



universidad
de león

Departamento de Ingenierías
Mecánica, Informática y Aeroespacial

MÁSTER UNIVERSITARIO EN ROBÓTICA E INTELIGENCIA ARTIFICIAL

Trabajo de Fin de Máster

Detección de células defectuosas sobre un conjunto
de imágenes médicas

Detection of Defective Cells in a Set of Medical
Images

Autor: Aitor García Blanco

Tutor: Laura Fernández Robles

UNIVERSIDAD DE LEÓN
Departamento de Ingenierías Mecánica, Informática y Aeroespacial
MÁSTER UNIVERSITARIO EN ROBÓTICA E INTELIGENCIA ARTIFICIAL
Trabajo de Fin de Máster

ALUMNO: Aitor García Blanco

TUTOR: Laura Fernández Robles

TÍTULO: Detección de células defectuosas sobre un conjunto de imágenes médicas

TITLE: Detection of Defective Cells in a Set of Medical Images

CONVOCATORIA: Septiembre, 2025

RESUMEN:

ABSTRACT:

Palabras clave:

Firma del alumno:

VºBº Tutor:

Índice

Índice de figuras	IV
Índice de tablas	V
Glosario de términos	VI
1. Introducción y objetivos	1
1.1. Introducción	1
1.2. Objetivos	2
2. Antecedentes	3
2.1. Contexto y motivación	3
2.2. Estado del arte e hipótesis	3
2.3. Definición del problema	3
3. Gestión de proyecto software	4
3.1. Alcance del proyecto	5
3.1.1. Definición del proyecto	5
3.1.2. Presupuesto	5
3.2. Plan de trabajo	5
3.2.1. Metodología	5
3.2.2. Identificación de tareas	5
3.2.3. Estimación de tareas	5
3.2.4. Planificación de tareas	5
3.3. Gestión de recursos	5
3.3.1. Especificación y asignación de recursos	5
3.4. Gestión de riesgos	5
3.4.1. Identificación y análisis de riesgos	5
3.5. Legislación y normativa	6
4. Metodología	8
4.1. Preprocesado del <i>dataset</i>	8
4.2. Entrenamiento YOLO	8
4.3. Entrenamiento Fast-RCNN	8

4.4. Ensamblado (Ensemble)	8
5. Experimentación	9
5.1. Dataset	9
5.2. Entorno de desarrollo	10
5.2.1. <i>Hardware</i>	10
5.2.2. <i>Software y Frameworks</i>	11
5.3. Configuraciones	12
5.3.1. Entrenamiento	12
5.3.2. Validación cruzada	17
5.4. Métricas	17
6. Resultados	19
6.1. Resultados de los modelos	19
6.1.1. YOLO	19
6.1.2. Fast-RCNN	20
6.1.3. Ensemble	20
6.1.4. Modelo personalizado	20
6.2. Discusión de resultados	20
7. Herramienta Web	22
7.1. Objetivos de la herramienta	22
7.2. Arquitectura	22
7.2.1. Organización del proyecto	23
7.2.2. Capas funcionales en Streamlit	23
7.2.3. Flujo de ejecución	24
8. Conclusión	25
8.1. Conclusión general	25
8.2. Limitaciones del estudio	25
8.3. Lineas de trabajo futuro	25
Bibliografía	26
A. Control de versiones	28
B. Seguimiento del proyecto	29
C. Herramienta Web	30
C.1. Manual de usuario	30

C.2. Recomendaciones de uso	33
C.3. Resolución de problemas	33

Índice de figuras

5.1.	Interfaz de la aplicación web.	14
5.2.	Interfaz de la aplicación web.	14
5.3.	Interfaz de la aplicación web.	15
5.4.	Interfaz de la aplicación web.	15
5.5.	Interfaz de la aplicación web.	16
5.6.	Interfaz de la aplicación web.	16
C.1.	Interfaz de la aplicación web.	30
C.2.	Herramienta web: <i>checkbox ground truth</i> .	31
C.3.	Herramienta web: <i>checkbox zoom</i> .	31
C.4.	Herramienta web para procesar un conjunto de imágenes.	32
C.5.	Herramienta web: métricas y matriz de confusión.	32

Índice de tablas

5.1.	Distribución del dataset original	9
5.2.	Resumen del dataset de actuación	10
5.3.	Hiperparámetros principales de entrenamiento para cada modelo. . .	13
5.4.	Principales hiperparámetros óptimos seleccionados para cada modelo.	13
6.1.	Resultados de los modelos YOLOv7 sobre el conjunto de test original.	19
6.2.	Resultados de los modelos YOLO sobre el conjunto de test original. .	19
6.3.	Evaluación de modelos YOLO sobre los diferentes conjuntos de prueba	20
6.4.	Evaluación de modelos YOLO sobre imágenes con artefactos	21

Glosario de términos

Bounding Box : Rectangulo que delimita la posición y tamaño de un objeto en una imagen.

Dataset : Conjunto de datos estructurados, utilizado para entrenar, validar o evaluar modelos de Inteligencia Artificial.

IoU (*Intersection over Union*) : Área de unión entre dos *bounding boxes* (anotación y predicción).

Métricas : Indicadores cuantitativos para la evaluación de los modelos de Inteligencia Artificial.

Experto : Persona con conocimientos especializados en un dominio específico (como biólogos, biotecnólogos, patólogos, etc)

Detección de Objetos : Tarea de localizar y clasificar instancias de objetos en imágenes mediante *bounding boxes* y etiquetas.

NMS (*Non-Maximum Suppression*) : Algoritmo para eliminar las detecciones redundantes, manteniendo la caja con mayor confianza entre las solapadas

Aprendizaje profundo o *deep learning* : Área del aprendizaje automático que emplea redes neuronales profundas para aprender representaciones y resolver tareas complejas.

Transfer Learning : Reutilizar un modelo preentrenado y adaptarlo para entrenar un nuevo modelo con el conocimiento ya adquirido previamente con un dataset generalmente mayor.

Ensemble : Técnica para la unificación de predicciones de diferentes modelos.

Cross Validatio : Técnica estadística para la evaluación de modelos, que se fundamenta en dividir el dataset en múltiples pliegues (*folds*) para entrenar y validar el modelo de forma iterativa.

Data augmentation : Conjunto de transformaciones sobre los datos como rotaciones, traslaciones, adición de ruido, entre otros, con el objetivo de aumentar la diversidad del dataset y mejorar la generalización del modelo.

Optuna : Biblioteca de Python para la optimización de hiperparámetros, permitiendo encontrar las mejores configuraciones de estos.

Streamlit : Framework de Python para crear aplicaciones web interactivas orientadas a la ciencia de datos y *Machine Learning*.

Monolítica : Arquitectura de software en la que la aplicación se despliega y ejecuta en un mismo proceso.

LabelImg : Herramienta gráfica de código abierto para la anotación de imágenes con *bounding boxes*.

Widget : Elemento de una interface interactiva que permite al usuario controlar el comportamiento de una aplicación.

Ground truth : Hace alusión a las anotaciones correctas que describen las *bounding boxes* de las imágenes.

Checkbox : Es un *widget* que permite al usuario activar o desactivar una opción (booleano).

linter : Herramienta que analiza automáticamente el código fuente para detectar errores de sintaxis u otros posibles fallos antes de ejecutar el programa.

1. Introducción y objetivos

1.1. Introducción

El presente Trabajo de Fin de Máster, desarrollado en colaboración con la empresa Microptic S.L [12], y enmarcado en el contexto de un proyecto europeo DIGIS3 [1], aborda el desarrollo de una Prueba de Concepto (*Proof of Concept*) para la detección de células redondas en imágenes médicas mediante Inteligencia Artificial. Este trabajo incluye, además, la creación de una interfaz digital que permite la interacción de expertos con los resultados del modelo.

El objetivo de esta solución es la identificación automatizada de células redondas en imágenes de muestras de semen. Esta capacidad es de gran interés para Microptic [12], ya que optimiza los tiempos de análisis de los expertos y enriquece el estudio de las muestras espermáticas, mejorando la caracterización de parámetros clave tanto en el ámbito de la reproducción asistida humana como en la investigación y producción animal.

La metodología se ha centrado, en primer lugar, en la consolidación de un conjunto de datos de alta calidad. Para ello, se procesó y validó el feedback proporcionado por los expertos de Microptic [12], refinando el etiquetado de las imágenes que constituyen la base para el entrenamiento y la evaluación de los modelos.

Posteriormente, aplicando técnicas de Aprendizaje Profundo (*Deep Learning*) y Visión por Computador, se adaptaron y reentrenaron varias versiones del modelo de detección de objetos en tiempo real YOLO. El propósito fue especializar su capacidad, pasando de la detección de objetos cotidianos a la identificación precisa de las células redondas para interés de la empresa. Finalmente, el modelo fue entrenado y evaluado para validar su rendimiento, eficacia y viabilidad.

De este modo, el proyecto trasciende la investigación académica para materializar la misión de DIGIS3 [1]: "impulsar la digitalización de las empresas a través de soluciones tecnológicas avanzadas". La herramienta web desarrollada actúa como un catalizador de la transformación digital para Microptic [12], traduciendo un complejo modelo de Inteligencia Artificial en una solución práctica que potencia sus capacidades de análisis y establece un precedente para futuras innovaciones.

1.2. Objetivos

Este Trabajo de Fin de Máster tiene como objetivo principal aplicar y consolidar los conocimientos adquiridos durante la maestría mediante el desarrollo de un sistema de detección automatizada de células redondas en imágenes biomédicas; demostrando la integración de competencias técnicas en inteligencia artificial, procesamiento de imágenes y desarrollo de software.

Para dar forma a este proceso, se definen los siguientes pasos:

- Investigar y estudiar casuísticas análogas en la detección de células en conjuntos de imágenes médicas.
- Realizar un preprocesamiento y anotación de los conjuntos de imágenes médicas.
- Implementar, entrenar y validar diferentes modelos u arquitecturas de modelos, comparar su rendimiento y optimizar hiperparámetros.
- Evaluar cuantitativamente los modelos mediante métricas estandar.
- Desarrollar una herramienta web interactiva para visualización, inferencia y exportación de resultados que facilite la validación por parte de expertos biomédicos.
- Garantizar la reproducibilidad y cumplimiento ético/legislativo.
- Documentar exhaustivamente el proyecto.

2. Antecedentes

2.1. Contexto y motivación

Añadir en introducción, las referencias bibliográficas a digis3, microptic y del grupo GVIS. Aquí queiro habalar del contexto de la IA, dela visión por computador, detección de objetos y las motivaciones

2.2. Estado del arte e hipótesis

Aqui hablamos de los papers, del proyecto de GVIS de la ule y las hipótesis.

2.3. Definición del problema

3. Gestión de proyecto software

3.1. Alcance del proyecto

3.1.1. Definición del proyecto

3.1.2. Presupuesto

Coste de personal

Coste del hardware

Costes indirectos

Coste total

3.2. Plan de trabajo

3.2.1. Metodología

3.2.2. Identificación de tareas

3.2.3. Estimación de tareas

3.2.4. Planificación de tareas

3.3. Gestión de recursos

3.3.1. Especificación y asignación de recursos

3.4. Gestión de riesgos

3.4.1. Identificación y análisis de riesgos

3.5. Legislación y normativa

En el marco de ejecución de este proyecto, se ha llevado a cabo un riguroso cumplimiento de la legislación y normativa vigente. A continuación, se detalla cómo el proyecto se ajusta y adhiere a las leyes pertinentes:

- **Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [6]**

Este proyecto respeta plenamente la Ley Orgánica 3/2018, la cual reconoce el derecho fundamental a la protección de datos personales. La creación y tratamiento del dataset de imágenes médicas se ha realizado conforme a las disposiciones de la ley, asegurando la legalidad en el tratamiento de datos biomédicos. Se han obtenido los permisos explícitos necesarios para el uso de las imágenes, garantizando la privacidad y anonimización de los datos de los pacientes.

- **Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (RGPD) [5]**

La creación y tratamiento del dataset de imágenes médicas en este proyecto, se ha realizado conforme a los principios del RGPD, garantizando la legalidad, transparencia en el tratamiento de datos biomédicos. Todas las imágenes han sido previamente anonimizadas y se han implementado medidas técnicas y organizativas para asegurar la seguridad y privacidad de los datos, cumpliendo así con las exigencias del RGPD para datos de salud considerados de categoría especial.

- **Reglamento (UE) 2024/1689 del Parlamento Europeo y del Consejo sobre Inteligencia Artificial [7]**

Este reglamento establece normas armonizadas para garantizar la seguridad, ética y transparencia en el desarrollo y aplicación de sistemas de IA en la Unión Europea. En el contexto de este TFM, el sistema desarrollado se considera una herramienta de investigación y apoyo a la evaluación biomédica —no está concebido ni validado para toma de decisiones clínicas autónomas— por lo que su uso actual no se presenta como IA de alto riesgo. No obstante, para alinearse con los requisitos del AI Act se han incorporado las siguientes medidas:

- Documentación técnica completa: descripción del dataset (origen, anonimización), preprocesado, arquitecturas, versiones de modelos y parámetros de entrenamiento, así como resultados de pruebas y validación.
 - Evaluación de riesgos y validación: análisis de riesgos potenciales (falsos negativos/positivos), pruebas experimentales y métricas reproducibles que soportan las conclusiones presentadas.
 - Supervisión humana: la salida del sistema está diseñada para ser interpretada y validada por personal experto; no se recomienda uso clínico sin supervisión experta.
 - Transparencia y comunicación de limitaciones: en la memoria y la interfaz se documentan el ámbito de aplicación, las limitaciones conocidas y los intervalos de confianza relevantes.
 - Trazabilidad y registro: control de versiones de datos y modelos, y registro de ejecuciones y resultados para facilitar auditoría y reproducibilidad.
 - Plan de monitorización: recomendaciones para monitorizar rendimiento y sesgos en caso de desplegarse en entornos reales.
- **Real Decreto Legislativo 1/1996 sobre Propiedad Intelectual [4]**

En conformidad con el Real Decreto Legislativo 1/1996, el proyecto respeta la normativa sobre propiedad intelectual. Se ha optado por utilizar únicamente código y herramientas de software libre y de código abierto para garantizar el cumplimiento de la normativa en materia de propiedad intelectual.

4. Metodología

En esta sección se describen de forma reproducible los cuatro pipelines experimentales implementados: (1) Preprocesado del *dataset*, (2) Entrenamiento YOLO, (3) Entrenamiento Fast-RCNN (4) Ensamblado (Ensemble).

Aquí tengo que hablar de la arquitectura del código.

4.1. Preprocesado del *dataset*

4.2. Entrenamiento YOLO

4.3. Entrenamiento Fast-RCNN

4.4. Ensamblado (Ensemble)

5. Experimentación

5.1. Dataset

El dataset inicial proporcionado por la empresa MICROPTIC está constituido por un conjunto de *train* y *test* en formato PascalVOC.

Partición	Resoluciones (px)	Nº imágenes (resolución)	Nº imágenes	Sin instancias	Escala grises
<i>Train</i>	1280 × 1024	356	373	23	3 canales
	768 × 616	17			
<i>Test</i>	1280 × 1024	87	94	6	3 canales
	768 × 616	7			

Tabla 5.1: Distribución del dataset original

El conjunto de datos inicial que nos proporciona Microptic [12] está compuesto por 373 imágenes para *train* y 94 para *test*; de las cuales no presentan anotaciones: 23 imágenes de *train* y 6 de *test*. El conjunto de entrenamiento se divide utilizando la función `train_test_split` de *scikit-learn*, reservando el 80% (298 imágenes) para entrenamiento y el 20% (75 imágenes) para validación. Esta división se realiza de forma aleatoria pero reproducible, fijando la semilla (`random_state = 42`) para garantizar la consistencia de los resultados.

Adicionalmente, la empresa proporciona un conjunto de datos del *test* reevaluado por expertos del dominio. Este conjunto se reetiqueta utilizando la herramienta *LabelImg* [2] en formato PascalVOC. Las correcciones afectan a un total de 60 imágenes, incrementando el número de instancias de 1273 a 1412.

Asimismo, se incluyeron dos conjuntos adicionales denominados *test2* y *test3*, inicialmente sin anotaciones pero que contenían *bounding boxes* generados por modelos YOLO preentrenados por el grupo GVIS de la Universidad de León. Estas predicciones fueron posteriormente corregidas y validadas por expertos, proporcionando dos conjuntos adicionales para evaluación. La composición final del dataset se presenta en la Tabla 5.2.

Partición	Imágenes	Instancias	1280×1024	768×616	Escala grises
<i>Train</i>	298	3934	282	16	3 canales
<i>Validation</i>	75	878	74	1	3 canales
<i>Original_test</i>	94	1273	87	7	3 canales
<i>Test</i>	94	1412	87	7	3 canales
<i>Test2</i>	10	144	0	10	1 canal
<i>Test3</i>	59	1135	56	3	1 canal

Tabla 5.2: Resumen del dataset de actuación

El conjunto de *train*, *validation* y *test* están constituidos por imágenes RGB de 3 canales, mientras que las imágenes de los conjuntos de *test2* y *test3* están compuestos por imágenes en escala de grises de un único canal.

Para el entrenamiento de diferentes arquitecturas de detección de objetos, se convierten los formatos de PascaVOC a YOLO y YOLO a COCO. Para más deralle, la información relativa al preprocesamiento del dataset se encuentra en el documento `preprocesamiento.ipynb` del repositorio [10].

5.2. Entorno de desarrollo

Se utilizan dos entornos de desarrollo complementarios, garantizando la reproducibilidad y escalabilidad de los resultados obtenidos.

5.2.1. *Hardware*

Hardware Local

El entorno principal de desarrollo es un equipo *Lenovo IdeaPad Gaming 3* con las siguientes especificaciones técnicas:

- **Procesador:** AMD Ryzen 5 5600H with Radeon Graphics
 - Velocidad base: 3,30 GHz
 - Núcleos físicos: 6
 - Procesadores lógicos: 12

- Caché L1: 384 kB
- Caché L2: 3,0 MB
- Caché L3: 16,0 MB
- Virtualización: Habilitada
- **GPU:** NVIDIA GeForce RTX 3050 Laptop GPU
 - Memoria dedicada: 4,0 GB GDDR6
 - Arquitectura: Ampere
 - Soporte CUDA: 12.9
 - Driver version: 576.02
- **Memoria RAM:** 16 GB DDR4 SODIMM
 - Velocidad: 3200 MHz
 - Configuración: 2 módulos de 8 GB
- **Almacenamiento:** SSD NVMe PCIe Gen3 x4
 - Capacidad: 512 GB
 - Modelo: Micron MTFDHBA512QFD

Servidor privado

Como entorno complementario se ha empleado un servidor remoto proporcionado por el grupo GVIS de la Universidad de León.

- **GPU:** Tesla T4 con 15 GB de memoria
- **RAM del sistema:** 12,7 GB
- **Almacenamiento temporal:** 78,2 GB SSD

5.2.2. *Software y Frameworks*

El desarrollo se realizó empleando el siguiente *stack* tecnológico:

- **Sistema Operativo:** Windows 11
- **Entorno Python:** Miniconda3 (entorno TFM)

- **IDE:** Microsoft Visual Studio Code
- **CUDA Toolkit:** Versión 12.9
- **Frameworks principales:**
 - PyTorch 2.6.0 con soporte CUDA 12.6
 - Ultralytics 8.3.177
 - OpenCV para procesamiento de imágenes
 - Optuna para optimización de hiperparámetros
 - Streamlit para desarrollo de interfaces web
 - Pandas para análisis y manipulación de datos
 - Matplotlib para visualización de datos
 - Scikit-learn para métricas y otras utilidades de *machine learning*
 - Jupyter Notebook para desarrollo y análisis interactivo
 - linter ruff para mantener un código limpio, coherente y más fácil de mantener.

Para el despliegue del proyecto se recomienda tener todas las dependencias del `requirements.txt` [10], el cual recoge todas las librerías y versiones necesarias para la correcta ejecución del entorno.

5.3. Configuraciones

5.3.1. Entrenamiento

Aquí tengo que poner una tabla con las configuraciones de entrenamiento para ser recreado y cumplir con las normas éticas.

Modelo	Nº Trials	Epoch Optuna	Epoch Train	Batch	Image Size
yolov8s	6	25	40	12	704
yolov9s	6	25	40	10	704
yolov10s	6	25	40	10	704
yolov11s	6	25	40	10	704
yolov12s	6	25	40	7	704
yolov12x	5	25	40	7	704
custom	10	25	40	12	704

Tabla 5.3: Hiperparámetros principales de entrenamiento para cada modelo.

Modelo	lr0	lrf	momentum	weight_decay	optimizer
yolov8s	0.00540	0.00399	0.90621	0.00010	SGD
yolov9s	0.00023	0.00153	0.84564	0.00033	AdamW
yolov10s	0.00540	0.00399	0.90621	0.00010	SGD
yolov11s	0.00023	0.00153	0.84564	0.00033	AdamW
yolov12s	0.00023	0.00932	0.91627	0.00087	AdamW
custom	0.00153	0.00111	0.89113	0.00015	AdamW

Tabla 5.4: Principales hiperparámetros óptimos seleccionados para cada modelo.

Hiperparámetros del data augmentation:

warmup_epochs: 5

warmup_momentum: 0.75

degrees: 45

translate: 0.1

scale: 0.06

flipud: 0.5

fliplr: 0.5

mosaic: 0

close_mosaic: 0

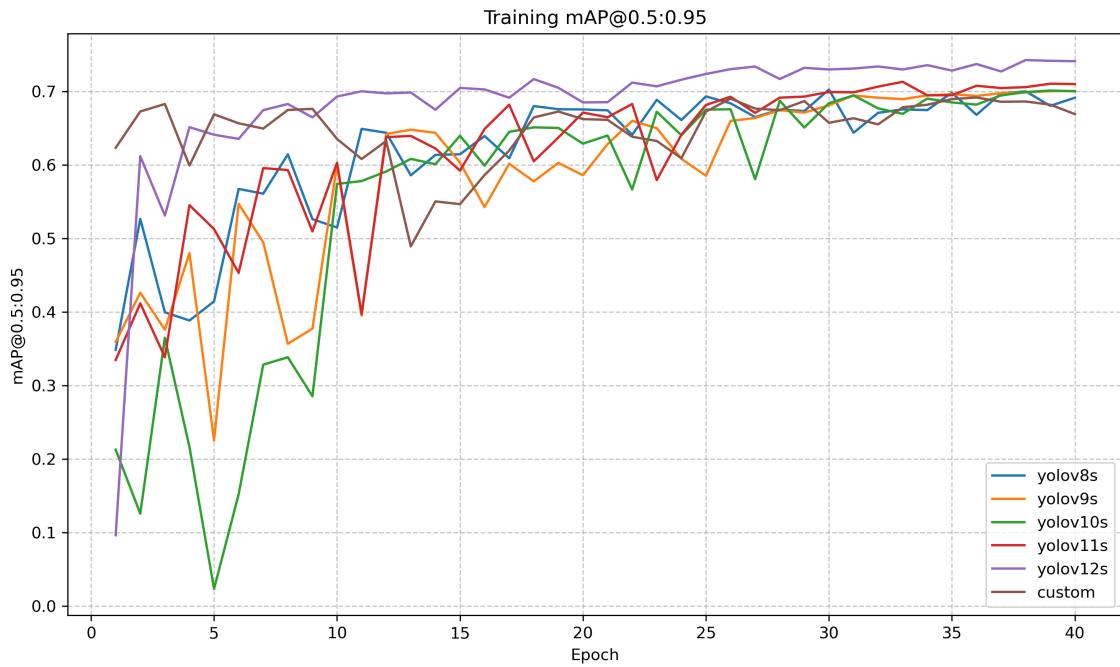


Figura 5.1: Interfaz de la aplicación web.

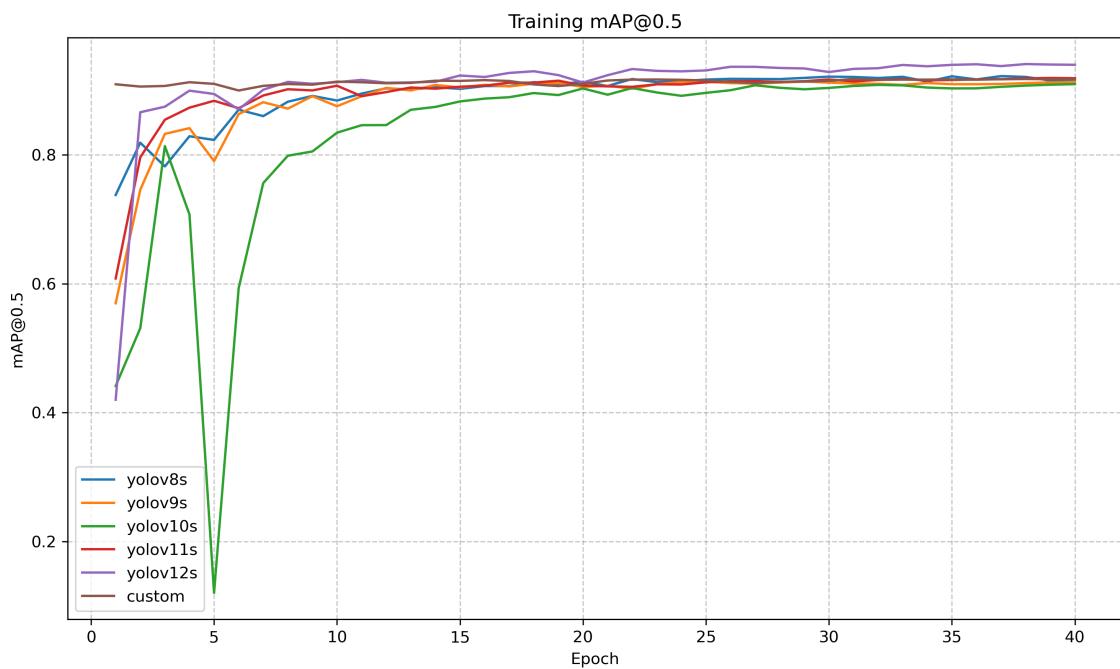


Figura 5.2: Interfaz de la aplicación web.

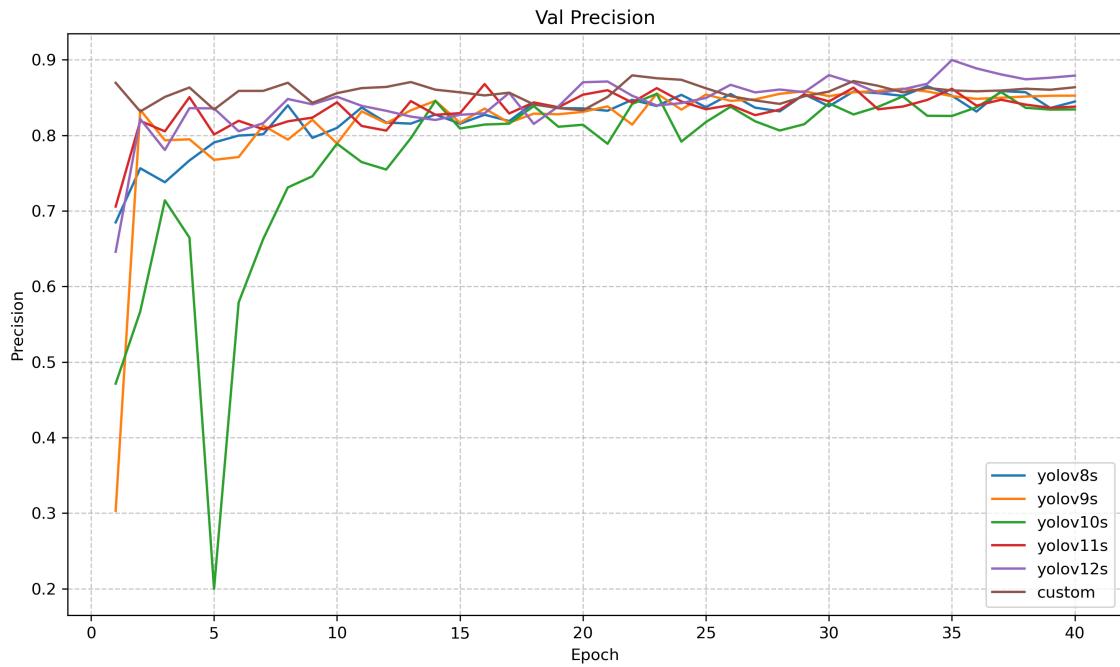


Figura 5.3: Interfaz de la aplicación web.

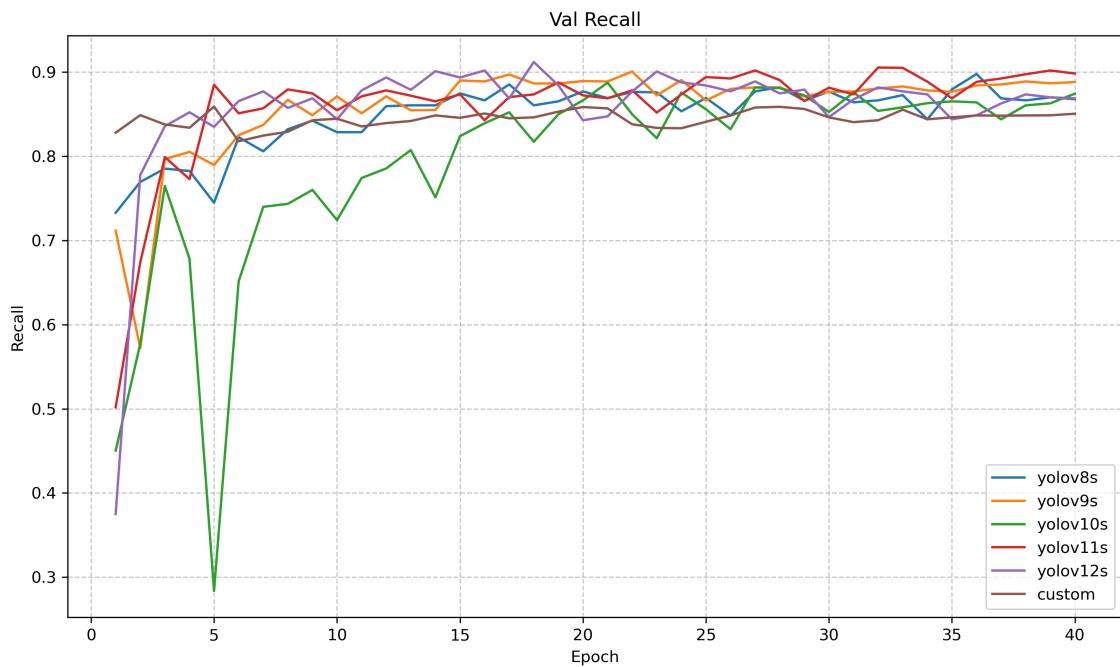


Figura 5.4: Interfaz de la aplicación web.

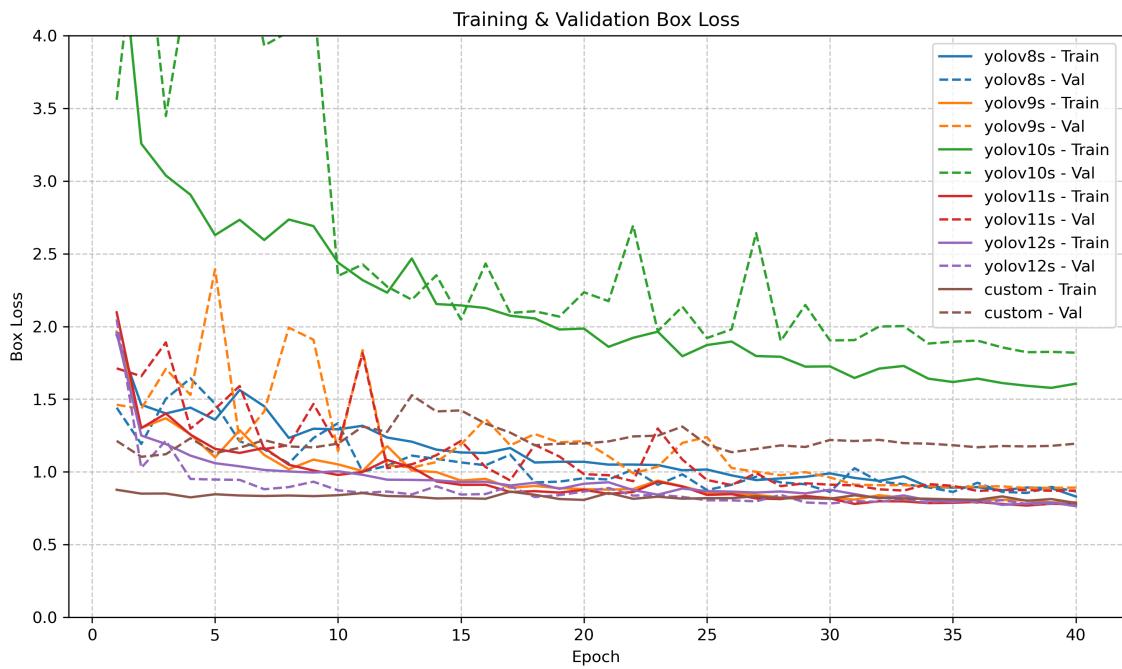


Figura 5.5: Interfaz de la aplicación web.

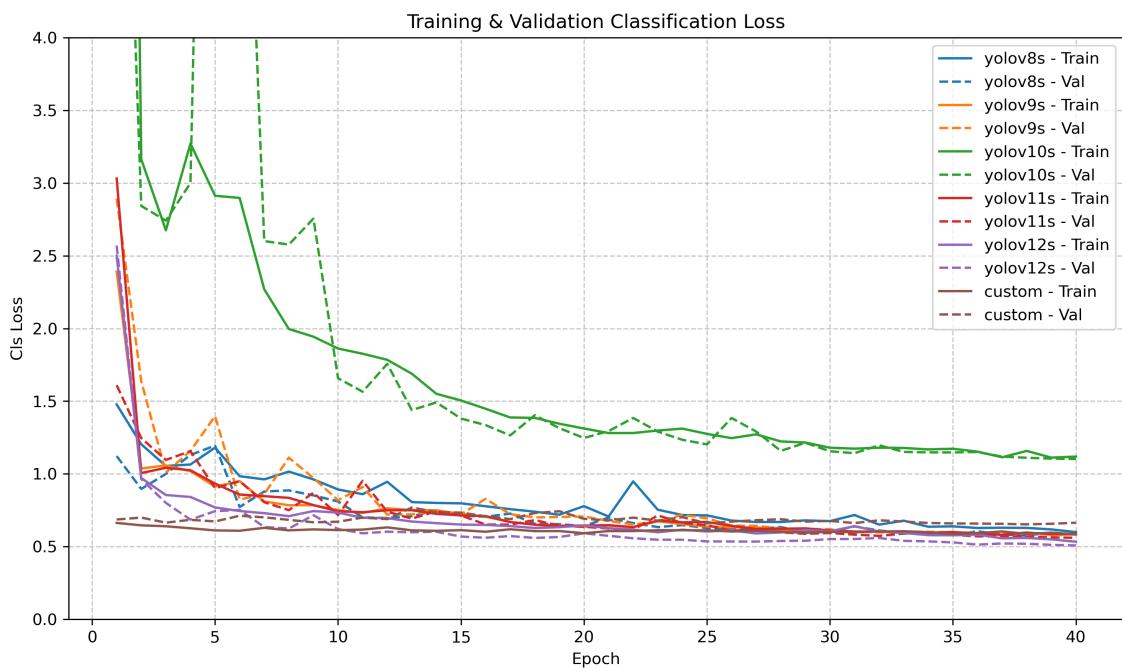


Figura 5.6: Interfaz de la aplicación web.

5.3.2. Validación cruzada

5.4. Métricas

Para la evaluación de los modelos, se emplea un conjunto de métricas fundamentales que se exponen a continuación.

Precision Mide la proporción de detecciones correctas entre todas las detecciones realizadas:

$$\text{Precision} = \frac{TP}{TP + FP}$$

donde TP son los verdaderos positivos y FP los falsos positivos. Alta precision indica pocas detecciones erróneas.

Recall : Mide la proporción de instancias reales que han sido detectadas:

$$\text{Recall} = \frac{TP}{TP + FN}$$

donde FN son los falsos negativos. Un recall alto indica que el modelo encuentra la mayoría de las instancias reales.

mean Average Precision (mAP) Para detección de objetos se calcula la curva precision–recall para cada clase y su área bajo la curva (AP). El *mean Average Precision* es la media de las AP sobre todas las clases:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^C \text{AP}_c$$

donde C es el número de clases. En detección se considera una predicción como verdadero positivo si el *Intersection over Union* (IoU) entre la caja predicha y la caja ground truth supera un umbral (por ejemplo $\text{IoU} \geq 0.5$).

mAP@0.5 (mAP50). Es la mAP calculada usando un único umbral de IoU igual a 0.5. Formalmente:

$$\text{mAP}@0.5 = \frac{1}{C} \sum_{c=1}^C \text{AP}_c(\text{IoU} = 0.5)$$

Es una medida menos exigente, que acepta solapamientos moderados entre predicción y *ground truth*.

mAP@[0.5:0.95] (mAP50:95). Es la mAP estándar del benchmark COCO que promedia la AP de cada clase sobre múltiples umbrales de IoU desde 0.50

hasta 0.95 con paso 0.05 (10 umbrales: 0.50, 0.55, ..., 0.95). Nota: COCO calcula cada AP integrando la curva precision–recall muestrada en 101 puntos, y después se promedian las AP en los 10 umbrales para obtener mAP@[0.5:0.95].

$$\text{mAP}_{[0.5:0.95]} = \frac{1}{C} \frac{1}{T} \sum_{c=1}^C \sum_{t \in \{0.50, 0.55, \dots, 0.95\}} \text{AP}_c(\text{IoU} = t)$$

donde $T = 10$. Esta métrica es más exigente porque penaliza detecciones con IoU bajos y refleja mejor la precisión espacial del modelo.

Inferencia (ms) Tiempo medio (en milisegundos) que tarda un modelo en procesar una única imagen (inferencia). Métrica relevante para proyectos en tiempo real.

6. Resultados

6.1. Resultados de los modelos

6.1.1. YOLO

Modelo	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5:0.95	Inferencia (ms)
YOLOv7	0.87	0.889	0.935	0.722	20.0
YOLOv7-W6	0.847	0.883	0.925	0.694	16.0
YOLOv7-E6E	0.906	0.857	0.934	0.721	127.5

Tabla 6.1: Resultados de los modelos YOLOv7 sobre el conjunto de test original.

Modelo	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5:0.95	Inferencia (ms)
yolov8s	0.843	0.844	0.920	0.722	11.006
yolov9s	0.838	0.852	0.921	0.721	13.485
yolov10s	0.821	0.867	0.918	0.717	11.221
yolov11s	0.861	0.824	0.921	0.732	11.248
yolov12s	0.856	0.863	0.936	0.761	14.615

Tabla 6.2: Resultados de los modelos YOLO sobre el conjunto de test original.

Modelo	Test	Precisión	Recall	mAP@0.5	mAP@0.5:0.95	Inferencia (ms)
yolov8s	test	0.855	0.838	0.926	0.704	11.067
yolov9s	test	0.851	0.876	0.933	0.706	12.904
yolov10s	test	0.853	0.861	0.927	0.701	11.041
yolov11s	test	0.846	0.866	0.930	0.716	10.644
yolov12s	test	0.861	0.848	0.936	0.740	15.066
yolov8s	test2	0.924	0.938	0.969	0.553	37.858
yolov9s	test2	0.917	0.927	0.961	0.542	35.944
yolov10s	test2	0.956	0.898	0.962	0.589	12.358
yolov11s	test2	0.929	0.917	0.964	0.534	14.967
yolov12s	test2	0.965	0.917	0.975	0.556	20.254
yolov8s	test3	0.842	0.862	0.937	0.700	11.314
yolov9s	test3	0.843	0.870	0.935	0.681	13.291
yolov10s	test3	0.873	0.826	0.926	0.667	11.109
yolov11s	test3	0.855	0.874	0.935	0.698	12.561
yolov12s	test3	0.836	0.821	0.922	0.723	15.442

Tabla 6.3: Evaluación de modelos YOLO sobre los diferentes conjuntos de prueba

6.1.2. Fast-RCNN

6.1.3. Ensemble

6.1.4. Modelo personalizado

6.2. Discusión de resultados

Imagen	Test	yolov8s 0.5	yolov8s 0.7	yolov9s 0.5	yolov9s 0.7	yolov10s 0.5	yolov10s 0.7	yolov11s 0.5	yolov11s 0.7	yolov12s 0.5	yolov12s 0.7
59 imagen	1	x	✓	x	✓	x	✓	✓	✓	✓	✓
61 imagen	1	x	✓	x	x	✓	✓	x	✓	✓	✓
219 imagen	1	x	✓	x	x	✓	✓	x	✓	✓	✓
369 imagen	1	✓	✓	x	x	✓	✓	x	✓	✓	✓
already tested-00	3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
already tested-15	3	✓	✓	x	x	✓	✓	x	✓	✓	✓
already tested-16	3	✓	✓	x	✓	✓	✓	✓	✓	✓	✓
already tested-17	3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
already tested-22	3	✓	✓	✓	✓	x	✓	✓	✓	✓	✓

Tabla 6.4: Evaluación de modelos YOLO sobre imágenes con artefactos

7. Herramienta Web

Con el propósito de definir una herramienta de análisis automatizado, se desarrolla una interfaz intuitiva e interactiva para la detección de células redondas sobre una muestra individual o colectiva. La información relativa a la aplicación, se puede encontrar en el manual de usuario (C.1).

7.1. Objetivos de la herramienta

La herramienta web para la detección de células redondas tiene como principales objetivos:

- **Detección automática de células en imágenes microscópicas en el contexto médico abordado con anterioridad:** Permitir identificar y localizar células redondas dotando al usuario de diferentes arquitecturas de modelos.
- **Comparación de rendimiento entre modelos:** Facilitar la evaluación comparativa entre diferentes modelos , permitiendo al usuario seleccionar el mejor modelo y el intervalo de actuación para el IoU.
- **Visualización de resultados:** Proporcionar una interface de visualización clara e intuitiva que permita al usuario visualizar las predicciones del modelo y las *bounding boxes* si se dispone de las mismas.
- **Análisis cuantitativo:** Permitir obtener métricas y estadísticas como: *precision*, *accuracy*, conteo o IoU promedio
- **Soporte a la investigación biomédica:** Facilitar la identificación temprana de diferentes grados de patología.

7.2. Arquitectura

La arquitectura de la aplicación sigue un diseño modular y escalable basada en *Streamlit*. Esto permite tener una aplicación monolítica que encapsula toda la funcionalidad en un único ejecutable y mantiene una clara separación entre la lógica de procesamiento y de presentación.

7.2.1. Organización del proyecto

La estructura principal es:

- `app.py`: orquestación de la interfaz (widgets Streamlit), gestión de sesión y del flujo de inferencia.
- `utils/utils.py`: servicios de *backend* lógico (carga de modelos, preprocesado, inferencia, postprocesado, visualización).
- `assets/styles.css`: estilo e integración visual mediante `st.markdown`.
- `models/{final_model_yolovXs}/`: modelos entrenados para la casuística.
- `runs/detect/`: salidas temporales de inferencia (imágenes anotadas, JSON/CSV).

Para más información, revisar la documentación `04.Code/cell_detection_App` en el repositorio del proyecto [10].

7.2.2. Capas funcionales en Streamlit

1. **Interfaz y orquestación (UI)**: componentes como `st.file_uploader`, `st.selectbox`, `st.slider`, `st.sidebar` y `st.image` permiten la interacción con el usuario de una forma intuitiva.
2. **Servicios de modelo e inferencia**: Funciones en `utils/utils.py` para cargar pesos YOLO, seleccionar dispositivo (CPU/GPU), normalizar entradas y ejecutar el modelo sobre las entradas preprocesadas.
3. **Pipeline de datos**: Preprocesado (lectura, redimensionado, normalización), postprocesado (NMS, filtrado por confianza, conversión a anotaciones Pascal-VOC), y renderizado de *bounding boxes*. Además, incluye la exportación de las predicciones en formato PascVOC.
4. **Estado y caché**: `st.session_state` para parámetros y resultados interactivos; `@st.cache_resource` evita recargar pesos al cambiar sólo parámetros de inferencia; `@st.cache_data` para memoizar resultados derivado (tablas/figuras).
5. **Estilos y experiencia de usuario**: `assets/styles.css` para mejorar el aspecto visual que por defecto ofrece streamlit y uso de layout responsivo (columnas, botones, tooltips) para mejorar la usabilidad.

6. **Manejo de errores:** `st.info` para informar de estados (dispositivo, imágener y xml subidos), `st.warning` para advertir de un error en la lectura del xml, `st.error` implide continuar con la ejecución si no encuentra modelo o al no ser capaz de procesar una imagen.

7.2.3. Flujo de ejecución

1. Inicio: se inicializa la sesión y se aplica el estilo personalizado CSS.
2. El usuario selecciona modelo e IoU: Yolov12s y 0,5.
3. Carga de modelo (una sola vez) vía `@st.cache_resource`.
4. Carga de imagen (individual o lote) con `st.file_uploader`. Opcional: cargar el xml con las anotaciones en formato PascVOC.
5. Preprocesado de la imagen y envío al dispositivo (CPU/GPU).
6. Inferencia YOLO y postprocesado (NMS, métricas básicas).
7. Visualización: imagen anotada, conteo, tablas y métricas, *bounding boxes*.

8. Conclusión

8.1. Conclusión general

8.2. Limitaciones del estudio

8.3. Lineas de trabajo futuro

Aportaciones realizadas

Problemas encontrados

Opiniones personales

Trabajos futuros

Agradecimientos

Bibliografía

- [1] DIGIS3, Digitalización Inteligente, Sostenible y Cohesiva. <https://digis3.eu/es>.
- [2] labelImg: Graphical image annotation tool. <https://github.com/tzutalin=labelImg>.
- [3] Online gantt — gantt chart maker. <https://www.onlinegantt.com/>.
- [4] Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual. Boletín Oficial del Estado, núm. 97, Abril 1996. Disponible en: <https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930>.
- [5] Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos. Diario Oficial de la Unión Europea, Abril 2016. Disponible en: <https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679>.
- [6] Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. Boletín Oficial del Estado, núm. 294, Diciembre 2018. Disponible en: <https://www.boe.es/eli/es/lo/2018/12/05/3/con>.
- [7] Reglamento (UE) 2024/1689 del Parlamento Europeo y del Consejo, de 13 de junio de 2024, por el que se establecen normas armonizadas en materia de inteligencia artificial (Reglamento de Inteligencia Artificial). Diario Oficial de la Unión Europea, L 2024/1689, julio 2024. Entrada en vigor: 1 de agosto de 2024. Disponible en: <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>.
- [8] Manual de laboratorio de la OMS para el examen y procesamiento del semen humano, sexta edición [WHO laboratory manual for the examination and processing of human semen, sixth edition]. Ginebra: Organización Mundial de la Salud, 2025.
- [9] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Tetsuro Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework.

- In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631. ACM, 2019.
- [10] Aitor García Blanco. Repositorio del TFM: Detección de células defectuosas en imágenes médicas. <https://github.com/AigarciaabFabero/TFM>, 2025.
 - [11] S. Long and S. Kenworthy. Round Cells in Diagnostic Semen Analysis: A Guide for Laboratories and Clinicians. *British Journal of Biomedical Science*, 79(10129), 2022.
 - [12] Microptic S.L. <https://www.micropticsl.com/>.
 - [13] Optuna Development Team. Optuna: Automl and hyperparameter optimization framework. <https://github.com/optuna/optuna>, 2025. Accessed: 2025-08-17.
 - [14] Priya S. Patil, Rajendra S. Humbarwadi, Ashalata D. Patil, and Anita R. Gune. Immature germ cells in semen — Correlation with total sperm count and sperm motility. *Journal of Cytology*, 30(3):185–189, July 2013.

Anexo A

Control de versiones

En el marco de desarrollo del Trabajo de Fin de Máster (TFM), se ha empleado GitHub como servicio de control de versiones para gestionar eficientemente el código fuente y la documentación del proyecto; facilitando el seguimiento, la trazabilidad y la colaboración.

Esta herramienta permite un seguimiento del proyecto por parte del responsable o tutor. Además, de ser necesario, GitHub presenta una funcionalidad para desarrolladores que permite editar el proyecto desde un navegador web. Facilitando la implementación de mejoras y la corrección de errores desde cualquier dispositivo con acceso a internet.

Casi la totalidad del desarrollo de este proyecto final lo podemos encontrar en el repositorio personal [10].

Anexo B

Seguimiento del proyecto

Anexo C

Herramienta Web

C.1. Manual de usuario

Para facilitar la interacción del usuario con la herramienta web vista con anterioridad, se define la siguiente guía de usuario.

Desde el entorno con la dependencia *Streamlit*, se ejecuta el comando `streamlit run app.py` se lanza la aplicación en el navegador predeterminado como se muestra en la siguiente imagen.

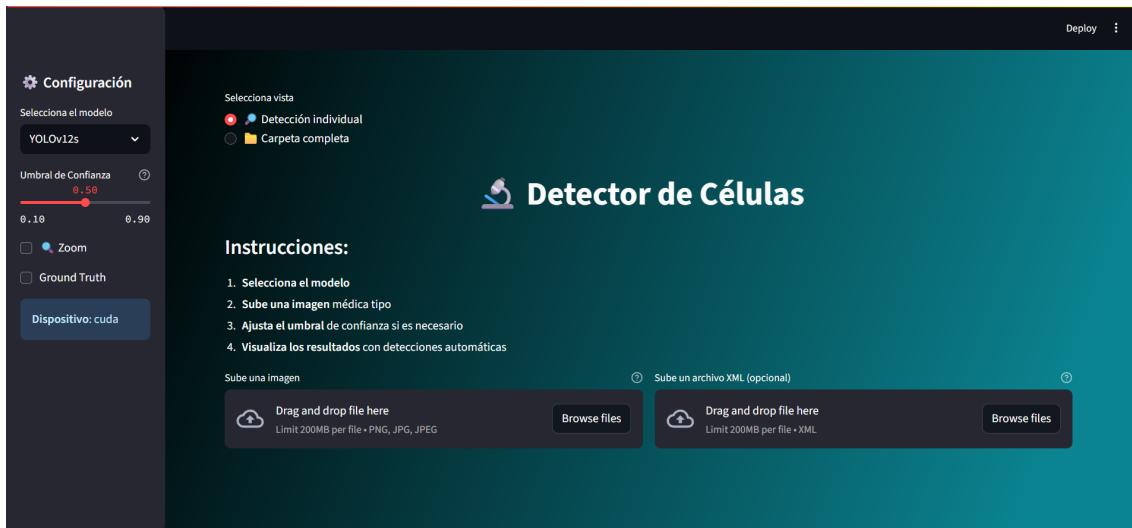


Figura C.1: Interfaz de la aplicación web.

En este punto, el usuario puede tomar la decisión de procesar una imagen individual o un conjunto de imágenes. Para esto, tenemos el *widget* que permite la selección entre "Detección individual" o "Carpeta completa". En ambos casos, la aplicación nos muestra una guia básica de funcionamiento como se puede apreciar en el apartado "Introducciones" de la imagen anterior. De manera totalmente opcional, el usuario puede cargar junto a la imagen a estudio, el correspondiente conjunto de anotaciones.

El *widget* de Configuración permite la selección manual del modelo a través de un desplazable, el umbral de confianza de IoU, la representación *ground truth* (siempre que exista el xml asociado a la imagen) mediante un *checkbox* (Figura C.2) y realizar un análisis exploratorio sobre la imagen con un zoom (Figura C.3). Asimismo, muestra el dispositivo (CPU,GPU) con el que se van a procesar las imágenes.

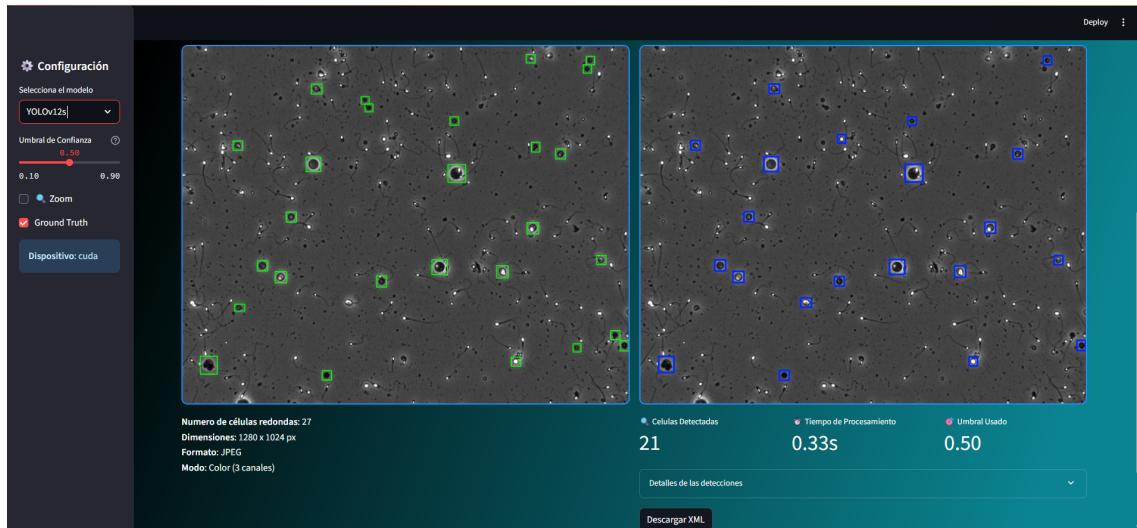


Figura C.2: Herramienta web: *checkbox ground truth*.



Figura C.3: Herramienta web: *checkbox zoom*.

Cabe destacar el *widget* "Descargar XML" que permite al usuario descargar las predicciones del modelo realizadas sobre la imagen de entrada en un formato Pascal-VOC. Además,

Por el contrario, si el usuario prefiere evaluar un conjunto de imágenes con sus respectivas anotaciones, se debe seleccionar la elección de "carpeta completa" como se muestra en la Figura C.4.

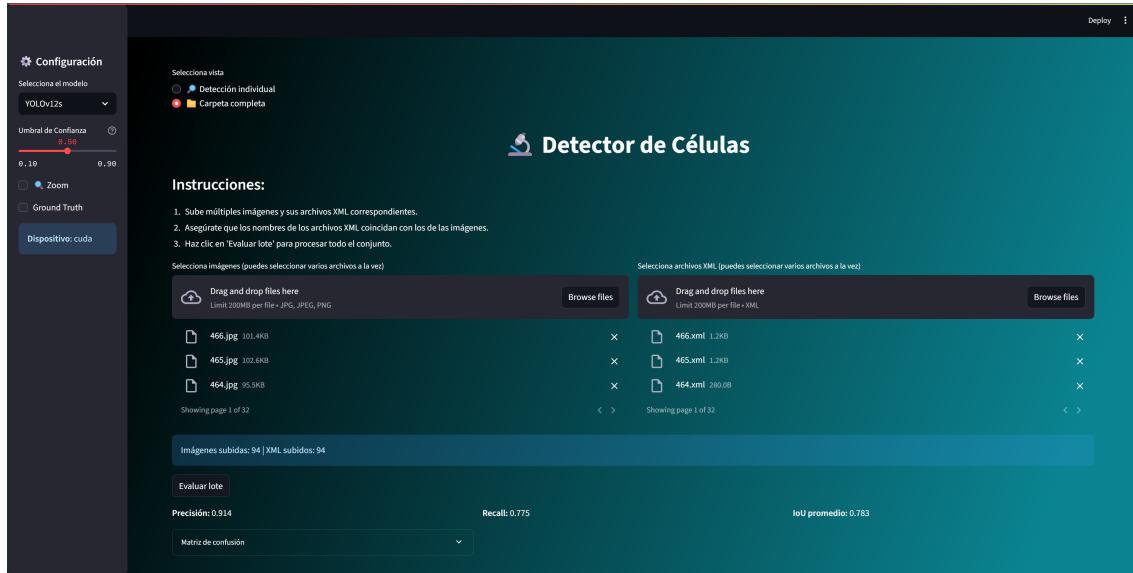


Figura C.4: Herramienta web para procesar un conjunto de imágenes.

Este es un modo de evaluación, no está permitido el uso de los *widgets*: *ground truth* y *zoom*. Sí embargo, el procesamiento del modelo seleccionado junto con su correspondiente umbral de procesamiento de IoU, está acompañado de un pequeño conjunto de métricas: *precision*, *recall* y *mAP*. Para una mejor evaluación, la herramienta muestra una matriz de confusión (Figura C.5).

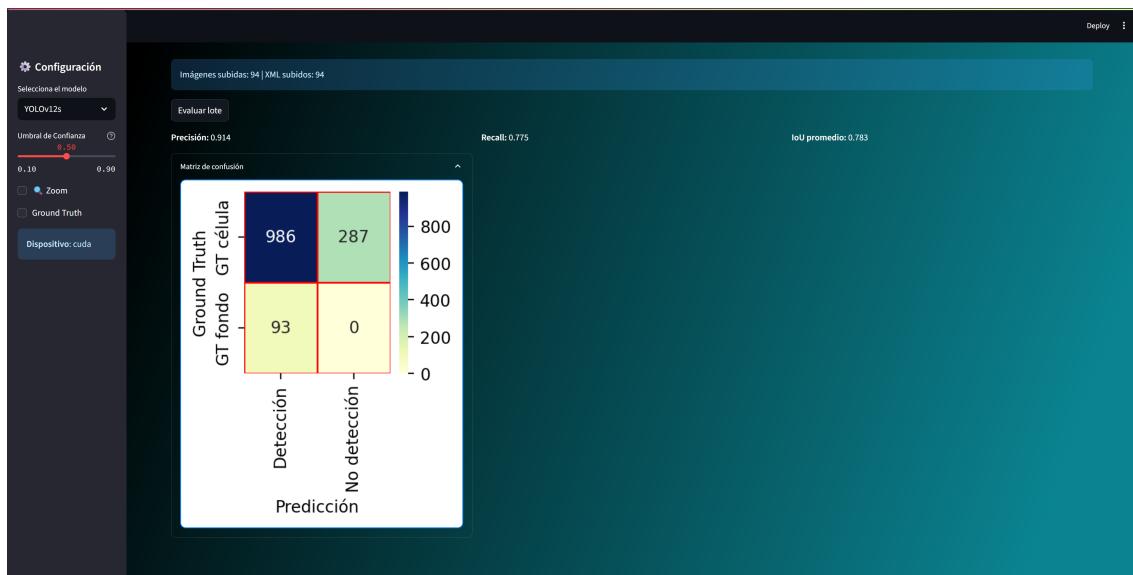


Figura C.5: Herramienta web: métricas y matriz de confusión.

C.2. Recomendaciones de uso

A continuación, se recogen pautas prácticas para el uso correcto funcionamiento de la herramienta web.

- **Entorno recomendado:** se recomienda usar un entorno conda para tener aisladas las dependencias del proyecto y evitar así conflictos entre dependencias. Desde el directorio 04.Code se proponen dos alternativas:
 - Instalación del entorno completo con conda (Recomendado):
 - `conda env create -f environment.yml`
 - `conda activate tfm_env`
 - Instalación de un subconjunto de dependencias con conda:
 - `conda create -n tfm_env python=3.11 -y`
 - `conda activate tfm_env`
 - `pip install -r 04.Codigo/requirements.txt`
- **Ejecución de la aplicación:** en la carpeta 04.Codigo/cell_detection_App ejecutar:
 - `streamlit run app.py`
 - Automáticamente se abre en el navegador por defecto la URL que Streamlit indique (por defecto `http://localhost:8501`).
- **Compatibilidad con modelos:** funciona únicamente con modelos de YOLO.

C.3. Resolución de problemas

Posibles problemas que pueden tener lugar durante la instalación:

- **La app no arranca / Streamlit muestra error:**
 - Verificar dependencias: `pip install -r 04.Codigo/requirements.txt`
 - Comprobar la salida en el terminal donde se ejecuta `streamlit run app.py` y revisar trazas de error.
 - Borrar caché de Streamlit: `streamlit cache clear`.

■ Modelo no encontrado o error de carga de pesos:

- Confirmar que el fichero de pesos existe en `models/` y que la ruta es correcta (`04.Codigo/cell_detection_App/models/`).

■ Problemas con GPU / CUDA:

- Comprobar estado de la GPU: ejecutar `nvidia-smi` en terminal.
- Verificar que PyTorch detecta la GPU:
 - `python -c "import torch; print(torch.cuda.is_available())"`
- Si falla, forzar ejecución en CPU desde la UI o variable de entorno: `CUDA_VISIBLE_DEVICES=""`.

■ Errores al procesar imágenes / XML mal formados:

- Validar que las imágenes están en formatos soportados (`.jpg`, `.jpeg`, `.png`) y que los XML cumplen con el formato PascalVOC.
- Revisar el log de parsing en `04.Codigo/cell_detection_App/utils/` y corregir los ficheros señalados.