*, *recall* y *F1* brindan información acerca de qué tan bien detecta el algoritmo. La interpretación de los resultados implica analizar y comprender cómo y por qué se obtienen esos valores. Esto permite validar la confiabilidad de los modelos, identificar posibles sesgos y errores, y tomar decisiones informadas basadas en la información que brindan los resultados. Además, esto permite validar los modelos, su precisión y eficacia. También se pueden detectar sesgos o prejuicios en los modelos, lo cual en aplicaciones críticas es muy importante discernir.

3.1.6. Desarrollo de prototipo de visualización

A partir de la metodología de trabajo seleccionada se definió una planificación que consiste de dos iteraciones en las cuales se siguen las etapas de XP: planificación, diseño, codificación, pruebas y liberación de un entregable a partir de la identificación del caso de uso.

3.2. Tecnologías y herramientas

En esta sección, se describen las diversas herramientas y tecnologías empleadas en el proceso de desarrollo del sistema de detección, así como en el entrenamiento y evaluación de la red neuronal. Estas herramientas fueron elegidas en función de sus capacidades para satisfacer los requisitos funcionales y técnicos del trabajo, al igual que para lograr una implementación eficiente y efectiva.

3.2.1. Lenguajes de Programación

El sistema se desarrolló utilizando una combinación de lenguajes de programación que permitieron abordar diferentes aspectos de la funcionalidad requerida.

PHP (Versión 8.2.4): Se utilizó *Hypertext Preprocessor* (PHP) como lenguaje principal para la lógica de los servicios del lado del servidor. PHP es conocido por su capacidad para generar contenido dinámico en el lado del servidor y su integración con servidores web como Apache.

Python (Versión 3.11): Se incorporó Python para realizar tareas específicas, como el procesamiento de imágenes y la ejecución de la red neuronal para la detección de cáncer de colon. Python es ampliamente utilizado en aplicaciones de procesamiento de datos y aprendizaje automático debido a su gran cantidad de bibliotecas y frameworks disponibles.

3.2.2. Entorno de Desarrollo Integrado (IDE)

Un IDE eficiente y adecuado es fundamental para la productividad y la calidad del código durante el proceso de desarrollo. En este proyecto, se utilizó Visual Studio Code como entorno principal para la codificación y depuración. Además, se utilizó Anaconda para gestionar las dependencias y configuraciones de Python necesarias para el entrenamiento de la red neuronal. Anaconda facilitó la creación de entornos virtuales y la instalación de paquetes.

3.2.3. Servidor Web

El sistema se ejecuta en un entorno de servidor web para estar accesible en línea. Se optó por XAMPP debido a su facilidad de instalación y configuración, lo que permitió una rápida implementación del entorno de desarrollo en el ámbito local.

3.2.4. Bibliotecas y Frameworks

La funcionalidad específica requerida para el procesamiento de imágenes y el uso de la red neuronal se logró mediante el uso de bibliotecas y frameworks especializados.

Ultralytics (Versión 8.0.x): Se utilizó la implementación de YOLOv8 de Ultralytics con el fin de poder trabajar sobre modelos de la red neuronal y entrenarla para la detección de cáncer de colon para luego realizar las predicciones. Esta herramienta permitió ajustar el proceso de entrenamiento y realizar un seguimiento detallado de métricas clave para la evaluación de los modelos.

Pillow (Versión 10.0.0): La biblioteca Pillow se utilizó para el procesamiento y manipulación de imágenes en el sistema. Esta biblioteca ofrece una amplia gama de funciones para cargar, editar y guardar imágenes en varios formatos.

Scikit-image (Versión 0.21.0): Esta biblioteca se utilizó para el pre procesamiento de las imágenes. Permitió extraer las características de las máscaras para generar los cuadros delimitadores.

Os (Versión 3.11): Este módulo proporciona herramientas para utilizar las funcionalidades dependientes del sistema operativo. Permitió acceder a las rutas de los directorios donde se guardaron las imágenes y crear nuevas carpetas para guardar las etiquetas generadas.

3.2.5. Recursos de Hardware

Para llevar a cabo el proceso de entrenamiento de la red neuronal previo a la implementación del sistema se utilizó una GPU, ya que permite disminuir considerablemente el tiempo de entrenamiento, pudiendo obtener resultados en minutos en lugar de horas o días en lugar de meses.

Todo el trabajo se realizó en una computadora personal portátil, cuyas características se encuentran resumidas en la Tabla 3.4.

Procesador	Intel Core i5-12500H 2.5 GHz
Memoria RAM	16 GB
GPU	NVIDIA GeForce RTX 3060 3584 núcleos - 8 GB memoria
Sistema Operativo	Windows 11 Home SL

Tabla 3.4: Características de la computadora utilizada

Capítulo 4

Resultados y Discusiones

En este capítulo se presentan todos los resultados relacionados con las experimentaciones realizadas en este trabajo. En las Secciones 4.1 y 4.2, se examinan los efectos de las diferentes estrategias propuestas para el entrenamiento y la validación de la red neuronal YOLOv8 en la detección de cáncer de colon sobre los conjuntos de datos CVC-ClinicDB y Kvasir-SEG, respectivamente. Luego, en la Sección 4.3 se realiza una comparación de las mejores configuraciones obtenidas del modelo propuesto con otros resultados del estado del arte. Además, se presentan los resultados de combinar y evaluar ambos conjuntos de datos en la Sección 4.4. Finalmente, en la Sección 4.5 se describe el proceso realizado para el desarrollo del prototipo de la herramienta de visualización.

4.1. Resultados sobre el conjunto de datos CVC-ClinicDB

En esta sección, se examinan los resultados del proceso de entrenamiento y validación de la red neuronal YOLOv8 sobre el conjunto de datos CVC-ClinicDB. Para ello, se utilizan las diferentes configuraciones de hiperparámetros que fueron definidos en la Tabla 3.3.

La Tabla 4.1 muestra los resultados de entrenar y validar la red desde cero (*from scratch*), es decir, sin la utilización de pesos obtenidos de otros entrenamientos previos. Mientras que la Tabla 4.2 muestra los resultados utilizando la técnica de aprendizaje por transferencia (*transfer learning*). Para ambas tablas, las columnas 1 a 4 representan

las diferentes configuraciones de hiperparámetros. La columna 1 indica el número de épocas (*epochs*) utilizado (100 o 200); la columna 2 indica el tamaño del lote (*batch*), cuyo valor es de 16 o 32; en la columna 3 se detalla el optimizador (*opt.*), que puede ser SGD o Adam; y en la columna 4 se indica la tasa de aprendizaje o *learning rate* (*LR*), que varía entre 0.001 o 0.0001. Por lo tanto, se presentan 8 configuraciones diferentes para un valor de 100 *epochs* y otras 8 configuraciones para el valor de 200 *epochs*. Luego, en las columnas restantes se muestran los valores de las métricas descritas en la Sección 2.4. La columna 5 presenta los resultados de la precisión (*prec.*), la columna 6 de la sensibilidad (*recall*), la columna 7 del valor F1 (*F1*) y la columna 8 de la precisión media (*mAP50*). Por último, en la columna 9 se detalla el tiempo total de ejecución del entrenamiento de la red (*time*) expresado en minutos. Los resultados más favorables de cada métrica se encuentran resaltados en negrita para los distintos valores de *epochs*.

Tabla 4.1: Resultados de la red entrenada desde cero sobre el conjunto de datos CVC-ClinicDB.

Conf.	Epochs	Batch	Opt.	LR	Prec.	Recall	F1	mAP50	Time
C1	100	16	SGD	0.001	0.820	0.741	0.779	0.820	4.92
C2				0.0001	0.013	0.375	0.025	0.019	4.80
C3			Adam	0.001	**0.987**	**0.961**	**0.974**	**0.983**	4.92
C4				0.0001	0.984	0.758	0.856	0.846	4.86
C5		32	SGD	0.001	0.785	0.738	0.761	0.788	4.08
C6				0.0001	0.002	0.688	0.005	0.029	4.02
C7			Adam	0.001	0.966	0.912	0.938	0.975	4.14
C8				0.0001	0.943	0.738	0.828	0.839	4.08
C9	200	16	SGD	0.001	0.951	0.825	0.884	0.893	10.38
C10				0.0001	0.211	0.200	0.205	0.112	9.54
C11			Adam	0.001	0.985	**0.963**	**0.974**	0.984	9.78
C12				0.0001	0.969	0.863	0.913	0.894	9.90
C13		32	SGD	0.001	0.971	0.843	0.902	0.912	8.16
C14				0.0001	0.214	0.263	0.236	0.133	7.86
C15			Adam	0.001	0.994	0.925	0.958	**0.985**	8.04
C16				0.0001	**1.000**	0.825	0.904	0.895	8.04

Considerando las primeras 8 configuraciones de la Tabla 4.1, la configuración 3 con un *batch size* de 16, el *opt.* Adam y un *learning rate* de 0.001 es la más exitosa en comparación a las demás, debido a que presenta los valores más altos para cada métrica evaluada. En segundo lugar, se encuentra la configuración 7, obteniendo valores muy cercanos a la configuración anterior, pero con un *batch size* de 32, mientras que en las otras 8 configuraciones para un valor de 200 *epochs* también se presentan resultados similares. En general, se puede observar que en todos los casos se obtienen mejores resultados utilizando un *LR* de 0.001 sobre 0.0001, esto puede ser debido a que un *LR* más

Tabla 4.2: Resultados de la red pre entrenada con el conjunto de datos COCO sobre el conjunto de datos CVC-ClinicDB.

Conf.	Epochs	Batch	Opt.	LR	Prec.	Recall	F1	mAP50	Time
C1	100	16	SGD	0.001	**1.000**	0.938	0.968	0.982	6.24
C2				0.0001	0.959	0.872	0.913	0.928	8.04
C3			Adam	0.001	0.985	0.963	0.974	0.985	5.10
C4				0.0001	0.985	0.950	0.967	0.981	5.40
C5		32	SGD	0.001	0.985	0.950	0.967	0.979	4.32
C6				0.0001	0.998	0.838	0.911	0.924	4.26
C7			Adam	0.001	0.986	**0.988**	**0.987**	**0.989**	4.26
C8				0.0001	0.987	0.949	0.968	0.980	4.20
C9	200	16	SGD	0.001	**1.000**	0.948	0.973	0.983	5.94
C10				0.0001	0.995	0.887	0.938	0.963	10.08
C11			Adam	0.001	0.951	0.975	0.963	0.984	6.48
C12				0.0001	0.963	0.974	0.968	0.984	9.96
C13		32	SGD	0.001	**1.000**	0.961	0.980	0.983	6.60
C14				0.0001	0.978	0.912	0.944	0.964	8.16
C15			Adam	0.001	0.975	**0.987**	**0.981**	**0.991**	8.16
C16				0.0001	0.975	0.972	0.973	0.987	6.54

bajo demora más en aprender, por lo que necesitaría que la red se entrene durante una mayor cantidad de *epochs*. Por otro lado, se observa que el *opt.* Adam resulta mejor que SGD, esto se debe a la capacidad de Adam para adaptar automáticamente el *LR* y su uso de momentos y promedios móviles, mientras que SGD puede ser un poco más inestable y tardar más tiempo en converger a una solución óptima. Asimismo, Adam es un algoritmo más avanzado que utiliza una tasa de aprendizaje adaptativa para actualizar los pesos. La tasa de aprendizaje se ajusta automáticamente durante el entrenamiento, lo que permite una convergencia más rápida y estable. Además, Adam utiliza un momento adaptativo para actualizar los pesos, lo que ayuda a evitar mínimos locales. En este caso, para la detección de imagenes, Adam presenta un mejor comportamiento dado los resultados obtenidos. En cuanto al *batch size*, no se encuentran diferencias significativas entre los valores de 16 y 32. Aquellas configuraciones que obtuvieron un rendimiento inferior con respecto a las demás configuraciones son las que utilizan el *opt.* SGD con un *LR* de 0.0001, que corresponden a las configuraciones 2 y 6 de esta tabla.

Por otro lado, en la Tabla 4.2, que muestra los resultados de aplicar *transfer learning*, la configuración que más sobresale para ambos valores de *epochs* es la número 7, obteniendo los valores más altos de *recall*, *F1* y *mAP* y valores cercanos al óptimo en *precision* (100 % de precisión). Además, se puede observar cómo la utilización de esta técnica aumenta rotundamente los valores en las configuraciones 2 y 6, que eran

significativamente inferiores a las demás en la otra tabla. Esto se debe al conocimiento previo obtenido a partir del uso de la transferencia de aprendizaje, dado que permite aprovechar el trabajo ya realizado en modelos pre entrenados y esto acelera significativamente el proceso de ajuste y mejora en la detección.

En cuanto al tiempo de ejecución, para una misma cantidad de *epochs* los valores son muy similares entre sí. Sin embargo, se puede destacar que al duplicar el número de *epochs*, el tiempo de ejecución aumenta, llegando a ser el doble en la mayoría de los casos con respecto a 100 *epochs*.

Si bien en ambas tablas se obtienen buenos resultados, las configuraciones que utilizan la técnica de *transfer learning* superan a las que se entrenan desde cero (*from scratch*), ya que adquieren un conocimiento previo en tareas relacionadas que les permite mejorar su rendimiento.

En base a estos resultados, se seleccionó la configuración 7 con 100 *epochs* de la Tabla 4.2 para demostrar el poder de detección de esta red neuronal, debido a que presenta valores muy altos para cada métrica y su tiempo de entrenamiento es menor. En la Figura 4.1 se presenta una matriz de 4×4 con diferentes imágenes de pólipos del conjunto de datos CVC-ClinicDB. En cada imagen se marca la detección realizada por la red con un rectángulo rojo junto al valor de confianza, que indica la certeza de la detección en una escala de 0 a 1. En la figura se observa que en todos los casos se detecta el pólipo con una confianza de entre 0.9 (90 %) y 1 (100 %).

4.2. Resultados sobre el conjunto de datos Kvasir-SEG

En esta sección, se examinan los resultados de aplicar las mismas configuraciones definidas en la sección anterior sobre el conjunto de datos Kvasir-SEG.

Primero se analizan los resultados de entrenar y validar la red desde cero (*from scratch*) en la Tabla 4.3, y posteriormente utilizando la técnica de *transfer learning* en la Tabla 4.4.

En la Tabla 4.3, se observa que las configuraciones 3 y 7 son las que obtienen mejores resultados y las configuraciones 2 y 6 resultan notablemente inferiores, como ocurría en el otro conjunto de datos. Además, en términos generales, el *optimizer* Adam obtiene mejores resultados que SGD y el *LR* de 0.001 supera al valor de 0.0001, mientras que los distintos tamaños de *batch* muestran resultados similares. Como se explicó anteriormente, Adam presenta mejoras sobre SGD debido a su capacidad para adaptar automáticamente el *LR* y el uso de un *LR* más chico implica necesitar mayor

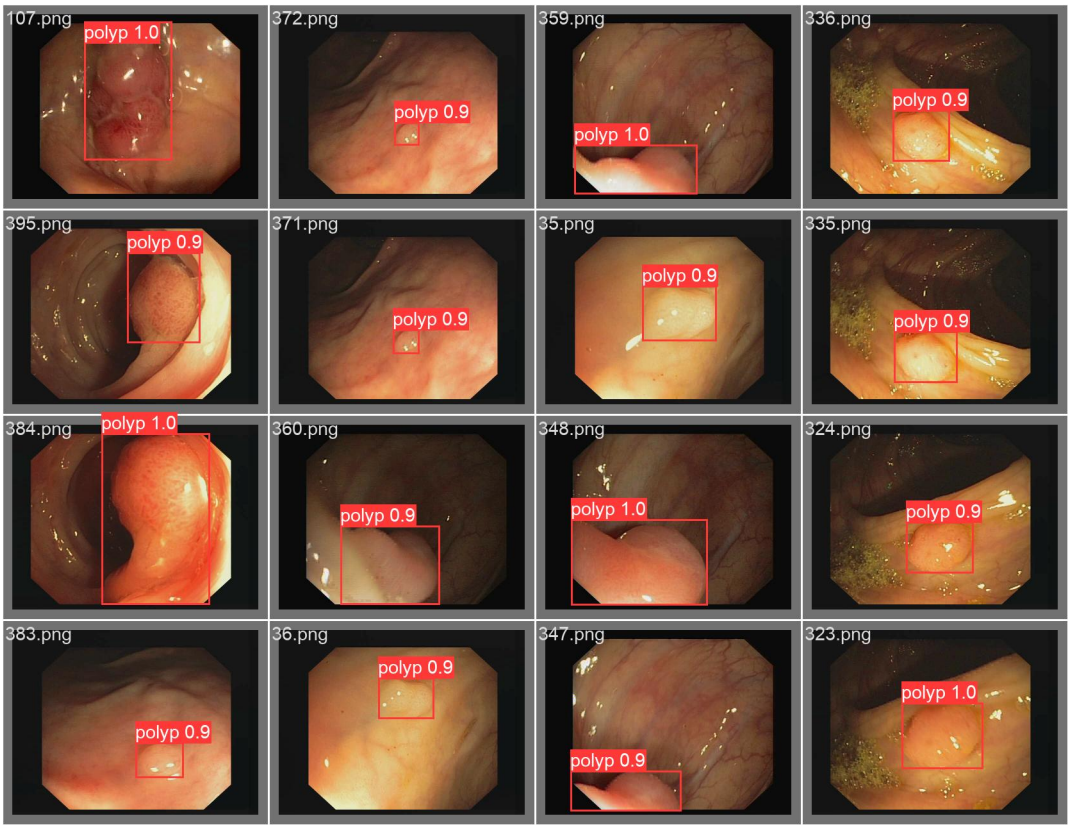

Figura 4.1: Ejemplo de resultados de detección sobre el conjunto de datos CVC-ClinicDB

cantidad de *epochs* para la convergencia de la red. Por otro lado, las configuraciones con un *epoch* de 200 mejoran el rendimiento de aquellas que utilizan 100 debido a que la red continúa ajustando los pesos del modelo. Además, al ser un conjunto con más imágenes, al tener más epochs para poder entrenar el modelo, se mejoran las métricas de detección. Sin embargo, el tiempo de ejecución para el entrenamiento de dichas configuraciones es de aproximadamente el doble en todos los casos.

Por otro lado, en la Tabla 4.4, donde se aplica *transfer learning*, los valores obtenidos para cada métrica en las distintas configuraciones son muy similares entre sí. La *precision* varía entre 0.849 y 0.95, siendo la diferencia más amplia con un valor del 10 %. Luego, el *recall* oscila entre 0.698 y 0.749, habiendo un 5 % de diferencia entre el valor más bajo y el más alto. Por último, el *F1* y el *mAP* varían entre 0.779 y 0.825 y entre 0.755 y 0.814 respectivamente, obteniendo también una diferencia aproximada del 5 %. Si bien los valores son cercanos entre ellos, considerando todas las métricas, se puede observar que se sigue destacando la configuración 3 con 100 *epochs*, ya que obtiene los valores más altos de *precision* y *mAP*, y valores muy buenos cercanos a los mejores en las otras métricas. En base a estos resultados, se decidió utilizar esta última configuración para visualizar ejemplos del comportamiento de la red.

Tabla 4.3: Resultados de la red entrenada desde cero sobre el conjunto de datos Kvasir-SEG.

Conf.	Epochs	Batch	Opt.	LR	Prec.	Recall	F1	mAP50	Time
C1	100	16	SGD	0.001	0.721	0.614	0.663	0.664	12.72
C2				0.0001	0.122	0.039	0.059	0.046	12.72
C3			Adam	0.001	0.823	**0.731**	**0.774**	0.769	12.72
C4				0.0001	0.699	0.668	0.683	0.683	12.66
C5		32	SGD	0.001	0.696	0.657	0.676	0.687	14.70
C6				0.0001	0.114	0.070	0.087	0.047	12.30
C7			Adam	0.001	**0.843**	0.686	0.756	**0.773**	13.86
C8				0.0001	0.746	0.596	0.663	0.677	13.38
C9	200	16	SGD	0.001	0.849	0.698	0.766	0.754	31.02
C10				0.0001	0.206	0.181	0.193	0.111	30.36
C11			Adam	0.001	**0.897**	0.725	0.802	0.795	25.68
C12				0.0001	0.785	0.719	0.751	0.744	25.56
C13		32	SGD	0.001	0.888	0.673	0.766	0.749	24.36
C14				0.0001	0.276	0.288	0.282	0.136	24.42
C15			Adam	0.001	0.866	**0.763**	**0.811**	**0.816**	24.66
C16				0.0001	0.858	0.686	0.762	0.739	24.54

Tabla 4.4: Resultados de la red pre entrenada con el conjunto de datos COCO sobre el conjunto de datos Kvasir-SEG.

Conf.	Epochs	Batch	Opt.	LR	Prec.	Recall	F1	mAP50	Time
C1	100	16	SGD	0.001	0.925	0.723	0.812	0.781	12.72
C2				0.0001	0.888	0.699	0.782	0.758	12.84
C3			Adam	0.001	**0.950**	0.712	0.814	**0.807**	12.78
C4				0.0001	0.895	0.725	0.801	0.803	15.84
C5		32	SGD	0.001	0.891	**0.749**	0.814	0.789	14.46
C6				0.0001	0.860	0.712	0.779	0.755	14.64
C7			Adam	0.001	0.882	0.735	0.802	0.802	14.76
C8				0.0001	0.936	0.725	**0.817**	0.801	12.42
C9	200	16	SGD	0.001	0.923	**0.745**	**0.825**	0.769	28.08
C10				0.0001	0.849	0.739	0.790	0.770	28.32
C11			Adam	0.001	0.922	0.719	0.808	**0.814**	25.56
C12				0.0001	0.903	0.730	0.807	0.810	22.26
C13		32	SGD	0.001	0.882	0.739	0.804	0.785	12.42
C14				0.0001	0.914	0.698	0.792	0.763	24.42
C15			Adam	0.001	0.903	0.727	0.805	0.811	24.54
C16				0.0001	**0.929**	0.739	0.823	**0.814**	18.30

En la Figura 4.2, se muestra la capacidad de detección de esta red neuronal presentando una matriz de 4 × 4 con diferentes imágenes de pólipos del conjunto de datos Kvasir-SEG. Las detecciones realizadas por la red se marcan con un rectángulo rojo

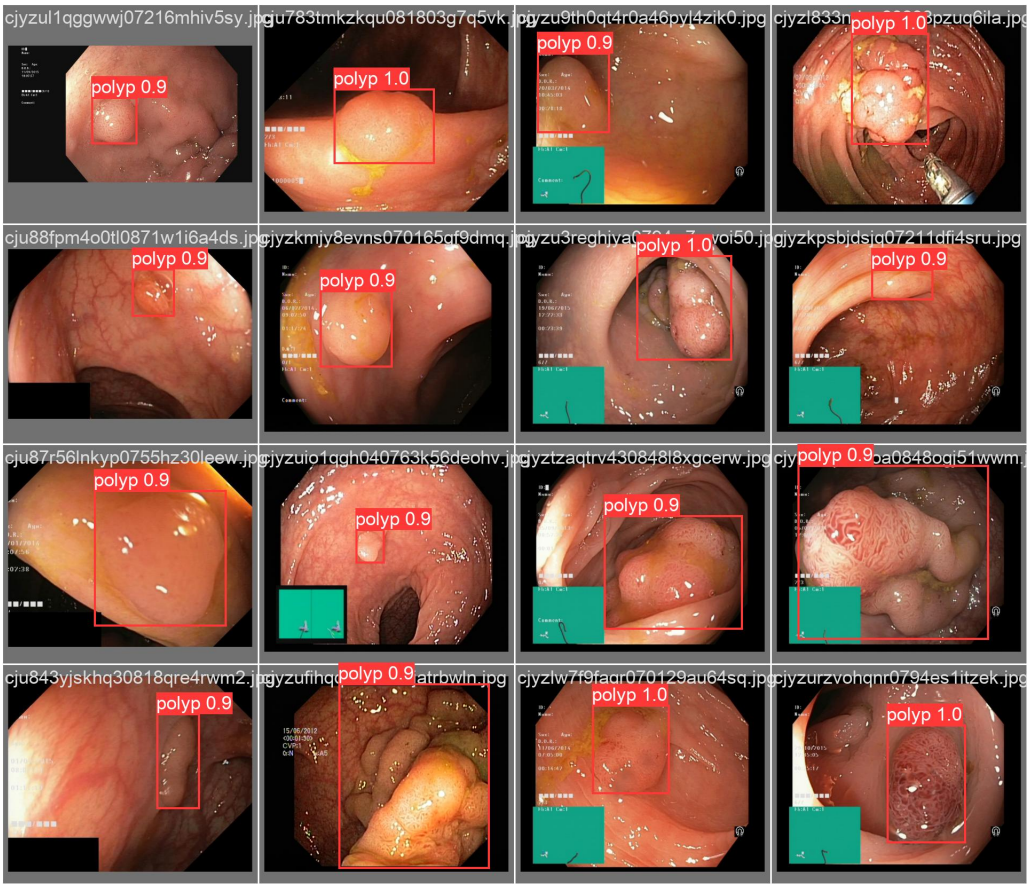

Figura 4.2: Ejemplo de resultados de detección sobre el conjunto de datos Kvasir-SEG

junto a los valores de confianza que indican la certeza de la detección en una escala de 0 a 1. Se observa que en todos los casos se detecta el pólipo con una confianza de entre 0.9 (90 %) y 1 (100 %).

4.3. Comparación con el estado del arte

En esta sección, se evalúa el rendimiento de algunos de los modelos entrenados en la sección anterior en un nuevo conjunto de datos (CVC-ColonDB) con el fin de comparar la YOLOv8 con los otros modelos del estado del arte. Para ello, se eligieron las configuraciones 3 y 7 con 100 *epochs* de la Tabla 4.2 que son las que obtuvieron los mejores resultados de entrenamiento y validación. A fin de realizar una comparación más justa, se utilizaron los modelos entrenados únicamente en el conjunto de datos CVC-ClinicDB, ya que los otros autores utilizaron ese conjunto de imágenes para entrenar la red y luego evaluarla en CVC-ColonDB. Por lo tanto, la red entrenada en el conjunto de datos Kvasir-SEG no se utiliza en esta comparación.

Como se explicó en la Sección 3.1.2, el conjunto de imágenes CVC-ColonDB no

estaba disponible en el enlace original, por lo que se utilizó otro enlace para accederlo que traía mayor cantidad de imágenes (380 imágenes). En consecuencia, se realizó un proceso de reducción de forma aleatoria hasta llegar a las 300 imágenes para coincidir con los demás trabajos ya que esa es la cantidad de datos reportada en el estado del arte. Dado que no existe información fehaciente sobre el motivo de esos cambios, los valores obtenidos pueden estar influenciados directamente por las imágenes que han quedado y las que fueron eliminadas.

En la siguiente tabla, se presenta la comparación de los modelos propuestos con aquellos utilizados en el estado del arte que fueron entrenados con el conjunto de datos CVC-ClinicDB y evaluados sobre el conjunto de datos CVC-ColonDB. Para comparar, se utilizan las métricas previamente explicadas (*percision*, *recall* y *F1*) y se agrega la velocidad (*speed*), que indica el tiempo de inferencia de la red en *ms*.

Tabla 4.5: Evaluación de métodos propuestos con otros modelos utilizados sobre el conjunto de datos CVC-ColonDB (IOU=0.5, batch size=1, confidence threshold=0.25)

Autor	Modelo	Prec.	Recall	F1	Speed
Sornapudi et al., 2019 [49]	R−CNN	78.85	95.67	86.58	220
Qadir et al., 2021 [38]	MDeNetplus (F−CNN)	88.35	91.00	89.65	39
Pacal et al., 2021 [33]	YOLOv4-CSP	96.04	96.68	96.36	8.2
Conf. 3 (propia)	YOLOv8 (nano)	91.90	78.70	84.79	7.1
Conf. 7 (propia)	YOLOv8 (nano)	86.60	79.30	82.79	7.2

Como se observa en la Tabla 4.5, en términos generales, la YOLOv8 obtiene resultados competitivos en cuanto a *precision*, superando a la R-CNN y la F-CNN con la Conf. 3 y quedando un 1.75 % por detrás de la F-CNN con la Conf. 7. Sin embargo, ambas configuraciones resultan inferiores en términos de *recall*, lo que implica que la capacidad de detección de TP es más baja. No obstante, ofrecen resultados similares en el valor de *F1*, con una diferencia de 1.79 % entre la Conf. 3 y la R-CNN y una diferencia un poco más grande de 4.86 % entre esa misma configuración y la F-CNN. Esto se debe a que el valor *F1* está influenciado por ambas métricas (tanto *precision* como *recall*). Por otra parte, en términos de velocidad, la YOLOv8 consume mucho menos tiempo de cómputo que los modelos R-CNN y F-CNN, siendo 96.77 % y 81.79 % más rápida respectivamente. Luego, si se compara con la YOLOv4, en general se obtienen resultados inferiores con ambas configuraciones. No obstante, la YOLOv4 utilizada por Pacal es la versión *small*. Cabe destacar que los valores que se obtuvieron de la red YOLOv8 fueron utilizando la versión *nano*, que es la más pequeña. Es muy probable que si se prueba con una versión de la red más grande, con mayor cantidad de parámetros y capas, los resultados que se obtengan tales como *precision* y *recall* sean mejores. Dado que el objetivo era

evaluar la capacidad de la red YOLOv8 en su versión *nano*, los resultados obtenidos en las evaluaciones experimentales permiten indicar que aumentar la capacidad de la YOLOv8 a una versión *small* o *medium* aumentaría la capacidad de detección y podría aumentar el valor de la métrica *precision*. Obviamente, esto también afectaría el tiempo de ejecución y seguramente sería más grande. Sin embargo, a los efectos de este trabajo, se decidió utilizar esta versión por las razones explicadas anteriormente. Aún así, los resultados que se observan resultan prometedores.

4.4. Resultados de combinar los conjuntos de datos

Además de los experimentos realizados sobre cada conjunto de datos por separado, se evaluó la red neuronal YOLO combinando las imágenes de ambos conjuntos de datos para examinar los efectos del entrenamiento y validación en la detección de pólipos. Para ello, se utilizaron las configuraciones 3 y 7 con 100 *epochs* obtenidas de las Tablas 4.2 y 4.4, que utilizan la técnica de *transfer learning*.

En las Tablas 4.6 y 4.7 la primera columna representa el conjunto de datos de entrenamiento y la primera fila indica el conjunto de datos que se utilizó para la evaluación, dando un total de nueve combinaciones. Para cada combinación, se muestran los resultados de las siguientes métricas: *precision*, *recall* y *F1*. Los mejores valores de cada columna para cada conjunto (entrenamiento y validación) se encuentran marcados en negrita. Para ambas tablas, en la mayoría de los casos, se obtuvieron mejores valores al evaluar los diferentes conjuntos de datos de validación con la red neuronal que se entrenó con el conjunto de datos combinado (*mix-training*). Además, se obtienen muy buenos resultados al entrenar y evaluar con el mismo conjunto de datos. Mientras que si se entrena solo con un conjunto de datos y se evalúa en el otro, se obtienen resultados inferiores. Esto tiene sentido ya que las imágenes que se están evaluando son diferentes a las que la red neuronal aprendió a reconocer.

Por otra parte, también se puede destacar que se obtienen mejores resultados al evaluar la red entrenada en Kvasir-SEG sobre el conjunto de datos CVC-ClinicDB que de la forma contraria. Esto puede suceder porque Kvasir-SEG tiene una mayor cantidad de imágenes y más variadas, mientras que las de CVC-ClinicDB son más simples, por lo que a la red le cuesta más aprender diferentes características.

Tabla 4.6: Resultados de entrenar y evaluar la red en los diferentes conjuntos de datos con la configuración 3

Entrenamiento	Validación								
	CVC-ClinicDB			Kvasir-SEG			Mix-val		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
CVC-ClinicDB	0.985	**0.963**	0.974	0.746	0.594	0.661	0.853	0.712	0.776
Kvasir-SEG	0.942	0.825	0.880	**0.953**	0.712	**0.815**	0.878	0.774	0.823
Mix-training	**0.998**	**0.963**	**0.980**	0.904	**0.739**	0.813	**0.945**	**0.817**	**0.876**

Tabla 4.7: Resultados de entrenar y evaluar la red en los diferentes conjuntos de datos con la configuración 7

Entrenamiento	Validación								
	CVC-ClinicDB			Kvasir-SEG			Mix-val		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
CVC-ClinicDB	0.987	**0.988**	**0.987**	0.719	0.621	0.666	0.848	0.725	0.782
Kvasir-SEG	0.920	0.860	0.889	**0.880**	0.732	0.799	0.895	0.764	0.824
Mix-training	**0.999**	0.975	0.982	0.870	**0.745**	**0.803**	**0.921**	**0.820**	**0.868**

4.5. Resultados del desarrollo de la herramienta de visualización

Esta sección proporciona una descripción detallada de cómo se aplicó la metodología XP en cada etapa del desarrollo del primer prototipo del sistema y el resultado obtenido, siguiendo los pasos descritos en la Sección 3.1.6.

Etapa de Planificación: Esta etapa se centró en establecer los objetivos del sistema, identificando los requisitos clave y priorizando las características a implementar. En base a esto, se definió la historia de usuario que se muestra en la Figura 4.3.

Visualización de detección
Se quiere visualizar la detección producida por la red neuronal sobre una imagen seleccionada por el usuario. Para ello, el sistema permitirá cargar una imagen desde el ordenador y contendrá una opción para procesar dicha imagen. Al procesar la imagen, se redirigirá a una nueva página que mostrará la imagen resultante con la detección producida por la red neuronal.

Figura 4.3: Historia de usuario

Etapa de Diseño: Esta etapa se enfocó en la creación de un diseño claro y efectivo para las características y funcionalidades identificadas en la etapa de planificación. Como resultado de esta etapa se obtuvo un prototipo de la interfaz de usuario que se muestra en la Figura 4.4.

Figura 4.4: Prototipo

Etapa de Codificación: Esta etapa se basó en la implementación de las funcionalidades requeridas por el sistema descritas en la historia de usuario.

En primer lugar, se programó en PHP las páginas de inicio (Anexo A.2.2.1) y resultado (Anexo A.2.2.3) basándose en el prototipo producido en la etapa anterior, obteniendo las funcionalidades básicas que permiten seleccionar una imagen del ordenador y un botón que servirá para procesarla. Este código se complementó con código HTML para producir las interfaces de usuario del *front-end*.

Y, en segundo lugar, se programó un *script* en Python (Anexo A.2.2.2) que permitiera realizar el procesamiento de la imagen con la red neuronal. Dicho procesamiento forma parte exclusivamente del *back-end*.

El *script* recibe la imagen de entrada y carga el modelo elegido para la red YOLO desde una carpeta denominada "neural_network_model", para luego realizar la predicción y obtener la imagen resultante. En este trabajo se optó por cargar el modelo con los pesos que se obtuvieron del entrenamiento de la red con la combinación de ambos conjuntos de datos (*mix-training*) y la configuración 3 de la Tabla 4.6, ya que se desea que la red del sistema pueda reconocer la mayor

variedad de imágenes posible. Esta implementación permite que en un futuro, si la red se entrena con más imágenes y obtiene mejores resultados, se puedan cargar los pesos de ese modelo para utilizarlos en el sistema de detección.

Etapa de Pruebas: Esta última etapa se centró en garantizar el correcto funcionamiento del sistema. Las pruebas se centraron exclusivamente en evaluar la funcionalidad de detección de imágenes, dado que era el objetivo primordial del prototipo. Para ello, se cargaron imágenes de los diferentes conjuntos de datos utilizados en este trabajo y se evaluaron los resultados producidos. En la Figura 4.5 se puede observar un ejemplo donde se carga una imagen de CVC-ClinicDB y se muestra el resultado de la detección producido por la red neuronal.

Figura 4.5: Ejemplo de Detección con la Herramienta de Visualización

Cabe destacar que las etapas mencionadas anteriormente forman parte de la primera iteración del ciclo. Luego se llevó a cabo una segunda iteración con el fin de perfeccionar el prototipo inicial, agregando la posibilidad de visualizar la imagen una vez seleccionada y añadiendo color a la página para que el diseño sea más amigable al usuario. En la Figura 4.6 se puede observar esta segunda versión del prototipo inicial de la herramienta de visualización.

Para concluir esta sección, se destaca que el uso de la metodología ágil (XP) fue muy útil, ya que permitió cumplir con los objetivos del trabajo, logrando desarrollar un prototipo de un sistema para la detección de cáncer de colon en imágenes médicas. Además, facilitó la obtención de resultados rápidos y tangibles a corto plazo.

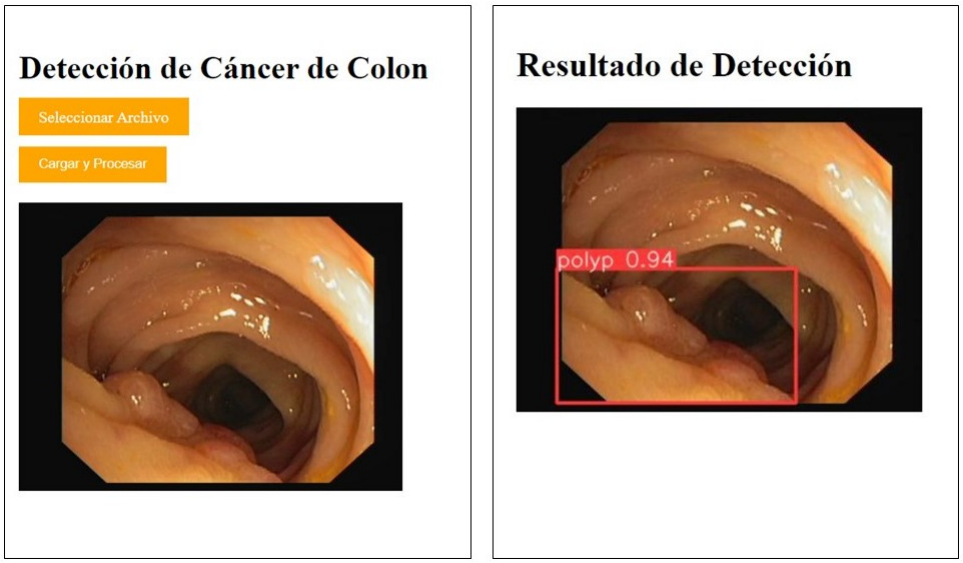

Figura 4.6: Herramienta de Visualización - Versión 2

4.5.1. Prueba del sistema con datos de Mendoza

Como se mencionó anteriormente, los conjuntos de datos utilizados en este trabajo fueron accedidos a través de enlaces públicos. Uno de los grandes desafíos del uso de herramientas de AI en la medicina es su aplicación en el ámbito real. Es por ello que se buscó evaluar el sistema desarrollado en imágenes no académicas, obtenidas de pacientes de Mendoza. Para esto, se solicitaron imágenes en diferentes instituciones médicas tales como Fuesmen, Hospital Universitario y Clínica de Cuyo. En algunos casos no realizaban colonoscopias por lo cual no tenían imágenes que sirvieran al estudio. La Dra. Olivera como directora del grupo CICDa [1] del ICB [2] al que pertenecemos tanto mi tutor, el Dr. Vidal, como yo en calidad de estudiante, logró contactarse con el Dr. Marcelo Di Cicco, especialista en Cirugía Endoscópica (MAAC), de la Unidad Quirúrgica del Hospital Universitario de Mendoza/UNCuyo. El Dr. De Cicco brindó amablemente un conjunto de imágenes (12 imágenes) con pólipos de colonoscopias realizadas por él para que puedan ser evaluadas por el prototipo desarrollado. Estas imágenes corresponden a colonoscopias realizadas sobre diferentes pacientes que fueron obtenidas en distintos años con, no necesariamente, la misma aparatología.

En la Figura 4.7, se muestra la capacidad de detección del sistema implementado en 12 imágenes, presentando una matriz de 4×3. Las detecciones realizadas por la red se marcan con un rectángulo rojo junto a los valores de confianza que indican la certeza de la detección en una escala de 0 a 1. Se observa que en la mayoría de casos

[1] Grupo de Investigación en Ciencias de la Computación y de los Datos
[2] Instituto Interdisciplinario de Ciencias Básicas, ICB CONICET UNCuyo

se detecta el pólipo con una confianza de entre 0.9 (90%) y 1 (100%). En dos casos particulares (fila 2, columna 1) y (fila 4, columna 2), el pólipo es detectado, aún cuando la confianza ronda alrededor del 0.38 y 0.45 respectivamente. Esto puede deberse a la naturaleza del pólipo y su ubicación dentro de la imagen. En ambos casos, existe similitud entre el pólipo y la zona circundante al mismo, teniendo en cuenta el color de toda la zona y su ubicación dentro del intestino a la hora de obtener la imagen. Esto hace que a primera vista pueda pasar desapercibido en el video, pero igualmente es detectado por el sistema desarrollado.

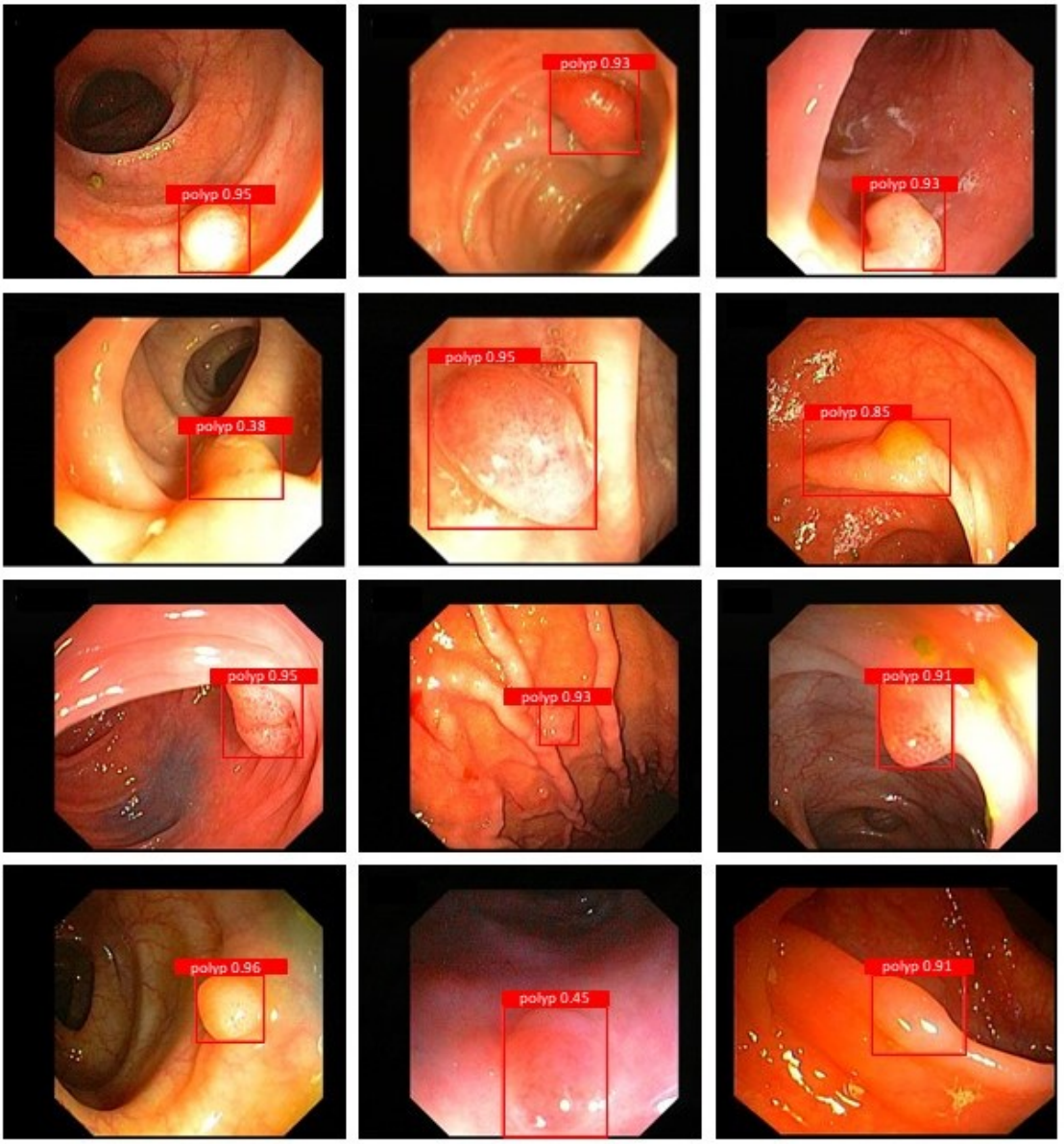

Figura 4.7: Imágenes de colonoscopias con la región identificada del posible pólipo a partir de la detección del sistema desarrollado. Imágenes cortesía del Dr. Di Cicco (Pacientes de la Prov. de Mendoza).

Capítulo 5

Conclusiones, desafíos y perspectivas futuras

Este capítulo contiene las conclusiones principales del presente Trabajo de Fin de Grado. Se resumen los objetivos cumplidos (Sección 5.1), los problemas y desafíos encontrados (Sección 5.2) y los posibles trabajos futuros (Sección 5.3).

5.1. Conclusiones

Los sistemas de detección asistidos por computadora ayudan a los médicos en la detección temprana de pólipos y cáncer de colon, lo que a su vez puede mejorar los tratamientos y sus resultados, llegando incluso a salvar vidas. No obstante, a medida que la tecnología continúa avanzando, se espera que se desarrollen enfoques aún más precisos y eficientes. Las herramientas de AI basadas en redes neuronales profundas han demostrado una alta precisión y eficiencia en la detección de pólipos para cáncer de colon a partir de imágenes médicas.

En relación a los objetivos planteados al principio de este documento podemos concluir:

- En lo referente a *comprender el funcionamiento de las técnicas comúnmente aplicadas a la detección de cáncer utilizando imágenes médicas, en particular, aquellas que corresponden a DL*, se ha podido examinar de manera exhaustiva el estado del arte buscando, por un lado, todo lo relacionado con el análisis de imágenes médicas de colonoscopias y, por otro lado, las diferentes arquitecturas de redes neuronales profundas utilizadas para el campo de detección de imágenes. El estudio del estado del arte ha permitido analizar y comprender cómo funcionan

las diferentes arquitecturas modificando o agregando módulos en pos de tener una mejor detección de imágenes.

- En lo que respecta a *diseñar y trabajar sobre un pipeline que permita evaluar el comportamiento de técnicas de AI para la detección del cáncer de colon*, se desarrolló una metodología general de trabajo que logró definir metas a través de diferentes etapas del trabajo final. La construcción del *pipeline* estuvo determinada por las tareas primordiales necesarias para cumplir los objetivos de este trabajo. Cada una de estas tareas tuvo una meta específica. Al principio, a partir del estudio del estado del arte, se seleccionó el algoritmo YOLO en su última versión (8) como herramienta de detección de pólipos en imágenes de colonoscopias. Luego, en la tarea de adquisición de datos, se seleccionaron diferentes conjuntos de datos conteniendo imágenes de colonoscopias (CVC-ClinicDB, CVC-ColonDB y Kvasir-SEG). Posteriormente, se realizaron diferentes experimentos con el fin de evaluar la YOLOv8. En esta tarea se trabajó sobre el estudio de hiperparámetros, se experimentó entrenando, validando y probando la capacidad de detección de la red YOLOv8 con diferentes formas de los conjuntos de datos, cada uno por separado, haciendo una mezcla de los mismos y comparándolo con el estado del arte. Se analizaron los diferentes resultados obtenidos a través de las métricas que son mayormente utilizadas para las tareas de detección. Se obtuvieron excelentes resultados, pudiendo clasificar con una precisión desde un 85 % y alcanzando el 100 % para algunas imágenes. A partir de todos los resultados obtenidos, se seleccionó una configuración de hiperparámetros, que demostró tener mayor robustez y un rendimiento más eficiente que el resto de las configuraciones evaluadas. La definición del *pipeline* como base para concretar tareas y definir tiempos fue necesaria y de gran ayuda para establecer períodos de trabajo y establecer subobjetivos a cumplir durante todo el proceso del trabajo final.

- Finalmente, en relación con la *implementación de un prototipo que permita evaluar diferentes imágenes y visualizar la detección del algoritmo seleccionado, ayudando a los médicos a detectar rápidamente los pólipos que pueden generar el cáncer de colon*, se ha podido desarrollar de manera satisfactoria un primer prototipo que permite trabajar de manera sencilla con la detección de pólipos de cáncer de colon. El uso de la metodología XP ha permitido enfocarse en las necesidades básicas del prototipo y avanzar de manera sistemática hasta conseguir un sistema funcional. Se ha tomado la mejor configuración encontrada en los experimentos como base para la red YOLOv8. Se construyó un sistema prototipo que permite

visualizar el resultado de la detección producido por la red neuronal, sirviendo como un sistema de apoyo al profesional médico.

Para finalizar, y desde un punto de vista más personal, gracias a la elección de este proyecto he conseguido aprender más en profundidad sobre un tema apasionante como el uso de las redes neuronales profundas para la detección del cáncer, lo cual considero que será de gran utilidad e importancia en un futuro próximo. Además, he podido aplicar el prototipo desarrollado en imágenes de pacientes de la ciudad de Mendoza, gracias a la predisposición del Dr. De Cicco para brindarlas y a la gestión de la Dra. Olivera para conseguirlas. Esto demuestra que el prototipo no solo sirve para imágenes de conjuntos de datos académicos, sino también para imágenes que utilizan a diario los especialistas para el diagnóstico.

5.2. Dificultades encontradas

A lo largo de este trabajo, se identificaron varias dificultades particulares que de alguna forma son cruciales para abordar en el campo de la detección de enfermedades utilizando conjuntos de datos de imágenes médicas y redes neuronales. Algunos de los problemas encontrados incluyen:

- Disponibilidad limitada de datos: Uno de los desafíos fundamentales encontrados fue la disponibilidad limitada de conjuntos de datos de imágenes médicas de alta calidad y en cantidad suficiente. Se buscó de manera incansable durante el desarrollo de este trabajo diferentes conjuntos de datos que pudieran aportar mayor claridad en los resultados, no obstante, solo se pudo trabajar con los conjuntos detallados en el documento, dado que eran los que se brindaban de acceso abierto. Se quiso utilizar el conjunto de datos ETIS-Larib mencionado en los antecedentes para realizar la prueba del algoritmo seleccionado y poder compararlo con mayor cantidad de modelos, sin embargo ya no se encontraba disponible en la página y a pesar de contactarse con los autores no fue posible conseguirlo. Si bien la recopilación y el etiquetado de datos médicos son procesos costosos y requieren tiempo, creo que también es necesario tener en cuenta este punto para futuros trabajos, dado que conseguir datos de fuentes confiables o que estén disponibles para evaluar de forma más robusta es necesaria. Además, la obtención del consentimiento informado para el uso de datos médicos en investigaciones es un proceso crucial y complicado, lo que limita la posibilidad de contar con grandes muestras de información.

- Requisitos de potencia computacional: El entrenamiento de redes neuronales en conjuntos de datos de imágenes médicas requiere una potencia computacional significativa, lo que resulta costoso y puede ser difícil de conseguir. La necesidad de hardware y recursos informáticos avanzados puede ser un obstáculo para muchas instituciones médicas y proyectos de investigación. La GPU utilizada en este trabajo es una de cómputo normal, de tipo hogareña, lo cual redujo la posibilidad de profundizar en algunas experimentaciones. Se probó la GPU del cluster TOKO perteneciente a la UNCuyo, pero lamentablemente no funcionaba muy bien el nodo que contenía dicha tarjeta gráfica, con lo cual no se pudo avanzar en este camino.

Estos problemas subrayan la importancia de abordar no solo los aspectos técnicos, sino también los éticos, prácticos y computacionales al trabajar con conjuntos de datos de imágenes médicas en la detección de enfermedades.

5.3. Trabajos futuros

En el futuro, es esencial abordar los desafíos identificados en este trabajo para avanzar en la detección de enfermedades utilizando imágenes médicas y redes neuronales.

Como trabajo futuro inicial se podría conseguir mayor disponibilidad a requisitos de potencia computacional, lo cual posibilitaría acelerar algunas pruebas o utilizar otras variaciones del modelo YOLOv8.

Por otro lado, las futuras investigaciones podrían enfocarse en la ampliación de conjuntos de datos diversificados, técnicas de mejora de calidad de imágenes y utilización y comparación de nuevos métodos de ML. Además, sería interesante contar con mayor cantidad de bases de datos regionales.

Por último, sería interesante la integración efectiva de estas herramientas en la práctica clínica y en la validación clínica a gran escala para garantizar su utilidad y seguridad en entornos médicos reales. Estas direcciones de investigación tienen el potencial de mejorar significativamente el diagnóstico y la atención médica en el futuro.

Apéndice A

Anexos

A.1. Multimedia

Como parte del Trabajo Final se grabó un video resumiendo los puntos más importantes del mismo y mostrando el funcionamiento de la herramienta de visualización. El mismo se encuentra subido en YouTube y se puede ver en el siguiente enlace:

https://youtu.be/kx7leP2I3Q0

A.2. Código fuente del proyecto

Todo el código se encuentra disponible y accesible en GitHub a través del siguiente enlace:

https://github.com/marielvol/Tesina-Final-de-Grado/

A.2.1. Generación de cuadros delimitadores *(bounding boxes)*

```
1  import os
2  import numpy as np
3  import cv2
4  from glob import glob
5  from tqdm import tqdm
6  from skimage.measure import label, regionprops, find_contours
7
8  # Create a directory
9  def create_dir(path):
10     if not os.path.exists(path):
11         os.makedirs(path)
```

```
12
13  # Convert a mask to border image
14  def mask_to_border(mask):
15      h, w = mask.shape
16      border = np.zeros((h, w))
17      contours = find_contours(mask, 128)
18      for contour in contours:
19          for c in contour:
20              x = int(c[0])
21              y = int(c[1])
22              border[x][y] = 255
23      return border
24
25  # Convert a mask to bounding boxes
26  def mask_to_bbox(mask):
27      bboxes = []
28      mask = mask_to_border(mask)
29      lbl = label(mask)
30      props = regionprops(lbl)
31      for prop in props:
32          x1 = prop.bbox[1]
33          y1 = prop.bbox[0]
34          x2 = prop.bbox[3]
35          y2 = prop.bbox[2]
36          bboxes.append([x1, y1, x2, y2])
37      return bboxes
38
39  # Transform bounding box to YOLO format
40  def bbox_to_YOLO(bboxes):
41      bboxes_YOLO = []
42      for bbox in bboxes:
43        w = bbox[2] - bbox[0]
44        h = bbox[3] - bbox[1]
45        x = bbox[0] + int(w/2)
46        y = bbox[1] + int(h/2)
47        bboxes_YOLO.append([x, y, w, h])
48      return bboxes_YOLO
49
50  # Normalize bounding boxes
51  def normalize_bboxes(bboxes,mask):
52      h, w = mask.shape
53      for bbox in bboxes:
54        bbox[0] = bbox[0] / w
55        bbox[1] = bbox[1] / h
56        bbox[2] = bbox[2] / w
57        bbox[3] = bbox[3] / h
58      return bboxes
59
60  if __name__ == "__main__":
61      # Load the dataset
62      folder_path = "CVC-ClinicDB" # Indicate dataset path
63      images = sorted(glob(os.path.join(folder_path, "images", "*")))
64      masks = sorted(glob(os.path.join(folder_path, "annotations", "*")))
```

```
65
66      # Create a folder to save the labels
67      create_dir(os.path.join(folder_path, "labels"))
68
69      # Loop over the dataset
70      for x, y in tqdm(zip(images, masks), total=len(images)):
71          # Extract the image name
72          name = x.split("\\")[-1].split(".")[0]
73
74          # Read image and mask
75          x = cv2.imread(x, cv2.IMREAD_COLOR)
76          y = cv2.imread(y, cv2.IMREAD_GRAYSCALE)
77
78          # Detect the bounding boxes
79          bboxes = mask_to_bbox(y)
80
81          # Convert the bounding boxes to YOLO format
82          bboxes_YOLO = bbox_to_YOLO(bboxes)
83
84          # Normalize the bounding boxes
85          bboxes_YOLO_norm = normalize_bboxes(bboxes_YOLO,y)
86
87          # Create a .txt file with the labels
88          file = open(f"labels\\{name}.txt", "w")
89          for bbox in bboxes_YOLO_norm:
90            line = str('0')+' '+str(bbox[0])+' '+str(bbox[1])+''
91                  +str(bbox[2])+' '+str(bbox[3])+'\n'
92            file.write(line)
93          file.close()
```

A.2.2. Prototipo de herramienta de visualización

A.2.2.1. Página de inicio

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>Detección de Cáncer de Colon</title>
5       <style>
6           h1 {
7               text-align: left;
8           }
9
10          form {
11              text-align: left;
12          }
13
14          input[type="file"] {
15              display: none;
16          }
17
```

```
18          .button-container {
19              display: flex;
20              justify-content: space-between;
21              margin-top: 20px;
22              margin-bottom: 20px;
23          }
24
25          .custom-button {
26              background-color: orange;
27              color: white;
28              padding: 10px 20px;
29              border: none;
30              cursor: pointer;
31          }
32
33          .custom-button:hover {
34              background-color: darkorange;
35          }
36
37          #image-preview {
38              max-width: 100%;
39              display: none;
40          }
41
42      </style>
43      <script>
44          function previewImage(input) {
45              var preview = document.getElementById('image-preview');
46              if (input.files && input.files[0]) {
47                  var reader = new FileReader();
48                  reader.onload = function (e) {
49                      preview.src = e.target.result;
50                      preview.style.display = 'block';
51                  }
52                  reader.readAsDataURL(input.files[0]);
53              }
54          }
55      </script>
56  </head>
57  <body>
58      <h1>Detección de Cáncer de Colon</h1>
59      <form action="upload.php" method="post" enctype="multipart/form-data">
60          <label for="uploaded_image" class="custom-button">Seleccionar Archivo</label>
61          <input type="file" name="uploaded_image" id="uploaded_image"
62              style="display:none;" onchange="previewImage(this);">
63          <div class="button-container">
64              <input type="submit" value="Cargar y Procesar" class="custom-button">
65          </div>
66          <img id="image-preview" alt="Vista Previa de la Imagen" src="">
67      </form>
68  </body>
69  </html>
```

A.2.2.2. Procesamiento de la red neuronal

```python
import sys
from PIL import Image
from ultralytics import YOLO

# Gets the path to the image by argument
image_path = sys.argv[1]

# Gets the path to the neural network model
model_path = "neural_network_model\\model.pt"

# Load the neural network model
model = YOLO(model_path)

# Load the image
image = Image.open(image_path)

# Perform the prediction on the image with the model
result = model.predict(source=image)

# Post-process the result
im_array = result[0].plot()

# Get the result
result_image = Image.fromarray(im_array[..., ::-1])

# Save the result
result_image.save("processed_images\\result.jpg")
```

A.2.2.3. Página de resultado

```php
<?php
$target_dir = "images/";
$target_file = $target_dir . basename($_FILES["uploaded_image"]["name"]);
move_uploaded_file($_FILES["uploaded_image"]["tmp_name"], $target_file);
$python_script = "python_scripts/cancer_detection_script.py";
$command = "python3 $python_script $target_file";
$output = shell_exec($command);

$result_image = "processed_images/result.jpg";
?>
<!DOCTYPE html>
<html>
<head>
    <title>Resultado de Detección</title>
</head>
<body>
    <h1>Resultado de Detección</h1>
    <img src="<?php echo $result_image; ?>" alt="Resultado">
</body>
</html>
```

Bibliografía

[1] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018.

[2] Aileen Scott. Ai, ml, or dl – learn what it means, 2022.

[3] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[4] J. Bernal, J. Sánchez, and F. Vilariño. Towards automatic polyp detection with a polyp appearance model. volume 45, pages 3166–3182, 9 2012.

[5] Jorge Bernal, F. Javier Sánchez, Gloria Fernández-Esparrach, Debora Gil, Cristina Rodríguez, and Fernando Vilariño. Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. *Computerized Medical Imaging and Graphics*, 43:99–111, 7 2015.

[6] Xieling Chen, Di Zou, Haoran Xie, Gary Cheng, and Caixia Liu. Two decades of artificial intelligence in education. *Educational Technology & Society*, 25(1):28–47, 2022.

[7] CuidatePlus. Colonoscopia: qué es y cómo se hace, 2021.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] J. Edge. *Kanban: La guía definitiva de la metodología Kanban para el desarrollo de software ágil*. Amazon Digital Services LLC - Kdp, 2018.

[10] Nicolle M Gatto, Harold Frucht, Vijaya Sundararajan, Judith S Jacobson, Victor R Grann, and Alfred I Neugut. Risk of perforation after colonoscopy and sigmoidoscopy: a population-based study. *Journal of the National Cancer Institute*, 95(3):230–236, 2003.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 11 2013.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. International edition. Prentice Hall, 1999.

[14] Instituto Nacional del Cáncer, Argentina. Estadísticas - incidencia. https://www.argentina.gob.ar/salud/instituto-nacional-del-cancer/estadisticas/incidencia.

[15] Instituto Nacional del Cáncer, Argentina. Tipos de cáncer. https://www.argentina.gob.ar/salud/cancer/tipos.

[16] Debesh Jha, Sharib Ali, Nikhil Kumar Tomar, Havard D. Johansen, Dag Johansen, Jens Rittscher, Michael A. Riegler, and Pal Halvorsen. Real-time polyp detection, localization and segmentation in colonoscopy using deep learning. *IEEE Access*, 9:40496–40510, 2021.

[17] Debesh Jha, Pia H Smedsrud, Michael A Riegler, Pål Halvorsen, Thomas de Lange, Dag Johansen, and Håvard D Johansen. Kvasir-seg: A segmented polyp dataset. In *International Conference on Multimedia Modeling*, pages 451–462. Springer, 2020.

[18] Xiao Jia, Xiaochun Mai, Yi Cui, Yixuan Yuan, Xiaohan Xing, Hyunseok Seo, Lei Xing, and Max Q.H. Meng. Automatic polyp recognition in colonoscopy images using deep learning and two-stage pyramidal feature prediction. *IEEE Transactions on Automation Science and Engineering*, 17:1570–1584, 7 2020.

[19] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[20] Laveen Kanal. Perceptron. pages 1383–1385, 01 2003.

[21] KeepCoding. ¿qué es una función de activación en deep learning?, 2022.

[22] Dmitry Koroteev and Zeljko Tekic. Artificial intelligence in oil and gas upstream: Trends, challenges, and scenarios for the future. *Energy and AI*, 3:100041, 2021.

[23] Adrian Krenzer, Michael Banck, Kevin Makowski, Amar Hekalo, Daniel Fitting, Joel Troya, Boban Sudarevic, Wolfgang G. Zoller, Alexander Hann, and Frank Puppe. A real-time polyp-detection system with clinical application in colonoscopy using deep convolutional neural networks. *Journal of Imaging*, 9, 2 2023.

[24] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[25] Chee-Peng Lim, Ashlesha Vaidya, Yen-Wei Chen, Tejasvi Jain, Lakhmi, and C Jain, editors. *Artificial Intelligence and Machine Learning for Healthcare*. Springer, 2023.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[27] Ming Liu, Jue Jiang, and Zenan Wang. Colonic polyp detection in endoscopic videos with single shot detection based deep convolutional neural network. *IEEE Access*, 7:45058–45066, 2019.

[28] Le Lu, Yefeng Zheng, Gustavo Carneiro, and Lin Yang, editors. *Deep Learning and Convolutional Neural Networks for Medical Image Computing*. Advances in Computer Vision and Pattern Recognition. Springer, Cham, 2017.

[29] Ministerio de Salud, Argentina. Cáncer colorrectal. http://www.argentina.gob.ar/salud/cancer/tipos/cancer-colorrectal-ccr.

[30] Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[31] Michael A. Nielsen. Neural networks and deep learning, 2018.

[32] Observatorio Global del Cáncer. Estadísticas del cáncer a nivel mundial. https://gco.iarc.fr/today/data/factsheets/cancers/40-All-cancers-excluding-non-melanoma-skin-cancer-fact-sheet.pdf, 2020.

[33] Ishak Pacal and Dervis Karaboga. A robust real-time deep learning based automatic polyp detection system. *Computers in Biology and Medicine*, 134, 7 2021.

[34] Ishak Pacal, Dervis Karaboga, Alper Basturk, Bahriye Akay, and Ufuk Nalbantoglu. A comprehensive review of deep learning in colon cancer. *Computers in Biology and Medicine*, 126, 11 2020.

[35] Ishak Pacal, Ahmet Karaman, Dervis Karaboga, Bahriye Akay, Alper Basturk, Ufuk Nalbantoglu, and Seymanur Coskun. An efficient real-time colonic polyp detection with yolo algorithms trained by using negative samples and large datasets. *Computers in Biology and Medicine*, 141, 2 2022.

[36] Enrique Leo. Portiansky. *Analisis multidimensional de imagenes digitales.* D - Editorial de la Universidad de La Plata, 2013.

[37] Roger S. Pressman. *Ingenieria del software : un enfoque practico.* McGraw-Hill, 2010.

[38] Hemin Ali Qadir, Younghak Shin, Johannes Solhusvik, Jacob Bergsland, Lars Aabakken, and Ilangko Balasingham. Toward real-time polyp detection using fully cnns for 2d gaussian shapes prediction. *Medical Image Analysis*, 68, 2 2021.

[39] Q Ramírez, A Juan, M Chacón, and I Mario. Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década. artificial neural networks for image processing, a review of the last decade, 2011.

[40] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 6 2015.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. 6 2015.

[42] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach.* Pearson, 3 edition, 2009.

[43] Younghak Shin, Hemin Ali Qadir, Lars Aabakken, Jacob Bergsland, and Ilangko Balasingham. Automatic colon polyp detection using region based deep cnn and post learning approaches. *IEEE Access*, 6:40950–40962, 7 2018.

[44] Himanshu Singh. *Practical Machine Learning and Image Processing.* Apress, 2019.

[45] Sociedad Americana del Cáncer. Estudios por imágenes y cáncer, 2016.

[46] Sociedad Americana del Cáncer. Colonoscopia, 2019.

[47] Ian Sommerville. *Software engineering.* Pearson, 2011.

[48] Amnon Sonnenberg, Fabiola Delco, and John M Inadomi. Cost-effectiveness of colonoscopy in screening for colorectal cancer. *Annals of internal medicine*, 133(8):573–584, 2000.

[49] Sudhir Sornapudi, Frank Meng, and Steven Yi. Region-based automated localization of colonoscopy and wireless capsule endoscopy polyps. *Applied Sciences (Switzerland)*, 9, 6 2019.

[50] Tsung-Ying Sun, Chan-Cheng Liu, Chun-Ling Lin, Sheng-Ta Hsieh, and Cheng-Sen Huang. *A Radial Basis Function Neural Network with Adaptive Structure via Particle Swarm Optimization.* 01 2009.

[51] J. Sutherland. *Scrum: El Arte de Hacer el Doble de Trabajo en la Mitad de Tiempo.* Editorial Oceano de Mexico, 2021.

[52] Richard Szeliski. Computer vision algorithms and applications, 2011.

[53] Luisa F. Sánchez-Peralta, Luis Bote-Curiel, Artzai Picón, Francisco M. Sánchez-Margallo, and J. Blas Pagador. Deep learning to find colorectal polyps in colonoscopy: A systematic literature review, 8 2020.

[54] Jordi Torres Viñals. *Python Deep Learning : introducción práctica con Keras y TensorFlow 2.* Marcombo, 2020.

[55] Dechun Wang, Ning Zhang, Xinzi Sun, Pengfei Zhang, Chenxi Zhang, Yu Cao, and Benyuan Liu. Afp-net: Realtime anchor-free polyp detection in colonoscopy. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2019-November:636–643, 11 2019.

[56] Jianwei Xu, Ran Zhao, Yizhou Yu, Qingwei Zhang, Xianzhang Bian, Jun Wang, Zhizheng Ge, and Dahong Qian. Real-time automatic polyp detection in colonoscopy using feature enhancement module and spatiotemporal similarity correlation unit. *Biomedical Signal Processing and Control*, 66, 4 2021.

[57] Ruikai Zhang, Yali Zheng, Carmen C.Y. Poon, Dinggang Shen, and James Y.W. Lau. Polyp detection during colonoscopy using a regression-based convolutional neural network with a tracker. *Pattern Recognition*, 83:209–219, 11 2018.

[58] Zhenwei Zhang and Ervin Sejdić. Radiological images and machine learning: Trends, perspectives, and prospects. *Computers in Biology and Medicine*, 108:354–370, 2019.

[59] Yali Zheng, Ruikai Zhang, Ruoxi Yu, Yuqi Jiang, Tony W. C. Mak, Sunny H. Wong, James Y. W. Lau, and Carmen C. Y. Poon. Localisation of colorectal polyps by convolutional neural network features learnt from white light and narrow band endoscopic images of multiple databases. pages 4142–4145, 2018.

Siglas

AI *Artificial Intelligence* (Inteligencia Artificial). 1, 4, 5, 7, 8, 51, 53, 54

ANN *Artificial Neural Network* (Red Neuronal Artificial). 7, 9, 11

AP *Average Precision* (precisión media). 15

CNN *Convolutional Neural Network* (Red Neuronal Convolucional). 7, 11, 12, 16–18, 22, 23

CRC *Colorectal Cancer* (Cáncer Colorrectal). 3

CT *Computed Tomography* (Tomografía Computada). 20

DL *Deep Learning* (Aprendizaje Profundo). 1, 4, 5, 7–9, 21, 22, 53

DNN *Deep Neural Network* (Red Neuronal Profunda). iii, v, 7

Faster R-CNN *Faster Regions with Convolutional Neural Networks.* 1, 17–19, 22, 23, 25

FN *False Negative* (Falso Negativo). 14, 15

FP *False Positive* (Falso Positivo). 14, 15

GCO *Global Cancer Observatory* (Observatorio Global del Cáncer). 3

GPU *Graphics Processing Unit* (Unidad de Procesamiento Gráfico). 29, 37, 56

IARC *International Agency for Research on Cancer* (Agencia Internacional de Investigación del Cáncer). 3

IOU *Intersection Over Union* (Intersección Sobre Unión). 14

MICCAI *Medical Image Computing and Computer Assisted Intervention.* 22, 30

ML *Machine Learning* (Aprendizaje Automático). 1, 4, 7–9, 14, 32, 56

MLP *Multi Layer Perceptron* (Perceptrón Multicapa). 11

MRI *Magnetic Resonance Imaging* (Resonancia Magnética). 20

NMS *Non-maximum Suppression* (Supresión No Máxima). 19

PHP *Hypertext Preprocessor*. 36, 49

R-CNN *Regions with Convolutional Neural Networks*. 17–19, 23, 25

RBFNN *Radial Basis Function Neural Network* (Red Neuronal con Función de Base Radial). 12

ReLU *Rectified Lineal Unit* (Función ReLU). 10, 11

RNN *Recurrent Neural Networks* (Redes Neuronales Recurrentes). 12

RPN *Region Proposal Network* (Red de Propuestas). 19

SLP *Single Layer Perceptron* (Perceptrón Simple). 11

SSD *Single Shot Detector* (detector de un solo golpe). 23

SVM *Support Vector Machine* (Máquina de Soporte Vectorial). 17

TL *Transfer Learning* (Aprendizaje por Transferencia). 13

TN *True Negative* (Verdadero Negativo). 14

TP *True Positive* (Verdadero Positivo). 14, 46

X-Ray *X-Ray* (Radiografía). 20

XP *Extreme Programming* (Programación Extrema). 1, 25, 26, 48, 50, 54

YOLO *You Only Look Once*. iii, v, 1, 17–20, 23–25, 27–29, 31, 32, 47, 49, 54, 56