



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	4
2.2.1. Visión por computador y modelos de detección de objetos . .	5
2.2.2. Técnicas de optimización y validación de modelos	8
2.2.3. Literatura de la investigación	9
2.2.4. Hipótesis de trabajo	9
2.3. Definición del problema	9
3. Gestión de proyecto software	13
3.1. Alcance del proyecto	14
3.1.1. Definición del proyecto	14
3.1.2. Presupuesto	14
3.2. Plan de trabajo	14
3.2.1. Metodología	14
3.2.2. Identificación de tareas	14
3.2.3. Estimación de tareas	14
3.2.4. Planificación de tareas	14
3.3. Gestión de recursos	14
3.3.1. Especificación y asignación de recursos	14
3.4. Gestión de riesgos	14
3.4.1. Identificación y análisis de riesgos	14
3.5. Legislación y normativa	15

4. Metodología	17
5. Experimentación	18
5.1. Dataset	18
5.2. Entorno de desarrollo	19
5.2.1. <i>Hardware</i>	19
5.2.2. <i>Software y Frameworks</i>	20
5.3. Configuraciones	21
5.3.1. Entrenamiento	21
5.3.2. Validación cruzada	26
5.3.3. Ensamblado de modelos	26
5.3.4. Modelo personalizado	26
5.4. Métricas	26
6. Resultados	28
6.1. Resultados de los modelos	28
6.1.1. YOLO	28
6.1.2. Ensemble	29
6.1.3. Modelo personalizado	29
6.2. Discusión de resultados	29
7. Herramienta Web	31
7.1. Objetivos de la herramienta	31
7.2. Arquitectura	31
7.2.1. Organización del proyecto	32
7.2.2. Capas funcionales en Streamlit	32
7.2.3. Flujo de ejecución	33
8. Conclusión	34
8.1. Conclusión general	34
8.2. Limitaciones del estudio	34
8.3. Lineas de trabajo futuro	34
Bibliografía	35
A. Control de versiones	37
B. Seguimiento del proyecto	38
C. Herramienta Web	39

C.1. Manual de usuario	39
C.2. Recomendaciones de uso	42
C.3. Resolución de problemas	42

Índice de figuras

2.1.	Diagramas de arquitectura de Yolov8, Yolov9 y Yolov10	6
2.2.	Diagramas de arquitectura de Yolov11 y Yolov12	7
2.3.	Ilustración con Anillos de Newton.	11
2.4.	Ilustración de annotaciones para <i>feedback</i> con experto.	12
5.1.	mAP@0.5:0.95 durante el proceso de entrenamiento.	23
5.2.	mAP@0.5 durante el proceso de entrenamiento.	23
5.3.	<i>Precision</i> del modelo sobre el conjunto de validación.	24
5.4.	<i>Recall</i> del modelo sobre el conjunto de validación.	24
5.5.	<i>Box Loss</i> sobre el conjunto de entrenamiento y validación.	25
5.6.	<i>Clasification Loss</i> sobre el conjunto de entrenamiento y validación.	25
C.1.	Interfaz de la aplicación web.	39
C.2.	Herramienta web: <i>checkbox ground truth</i>	40
C.3.	Herramienta web: <i>checkbox zoom</i>	40
C.4.	Herramienta web para procesar un conjunto de imágenes.	41
C.5.	Herramienta web: métricas y matriz de confusión.	41

Índice de tablas

5.1.	Distribución del dataset original	18
5.2.	Resumen del dataset de actuación	19
5.3.	Hiperparámetros principales de entrenamiento para cada modelo. . .	22
5.4.	Principales hiperparámetros óptimos seleccionados para cada modelo.	22
6.1.	Resultados de los modelos YOLOv7 sobre el conjunto de test original.	28
6.2.	Resultados de los modelos YOLO sobre el conjunto de test original. .	28
6.3.	Evaluación de modelos YOLO sobre los nuevos conjuntos de <i>test</i> . . .	29
6.4.	Evaluación de modelos YOLO sobre imágenes con artefactos	30

Glosario de términos

Bounding Box : Rectángulo 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.

framework : Conjunto estructurado de herramientas, librerías y convenciones que facilita y agiliza el desarrollo de aplicaciones o sistemas.

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.

Células redondas : Según la OMS [1], toda célula no espermática que se encuentra en el eyaculado. Agrupa principalmente dos tipos: células germinales inmaduras y leucocitos.

Células germinales inmaduras : Células precursoras de los espermatozoides que no han completado su desarrollo.

Leucocitos : Glóbulos blancos, principalmente neutrófilos.

Espermograma : Consiste en analizar los espermatozoides de un hombre o un animal para evaluar la fertilidad y detectar posibles anomalías.

Espermatogénesis : Proceso biológico de formación y producción de espermatozoides.

Anillos de Newton : Interferencia de la luz cuyo resultado es un patrón de anillos muy característico.

1. Introducción y objetivos

1.1. Introducción

El presente Trabajo de Fin de Máster, desarrollado en colaboración con la empresa Microptic S.L [2] compañía de Hamilton Thorne [3] y, enmarcado en el contexto de un proyecto europeo DIGIS3 [4], 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 [2], 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 [2], 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 [5]. El propósito fue especializar su capacidad, pasando de la detección de objetos cotidianos a la identificación precisa de células redondas para facilitar el espermograma. 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 [4]: "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 [2], 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

En este capítulo se expone el problema abordado en este trabajo, sus antecedentes e hipótesis y motivaciones. Asimismo, se revisa la literatura existente para situar el estado del arte y justificar la necesidad de la investigación realizada.

2.1. Contexto y motivación

El análisis de muestras de semen es un pilar fundamental en el campo de la medicina reproductiva. Su relevancia abarca desde el diagnóstico de la infertilidad en parejas y técnicas de reproducción asistida en humanos, hasta la investigación y producción en el sector ganadero. Tradicionalmente, este análisis se ha centrado en la motilidad, concentración y morfología de los espermatozoides.

Sin embargo, un análisis seminal requiere la evaluación de otros componentes celulares, entre los que destacan las células redondas. Estas constituyen un indicador diagnóstico crucial, ya que abarca linfocitos, células plasmáticas, macrófagos, mastocitos y células tumorales redondas; cuya presencia puede ser signo de infecciones o inflamaciones del tracto genital, como células germinales inmaduras, que pueden indicar una alteración en el proceso de espermatogénesis [6].

El método estandar para la identificación y conteo de células redondas depende de la agudeza visual de un experto a través de un microscopio; lo que convierte al proceso de espermatogenia es un proceso subjetivo (depende del observador), lento, laborioso y cansado para el observador. Estas limitaciones motivan la necesidad de desarrollar alternativas mediante herramientas automatizadas que permitan obtener resultados más rápidos, objetivos y reproducibles, optimizando el espermograma.

El auge de la Inteligencia Artificial, y en particular las técnicas de Visión por Computador basadas en Aprendizaje Profundo, están protagonizando una revolución dentro del mundo de la medicina. La capacidad de las redes neuronales en la detección de patrones complejos han demostrado ser una herramienta de apoyo en diversas especialidades como son: análisis de imágenes radiológicas o análisis de lesiones cutáneas en dermatología, entre otras. En *computer vision*, modelos de detección de objetos como YOLO (You Only Look Once) [5] han demostrado un

equilibrio entre precisión y velocidad de inferencia. Estos algoritmos son capaces de procesar imágenes y localizar multiples objetos de interes en milisegundos. Siendo idóneo en este tipo de trabajos clínicos.

Atendiendo a estas necesidades y casuísticas, por parte de una empresa lider en el análisis espermático, surge la necesidad de integrar estos sistemas inteligentes para la detección de células redondas al que dió solución el grupo GVIS de la Universidad de León mediante una prueba de concepto [6]. Demostrando así, la capacidad de estas tecnologías en este tipo de tareas complejas.

Aprovechando este hito, surge la motivación de definir una nueva prueba de concepto integrada en un entorno digital, que permita igualar o superar con menos recursos, la anterior prueba de concepto. Esto se aborda a través de diferentes vías:

- Explorar diferentes arquitecturas de YOLO [5], desde modelos oficiales hasta prototipos más recientes que emplean capas de atención.
- Diseñar un modelo personalizado para competir con los modelos existentes.
- Optimización y mejora de la detección: buscar la configuración optima que maximice diferentes métricas y abordar la detección de artefactos visuales que fueron identificados como una limitación en la PoC previa.
- Desarrollar de una herramienta web interactiva que no solo sirva para evaluar imágenes sino que además, de soporte a expertos.

2.2. Estado del arte e hipótesis

En esta sección se analiza el panorama tecnológico que sirve como fundamento para este proyecto. Se parte de las tecnologías de Inteligencia artificial que tienen relevancia en el campo de la medicina como es la visión por computador, se abordan los modelos de detección de objetos utilizados y se revisan las diferentes técnicas de optimización. Finalmente, se realiza un breve y conciso estudio sobre la literatura actual en este campo y se define la hipótesis de trabajo.

2.2.1. Visión por computador y modelos de detección de objetos

La Inteligencia Artificial está en auge y algunos de los campos en los que está tomando protagonismo son el Aprendizaje Profundo y la Visión por Computador; campos en los que se fundamenta esta prueba de concepto. La Visión por Computador está revolucionando el análisis de imágenes dadas las capacidades de estos sistemas de interpretar datos visuales en tareas como la clasificación, segmentación semántica o la detección de objetos en la que se fundamenta este proyecto. Que fundamentadas en Aprendizaje Profundo, mediante el uso de Redes Neuronales, los algoritmos pueden identificar patrones complejos en las imágenes con una precisión igual o superior a la del ojo humano.

Dentro de este campo, la detección de objetos tiene como objeto localizar espacialmente (mediante *bounding boxes*) y clasificar cada una de las múltiples instancias de un objeto de interés dentro de una misma imagen. Esta tarea es crucial para este proyecto, pues va a permitir detectar y contar las células redondas en una imagen de esperma humano.

La familia de modelos de YOLO (You Only Look Once) [5] se ha consolidado como un referente para tareas de detección, permitiendo un equilibrio entre precisión y velocidad. Dentro de las 12 arquitecturas de YOLO [5], se emplean las siguientes [7]:

- **Yolov8:** Adopta un diseño eficiente y sin anclajes (anchor-free). Su arquitectura utiliza una versión refinada de módulos basados en CSP y una cabeza desacoplada que separa las tareas de clasificación y regresión, logrando un buen equilibrio entre velocidad y precisión [7].
- **Yolov9:** Introduce la Información de Gradiente Programable (PGI) y la Red de Agregación de Capas Generalizada y Eficiente (GELAN). Estas innovaciones ayudan a mantener gradientes robustos a través de la red, mejorando la convergencia y la capacidad para detectar objetos pequeños al evitar la pérdida de información [7].
- **Yolov10:** Elimina la necesidad de la supresión no máxima (NMS), un paso de postprocesamiento que consume tiempo, mediante una novedosa estrategia de asignación dual. Esto reduce significativamente la latencia de inferencia y lo hace más eficiente para la detección en tiempo real [7].

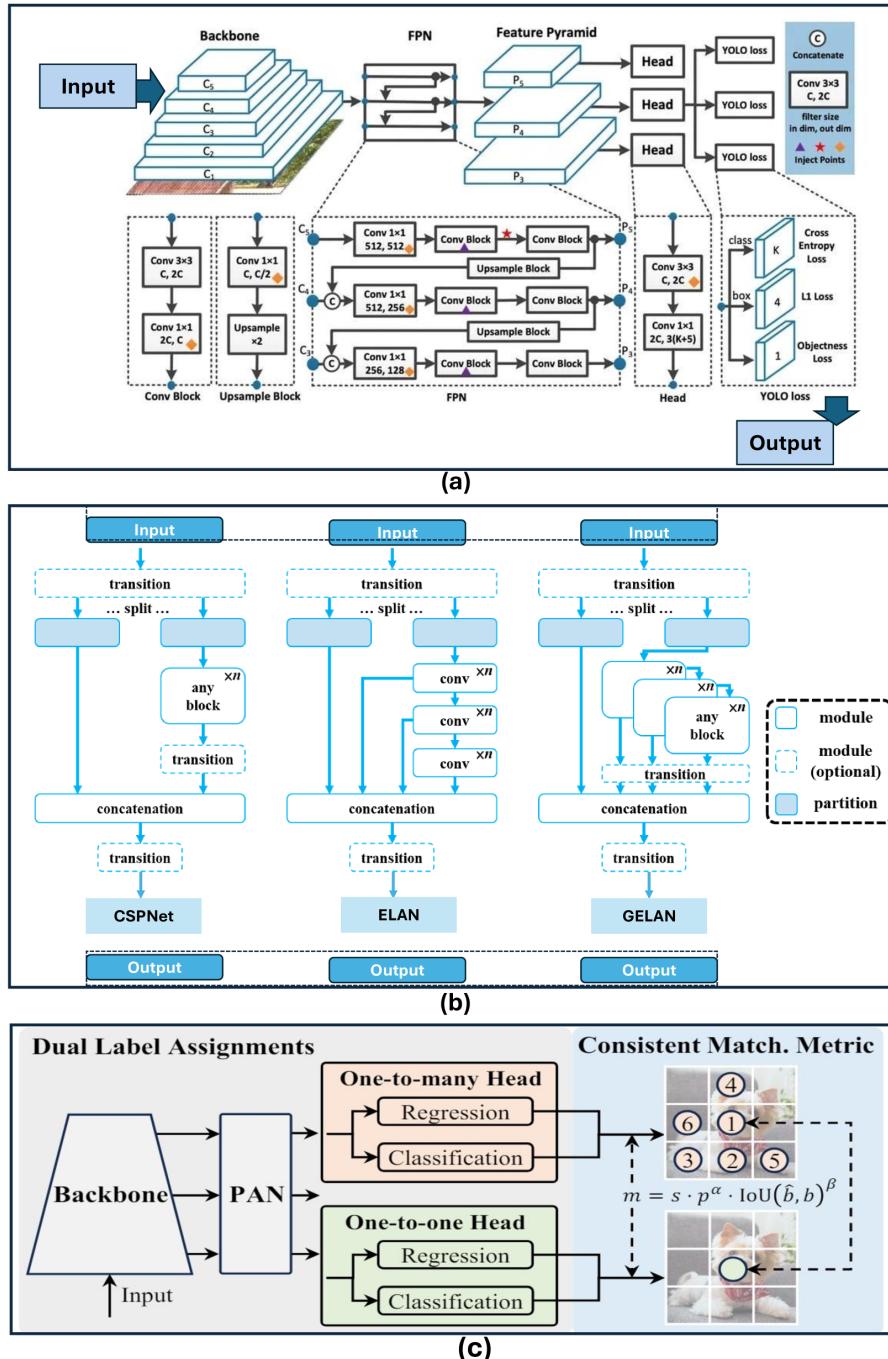


Figura 2.1: Diagramas de arquitectura de Yolov8, Yolov9 y Yolov10

(a) Yolov8 presenta una estructura principal basada en CSP, un cabezal desacoplado y libre de anclajes (anchor-free), y una red de pirámide de características (FPN) optimizada para una extracción eficiente de características a múltiples escalas. (b) Yolov9 integra la Información de Gradiente Programable con GELAN para una agregación robusta de características. (c) Yolov10 presenta una estrategia de asignación dual, cabezales ligeros y un submuestreo desacoplado de canal espacial para mejorar la precisión y velocidad de la inferencia.

- **Yolov11:** Se enfoca en mejorar la extracción de características mediante el bloque C3k2, el módulo Spatial Pyramid Pooling-Fast (SPPF) y la atención espacial paralela (C2PSA). Estas mejoras le permiten un rendimiento más preciso, especialmente en escenarios con objetos ocluidos [7].
- **Yolov12:** Su principal innovación es el módulo de Atención de Área (A^2), que ajusta dinámicamente el campo receptivo para capturar señales contextuales globales y locales con un costo computacional mínimo. Esto, junto con la red R-ELAN, optimiza la fusión de características y aumenta drásticamente la velocidad de inferencia [7].

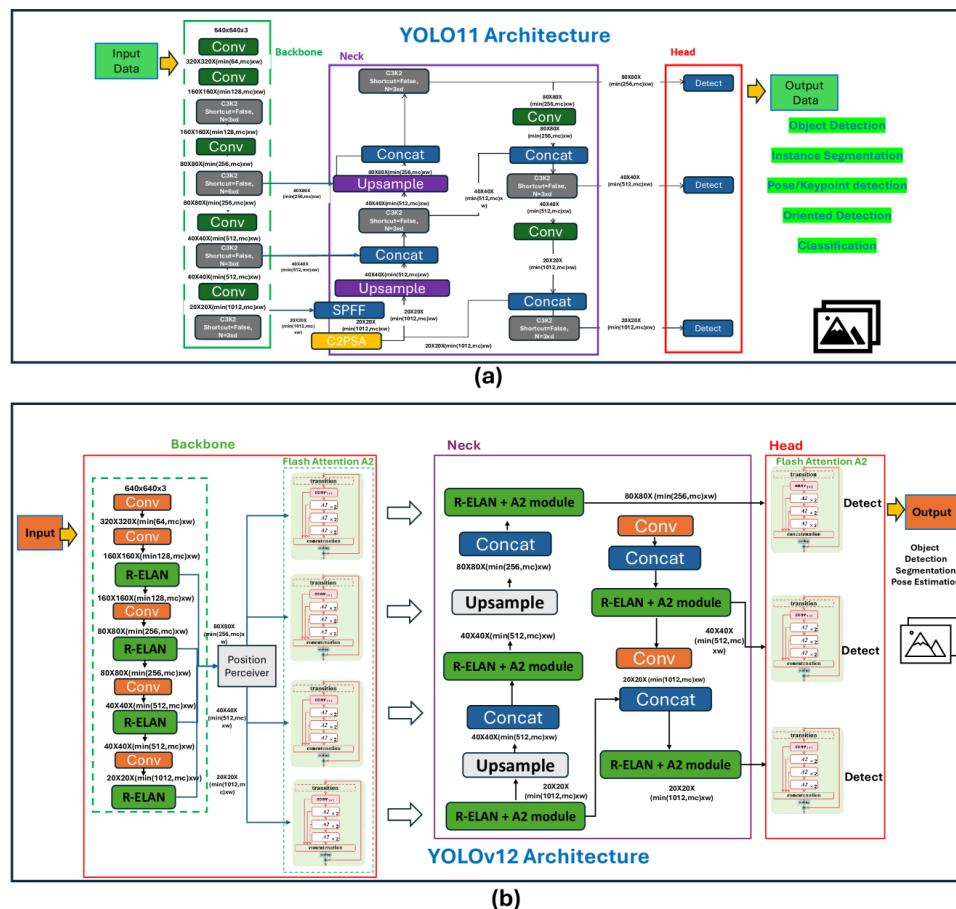


Figura 2.2: Diagramas de arquitectura de Yolov11 y Yolov12

(a) YOLOv11 utiliza una arquitectura mejorada que incluye bloques C3k2, SPPF y C2PSA. Estos componentes optimizan la extracción de características en diversas escalas y mejoran la atención espacial, lo que se traduce en una mayor precisión de detección. (b) YOLOv12 avanza sobre este diseño con una arquitectura centrada en la atención. Al integrar módulos de Área Attention y bloques R-ELAN, optimiza la combinación de características, logrando un aumento drástico en la velocidad de inferencia para una detección de objetos en tiempo real de vanguardia.

2.2.2. Técnicas de optimización y validación de modelos

Es bien sabido de la existencia de diferentes tecnologías para asegurar la robustez y maximizar el rendimiento de los modelos. Algunas de estas tecnologías, son las siguientes:

- **Optimización de hiperparámetros (Optuna):** El *framework* Optuna, de optimización automática de hiperparámetros (tales como la tasa de aprendizaje, el *momentum* o regulación) explora de manera inteligente y eficiente el espacio de búsqueda y, al final, devuelve la mejor combinación de hiperparámetros encontrada. Se fundamenta en algoritmos de muestreo bayesiano y mecanismos de poda (*pruning*) que descartan de manera anticipada los ensayos poco prometedores.
- **Aumento de datos (*data augmentation*):** Para mejorar la capacidad de generalización de un modelo y reducir el riesgo de sobreajuste (*overfitting*) en conjuntos de datos limitados se recurre a esta técnica. Esta técnica permite generar nuevas muestras de entrenamiento aplicando transformaciones geométricas (rotaciones, traslaciones, escalado, etc) y fotométricas (cambios de brillo, contraste, etc) sobre las imágenes originales, incrementando así la diversidad del conjunto de datos.
- **Validación cruzada (*cross validation*):** Dentro de sus múltiples variantes, la más común consiste en dividir el conjunto de datos en "k" subconjuntos o pliegues. El modelo se entrena K veces, utilizando en cada iteración un pliegue diferente para la validación y los K-1 restantes para el entrenamiento. El rendimiento final se calcula como la media de los resultados de las K iteraciones, proporcionando una medida de la capacidad de generalización del modelo menos dependiente de una única división de datos. Esta técnica permite conocer si un modelo es fiable y robusto.
- **Ensamblado de modelos (*ensemble*):** Esta técnica avanzada busca mejorar la precisión y la robustez de las predicciones combinando las salidas de múltiples modelos entrenados de forma independiente. La hipótesis subyacente es que los errores de un modelo pueden ser compensados por los aciertos de otros (al promediar o dar pesos sobre las predicciones individuales), especialmente si los modelos son diversos.

2.2.3. Literatura de la investigación

Esto se puede apreciar en aplicaciones como la medicina, donde podemos destacar ejemplos relevantes como:

- **Oncología:** Se han desarrollado modelos de Aprendizaje Profundo capaces de detectar cancer de mama en mamografías, detección de nódulos pulmonares en tomografías computarizadas y la identificación de piel a través de imágenes de lesiones cutáneas.
- **Neurología y Radiología:** Arquitecturas como U-Net son capaces de delinear con precisión tumores cerebrales en resonancias magnéticas.
- **Oftalmología:** Análisis de imágenes del fondo de ojo para detectar retinopatía diabética.

2.2.4. Hipótesis de trabajo

Considerando el potencial de las arquitecturas YOLO y las diferentes técnicas de optimización, se considera que mediante la exploración de las arquitecturas YOLO que van desde la versión 8 a la 12, junto con el diseño personalizado y la optima optimización de hiperparámetros, se consiga igualar o mejorar los modelos de la anterior prueba de concepto del grupo GVIS, en las métricas mAP y velocidad de inferencia. Adicionalmente, se espera que gracias a la arquitectura en desarrollo de Yolov12 basada en módulos de atención, el algoritmo no identifique como células redondas a los Anillos de Newton. la integración de estos modelos con la herramienta web interactiva validará su viabilidad como una solución en un marco de análisis real.

2.3. Definición del problema

La problemática que da origen a este proyecto no es otra que la ardua y extenuante tarea que le supone a un experto la revisión manual de imágenes médicas de muestras de semen.

Por esta razón, el problema central que aborda este proyecto es diseñar, desarrollar y evaluar un sistema inteligente capaz de automatizar la detección de células

redondas sobre un conjunto de imágenes médicas de muestras de semen, utilizando para ello, técnicas de Visión por Computador (*Computer Vision*) y Aprendizaje Profundo (*Deep Learning*).

Este sistema inteligente debe ser capaz de procesar imágenes microscópicas donde las células objetivo coexisten con una densa población de espermatozoides, diversos artefactos visuales (Anillos de Newton) y restos residuales ajenos al análisis. Asimismo, la dificultad que supone la distinción de células redondas de otros elementos morfológicamente similares (espermatozoides sin cola), esferas de las que se desconoce qué son, etc.

Para más inri, la asignación de anotaciones sobre el conjunto de imágenes es un problema que va a dar dolores de cabeza durante todo el proyecto. Por ejemplo, en la imagen 2.3 anotada por un experto, presenta la problemática de los Anillos de Newton. Se verá posteriormente como afectan estos objetos a la correcta identificación de células redondas.

Asimismo, en la imagen 2.4 se evidencian las constantes dudas a cerca de las anotaciones de células redondas, no solo por el personal que define el proyecto sino por los propios expertos de Microptic [2]. Cuyo *feedback* fue:

“La mayoría de células marcadas con los rectángulos verdes son ciertas células redondas. Los círculos rosas también serían células (probablemente germinales, pero se necesitaría hacer una evaluación morfológica específica para saberlo). Los círculos azules marcan correctamente algunas células redondas pero también indican espermatozoides que retienen la mayor parte del citoplasma (esferas grises con un área blanca bien contrastada). En cuanto a los círculos naranjas, muchos son cabezas de espermatozoides sin cola y las esferas más pequeñas desconocemos qué son.”

Esto evidencia, una vez más, el alto grado de complejidad que supone anotar este tipo de imágenes médicas, incluso bajo el marco de recomendaciones de la OMS [1] [8].

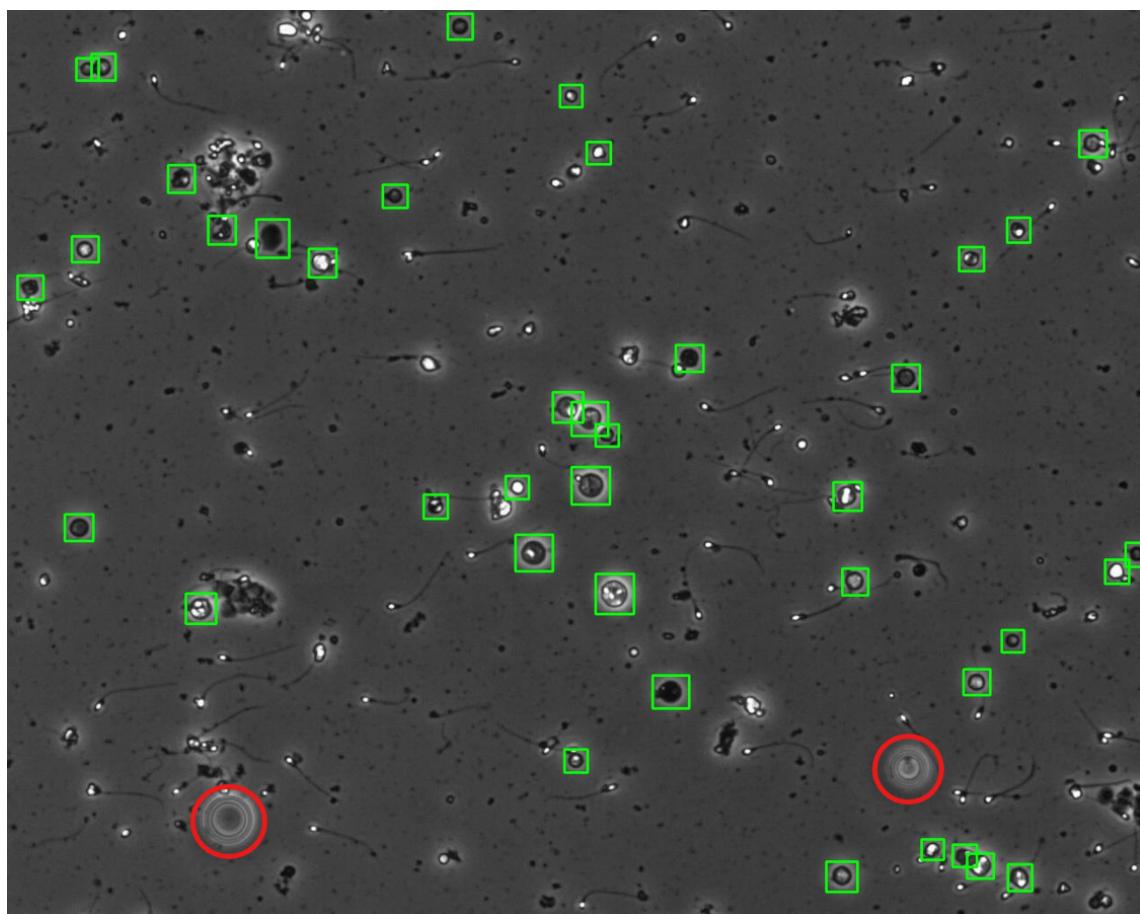


Figura 2.3: Ilustración con Anillos de Newton.

Las *bounding boxes* corresponden con las células redondas y los círculos rojos con los Anillos de Newton.

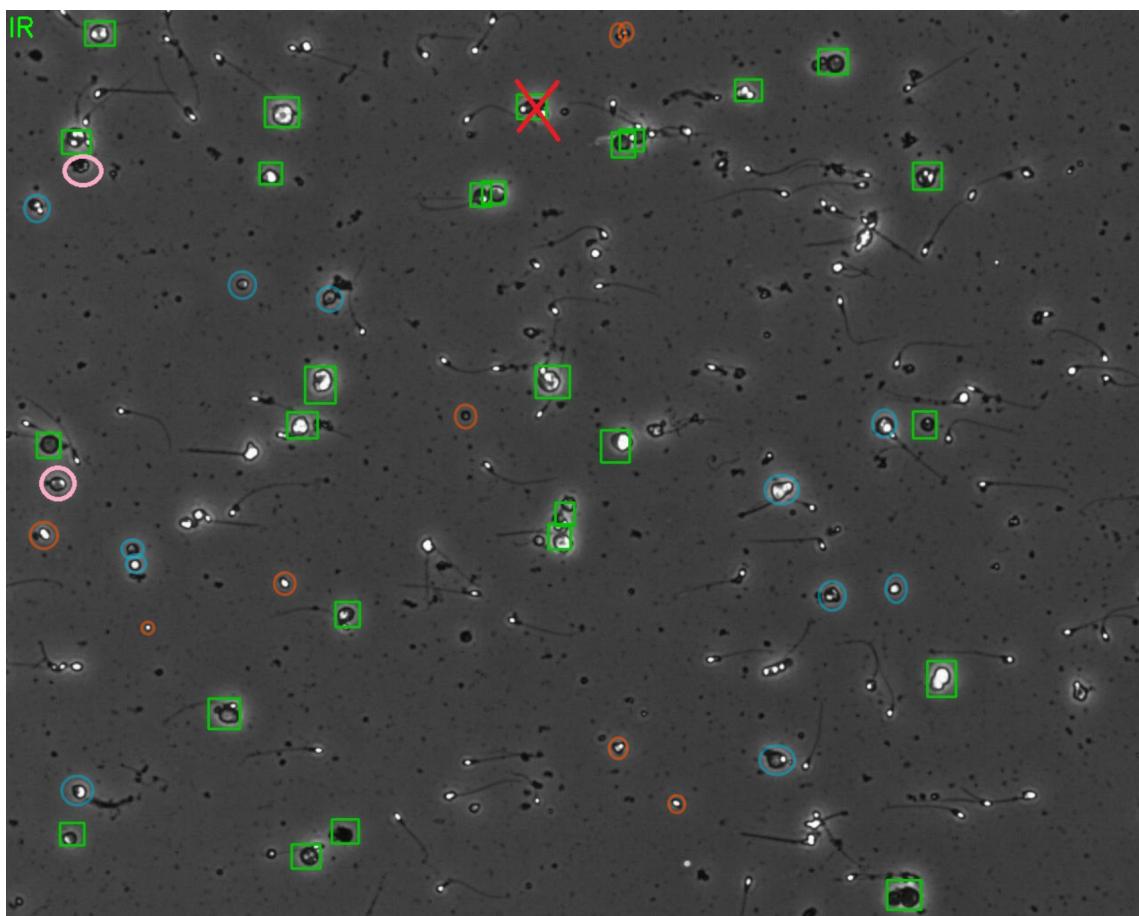


Figura 2.4: Ilustración de annotaciones para *feedback* con experto.

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 [9]**

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) [10]**

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 [11]**

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 se incorporan las siguientes medidas: documenta-

ción completa, evaluación de riesgos y validación, trazabilidad y registro para una posterior reproducibilidad.

■ **Real Decreto Legislativo 1/1996 sobre Propiedad Intelectual [12]**

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, optimización, validación de modelos YOLO (3) Ensamblado (Ensemble). (4) Modelo personalizado.

Aquí tengo que hablar de la arquitectura del código.

5. Experimentación

5.1. Dataset

El dataset inicial proporcionado por la empresa MICROPTIC [2] 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 [2] 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* [13] 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 [5] 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 [14].

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` [14], 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

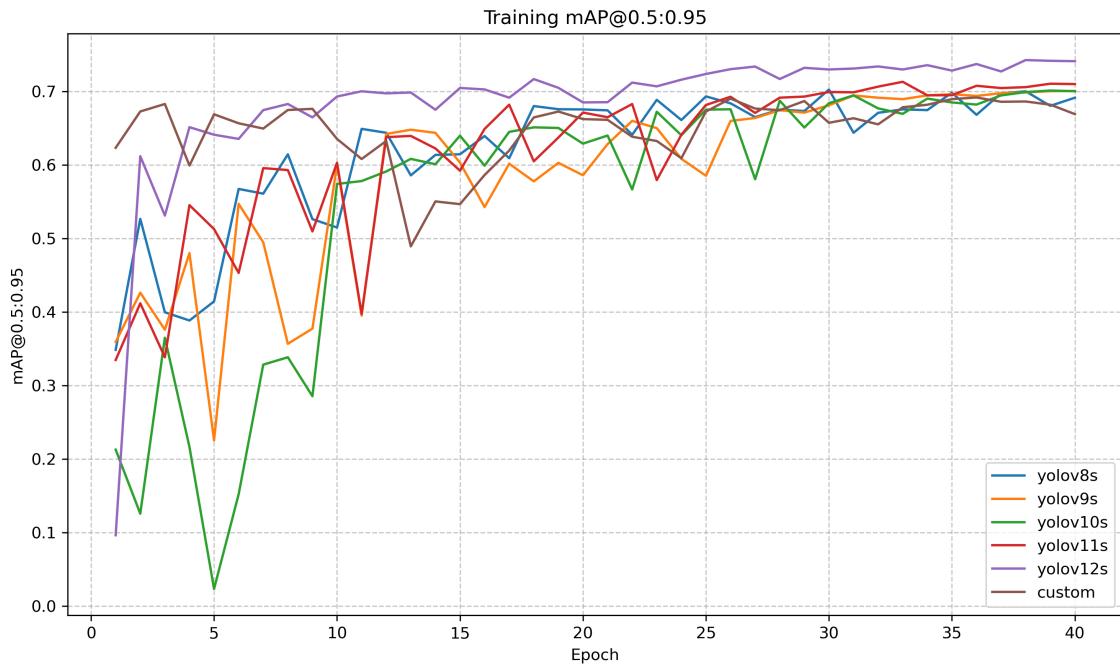


Figura 5.1: mAP@0.5:0.95 durante el proceso de entrenamiento.

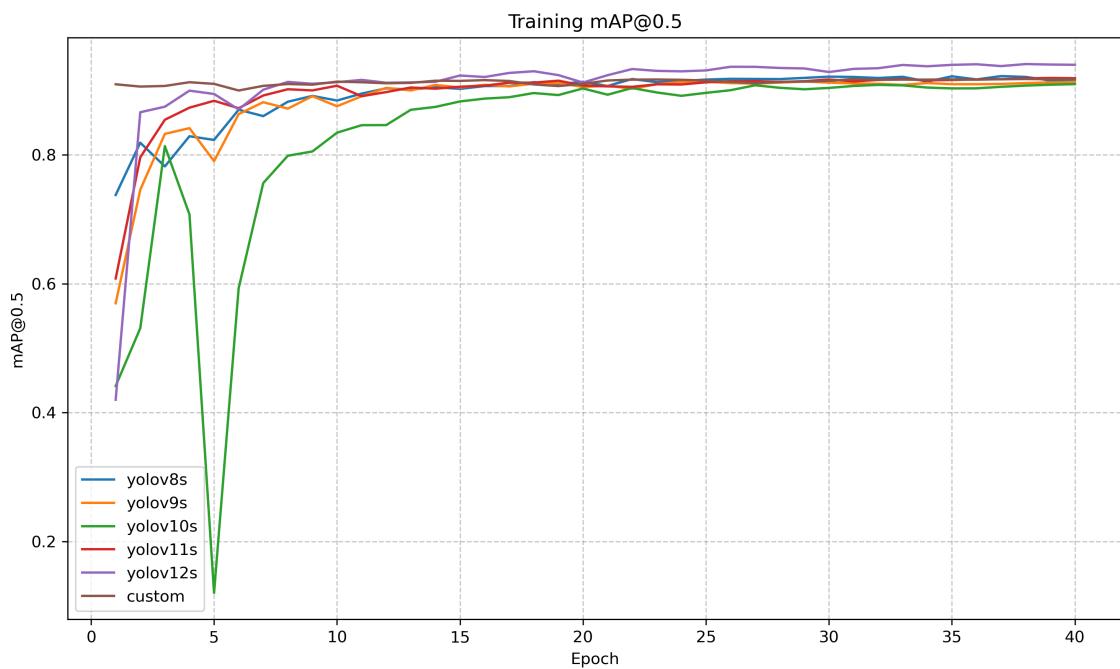


Figura 5.2: mAP@0.5 durante el proceso de entrenamiento.

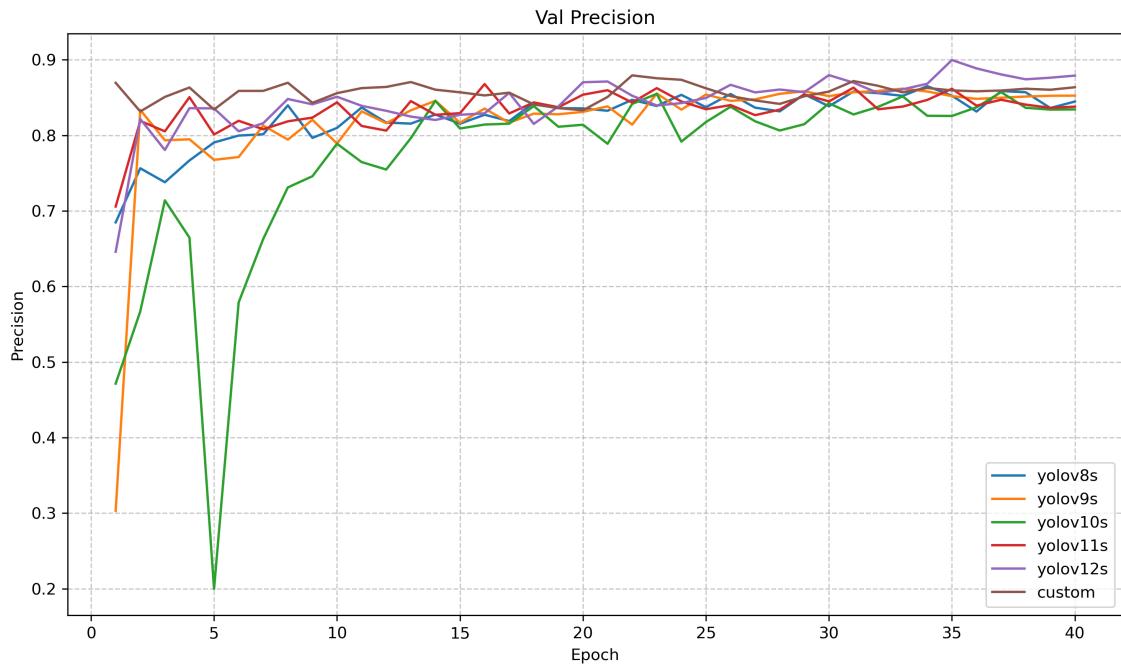


Figura 5.3: Precision del modelo sobre el conjunto de validación.

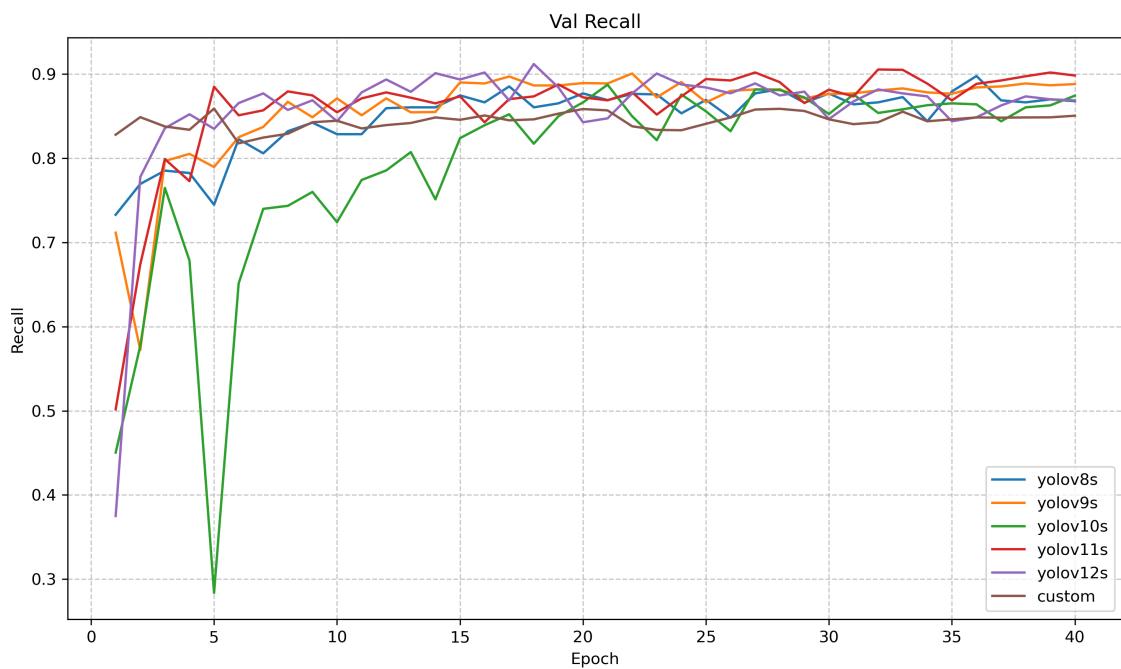


Figura 5.4: Recall del modelo sobre el conjunto de validación.



Figura 5.5: *Box Loss* sobre el conjunto de entrenamiento y validación.

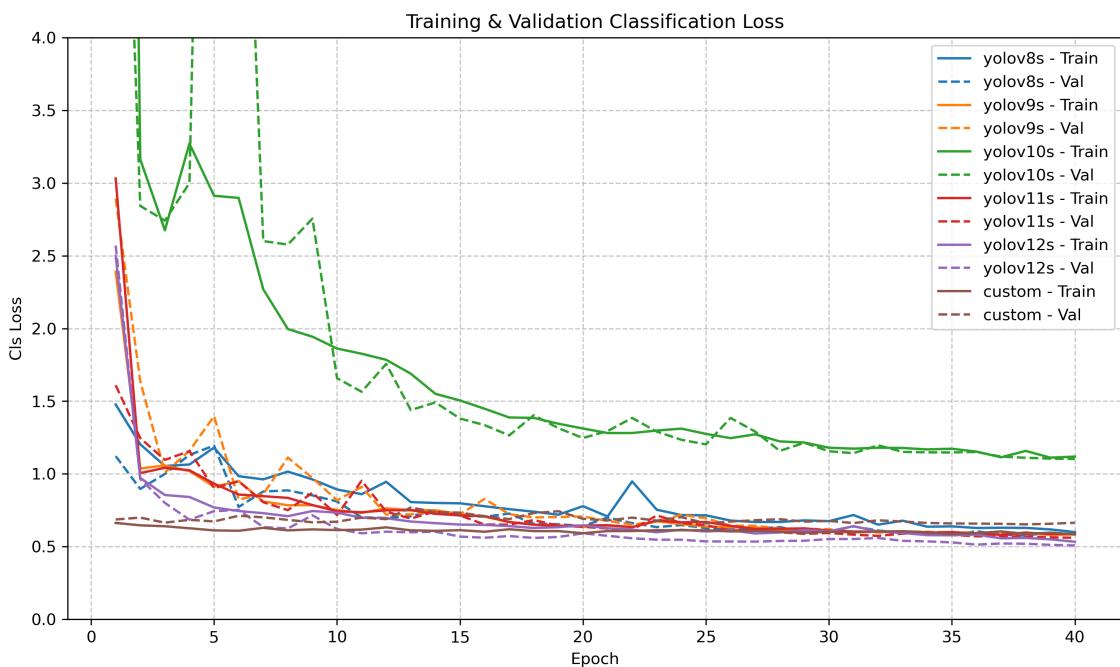


Figura 5.6: *Clasification Loss* sobre el conjunto de entrenamiento y validación.

5.3.2. Validación cruzada

5.3.3. Ensamblado de modelos

5.3.4. Modelo personalizado

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	0.76	14.615

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

Es importante que hablemos de esta problemática y como evitamos un reetiquetado de imágenes que pueden dar más confusión al confundir clases, y continuar la tendencia monoclas.

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 nuevos conjuntos de *test*

6.1.2. Ensemble

6.1.3. 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 [14].

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

hacer una evaluación de riesgos. Por lo tanto, este Trabajo de Fin de Máster no es un ejercicio puramente académico, sino que se constituye como una Prueba de Concepto (PoC) con un objetivo industrial definido y un respaldo institucional estratégico.

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] 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.
- [2] Microptic S.L. <https://www.micropticsl.com/>.
- [3] Hamilton Thorne. Advanced reproductive technologies. <https://www.hamiltonthorne.com/>.
- [4] DIGIS3, Digitalización Inteligente, Sostenible y Cohesiva. <https://digis3.eu/es>.
- [5] Ultralytics. Ultralytics models documentation. <https://docs.ultralytics.com/es/models/>.
- [6] Hamilton Thorne. Relevance of round cells detection in spermograms. <https://www.hamiltonthorne.com/relevance-of-round-cells-detection-in-spermograms/>, 2024.
- [7] Ranjan Sapkota, Zhichao Meng, Martin Churuvija, Xiaoqiang Du, Zenghong Ma, and Manoj Karkee. Comprehensive performance evaluation of yolov12, yolo11, yolov10, yolov9 and yolov8 on detecting and counting fruitlet in complex orchard environments. *arXiv preprint arXiv:2407.12040v7*, 2025.
- [8] 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.
- [9] 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>.
- [10] 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>.

- [11] 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>.
- [12] 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>.
- [13] labelImg: Graphical image annotation tool. <https://github.com/tzutalin/labelImg>.
- [14] Aitor García Blanco. Repositorio del TFM: Detección de células defectuosas en imágenes médicas. <https://github.com/AigarciabFabero/TFM>, 2025.
- [15] 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.
- [16] 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.
- [17] Optuna Development Team. Optuna: Automl and hyperparameter optimization framework. <https://github.com/optuna/optuna>, 2025. Accessed: 2025-08-17.
- [18] Online gantt — gantt chart maker. <https://www.onlinegantt.com/>.

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 [14].

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.

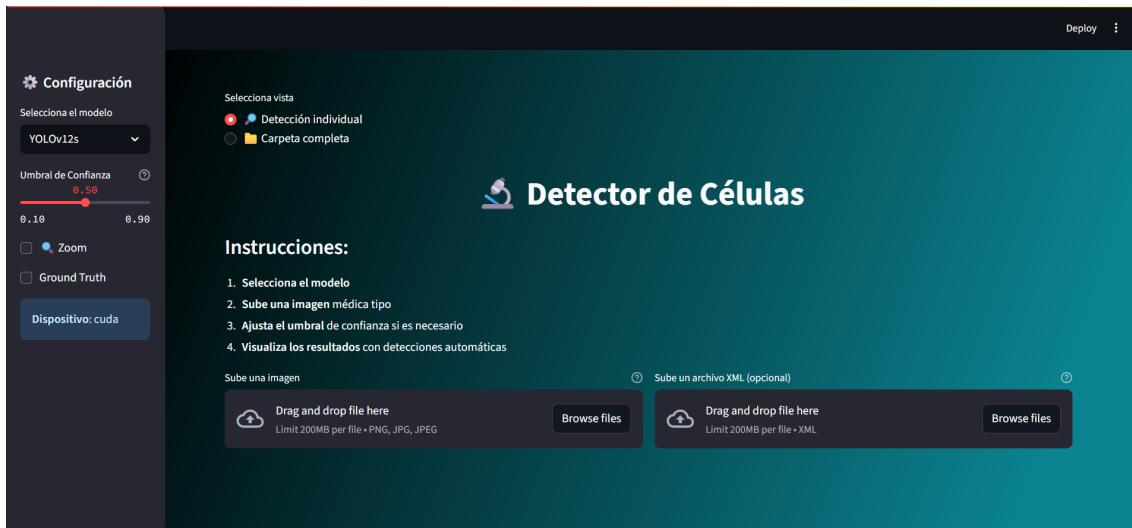


Figura C.1: Interfaz de la aplicación web.

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.

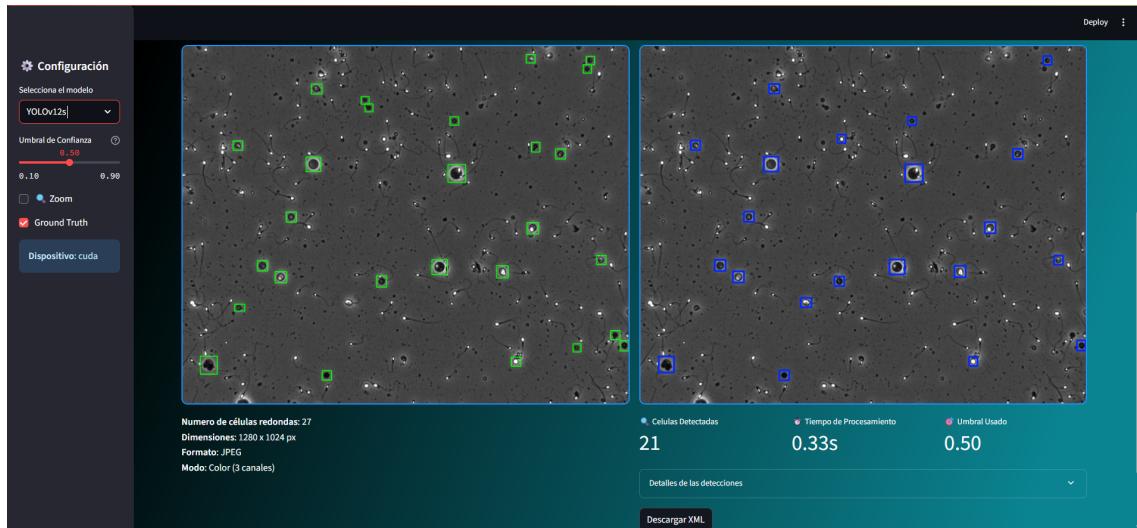


Figura C.2: Herramienta web: *checkbox ground truth*.



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.

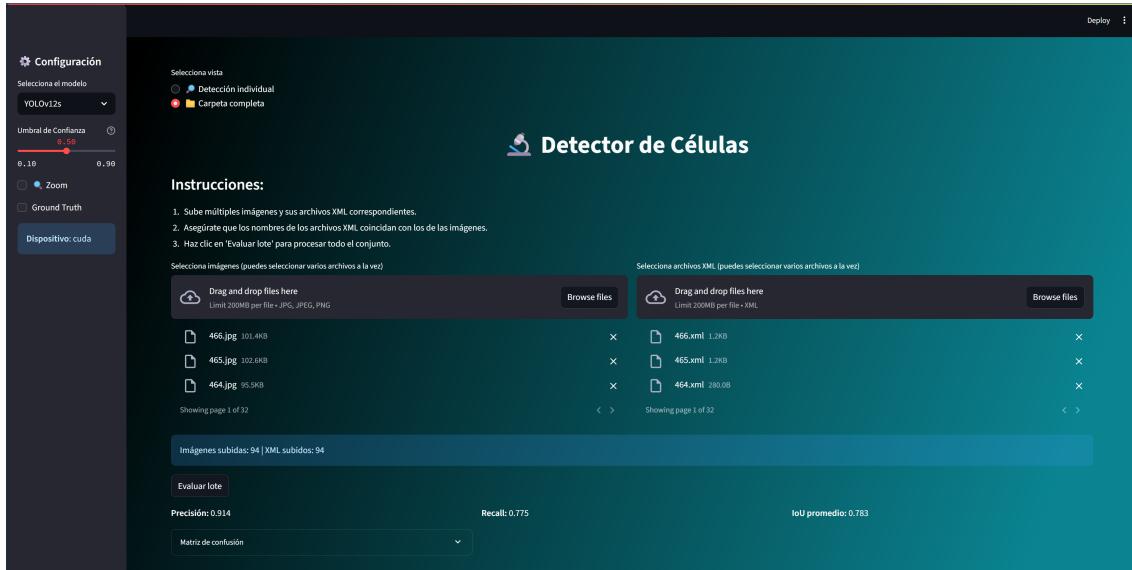


Figura C.4: Herramienta web para procesar un conjunto de imágenes.

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).

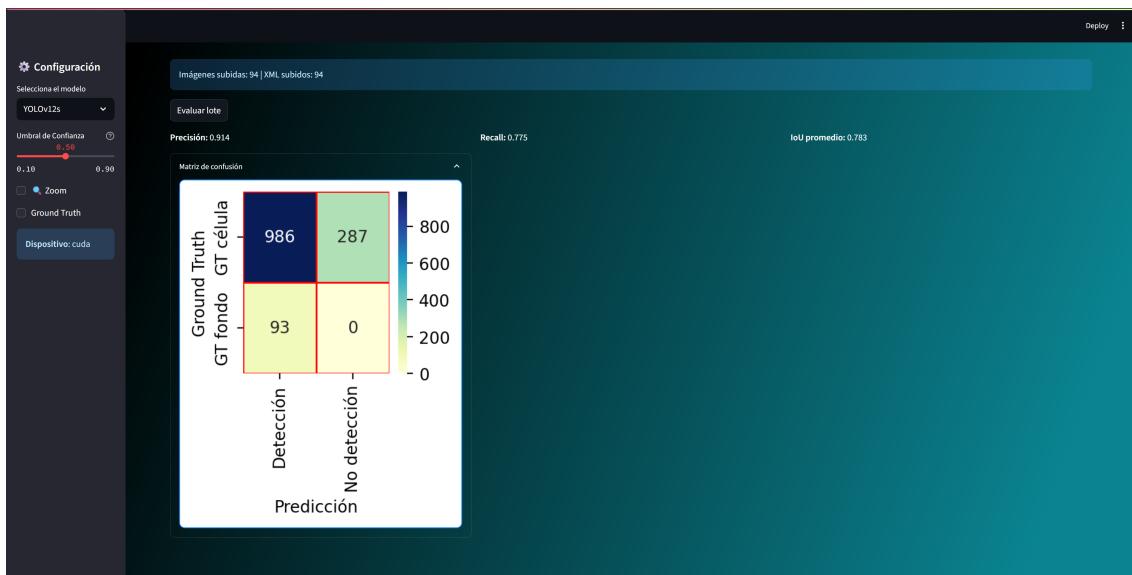


Figura C.5: Herramienta web: métricas y matriz de confusión.

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 [5].

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.