YOLOv8 vs YOLOv11: revisión sistemática y validación experimental en detección de objetos

Autores
Luis David Izquierdo Serrano
Carlos David Rodriguez Quintero
Jhon Arley Forero Velasco

Título por el que opta
Ingeniero de sistemas

Directores
Andrés Felipe Arboleda Duque
Víctor Viera Balanta

Grupo de Investigación
Comba I+D

Línea de Investigación
Desarrollo De Sistemas Informáticos

Facultad de Ingeniería
Ingeniería de sistemas
Universidad Santiago de Cali
Santiago de Cali - Colombia
2026

YOLOv8 vs YOLOv11: revisión sistemática y validación experimental en detección de objetos

YOLOv8 vs YOLOv11: Systematic Review and Experimental Validation in Object Detection

Luis David Izquierdo Serrano[1]
luis.izquierdo02@usc.edu.co

Jhon Arley Forero Velasco[1]
jhon.forero01@usc.edu.co

Carlos David Rodriguez Quintero[1]
carlos.rodriguez21@usc.edu.co

Andrés Felipe Arboleda Duque[2]
dirtecdesis@usc.edu.co

Víctor Viera Balanta[1]
victor.viera00@usc.edu.co

Universidad Santiago de Cali, Facultad de Ingeniería, Programa de Ingeniería de Sistemas (1)
Universidad Santiago de Cali, Facultad de Ingeniería, Programa de Tecnología en Desarrollo de Sistemas de Información y de Software (2)

Resumen
Este artículo presenta un análisis comparativo del desarrollo y desempeño de YOLOv8 y YOLOv11, dos versiones recientes del algoritmo de detección en tiempo real "You Only Look Once" (YOLO), reconocido por su velocidad y precisión. El enfoque principal es analizar y comparar los resultados de las diferentes métricas en ambas versiones utilizando distintos datasets. Para ello, se consideran múltiples variantes del modelo ("n", "s" y "m"), para evaluar su comportamiento bajo diferentes niveles de capacidad. Asimismo, el estudio se fundamenta en principios teóricos del conocimiento libre, la optimización computacional y las licencias abiertas, que respaldan el uso de código abierto en entornos académicos y científicos. Se adopta una metodología documental y experimental, aplicando PRISMA para la revisión de literatura y el marco OSEMN para la preparación, modelado y evaluación. En esa misma línea se utilizaron métricas estandarizadas como mAP@0.5, mAP@0.5-0.95, junto con indicadores como el tiempo de entrenamiento. Los resultados muestran que el rendimiento de cada versión depende del dataset y del tamaño del modelo. YOLOv8 obtuvo mayor precisión y mAP en escenarios como African Wildlife y PI3final, mientras que en YOLOv11 destaco por su eficiencia computacional y tiempos de entrenamiento en las variantes n y s. En conclusión, YOLOv8 destaca por su exactitud y robustez, mientras que YOLOv11 resulta más eficiente en hardware limitado, ambas versiones contribuyen al avance académico sobre la evolución de YOLO en visión por computador.

Palabras Clave: detección de objetos, yolov11, yolov8, precisión, ultralytics, rendimiento

Abstract
This article presents a comparative analysis of the development and performance of YOLOv8 and YOLOv11, two recent versions of the "You Only Look Once" (YOLO) real-time detection algorithm, known for its speed and accuracy. The main focus is to analyze and compare the results of different metrics in both versions using various datasets. To this end, multiple model variants ("n", "s", and "m") are considered to evaluate their behavior under different capacity levels. Furthermore, the study is founded on theoretical principles of free knowledge, computational optimization, and open licenses, which support the use of open source code in academic and scientific environments. A documentary and experimental methodology is adopted, applying PRISMA for literature review and the OSEMN framework for preparation, modeling, and evaluation. Along the same lines, standardized metrics such as mAP@0.5 and mAP@0.5-0.95 were used, along with indicators like training time. The results show that the performance of each version depends on the dataset and the model size. YOLOv8 achieved greater precision and mAP in scenarios such as African Wildlife and PI3final, while YOLOv11 stood out for its computational efficiency and training times in the 'n' and 's' variants. In conclusion, YOLOv8 excels in accuracy and robustness, while YOLOv11 proves more efficient on limited hardware; both versions contribute to academic progress on the evolution of YOLO's in computer vision.

Keywords: object detection, yolov11, yolov8, accuracy, ultralytics, performance

1. INTRODUCCIÓN

La detección de objetos ha avanzado significativamente con la familia de algoritmos YOLO (You Only Look Once), denominada así porque el modelo analiza toda la imagen en una sola pasada para identificar y localizar múltiples objetos simultáneamente, lo que permite una detección en tiempo real. Esta familia se ha consolidado como referente en visión por computador, destacando por su equilibrio entre precisión y velocidad. En este desarrollo, Ultralytics ha desempeñado un papel clave con versiones como YOLOv8 y YOLOv11, que amplían las capacidades del modelo al incluir tareas de segmentación y clasificación.

La visión por computador posee una creciente relevancia económica y tecnológica, con un mercado que, según Fortune Business Insights (2025), se proyecta en 58,33 mil millones de dólares para 2032, lo que evidencia su impacto global. Además, su carácter disruptivo se refleja en múltiples sectores: en salud, impulsa diagnósticos más rápidos y precisos en áreas como radiología, endoscopia, patología y análisis de lesiones pulmonares (Palaniappan et al., 2025); mientras que en el sector industrial, optimiza procesos mediante la monitorización, la seguridad automatizada, la conducción autónoma y la producción inteligente (Kang et al., 2025).

1.1 Marco Teórico

1.1.1 Teoría del conocimiento libre y bienes comunes digitales

La investigación se fundamenta en los principios del software libre (Stallman & Lessig, 2004) y la cultura libre (Lessig, 2002), que promueven el acceso abierto y la colaboración. Esto se refleja en el uso de herramientas open source, como los modelo YOLO y repositorios de datos como RoboFlow, facilitando la replicación en entornos de investigación.

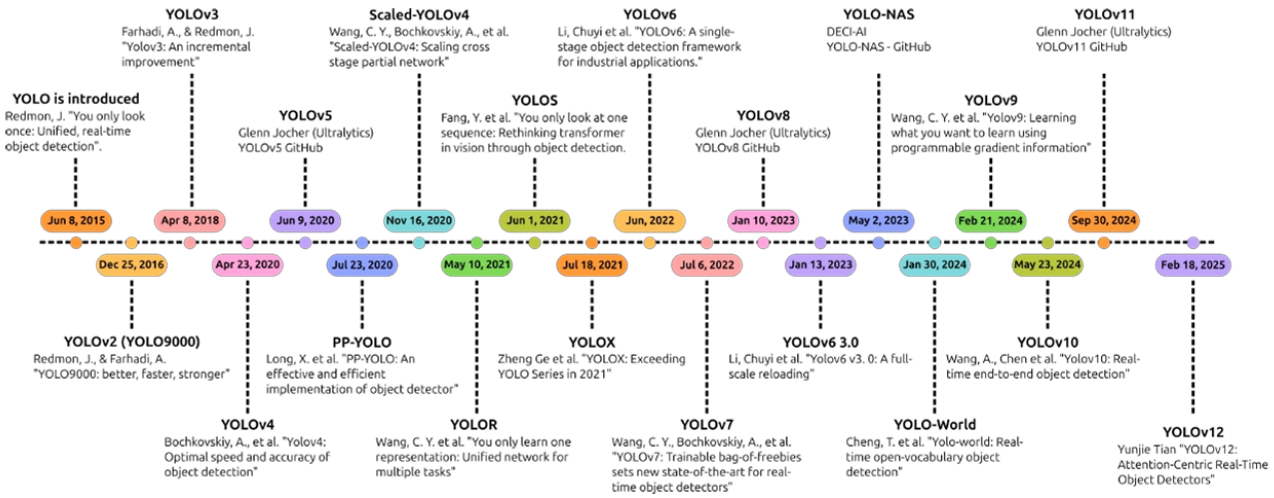
1.1.2 Teoría de la optimización

La teoría de la optimización proporciona herramientas para tomar decisiones óptimas bajo restricciones, mediante la maximización o minimización de una función objetivo (Journal of Network and Computer Applications, s. f.).

1.1.3 Historia y evolución

YOLO es un algoritmo de detección de objetos que, en una única evaluación, predice recuadros delimitadores y las probabilidades de clase con rendimiento en tiempo real. Presentado por Redmon et al (2016), revolucionó la visión por computadora al unificar localización y clasificación en un solo modelo, acelerando la inferencia sin sacrificar precisión. Ha evolucionado en múltiples versiones que muestran mejoras arquitectónicas progresivas (Figura 1), manteniendo un equilibrio destacado entre velocidad y exactitud.

Figura 1. Evolución de los algoritmos YOLO



Nota. La figura muestra la evolución de YOLO a lo largo de los años. Fuente: Jegham (2025)

1.2 Marco Conceptual

1.2.1 Visión por computador

La visión por computador es una disciplina de la inteligencia artificial que busca proporcionar a las máquinas la capacidad de interpretar el mundo visual a partir de imágenes o secuencias de video. Szeliski (2010) señala que los humanos perciben con facilidad su entorno, identifican objetivos y reconocen emociones a partir de rasgos faciales, capacidades que han resultado desafiantes de replicar computacionalmente. Bradski & Kaehler (2011) definen como la transformación de datos visuales en información útil, como decisiones o nuevas representaciones. En los últimos años, su desarrollo se ha acelerado gracias a las redes neuronales, que permiten el reconocimiento, la localización y la percepción visual en tiempo real (Gionfrida et al., 2024).

1.2.2 Detección de objetos

La detección de objetos es una tarea clave en visión por computador que localiza y clasifica objetos mediante bounding boxes. Inicialmente se empleaban métodos basados en subventanas que exploran todas las posibles posiciones, resultando ineficiente por su alto costo computacional (Szeliski, 2010). Para superar estas limitaciones surgieron detectores especializados como YOLO. Bayoudh et al. (2022) señalan que la redes convolucionales han permitido integrar eficientemente localización con clasificación, superando las restricciones computacionales de los métodos tradicionales.

1.2.3 Arquitecturas

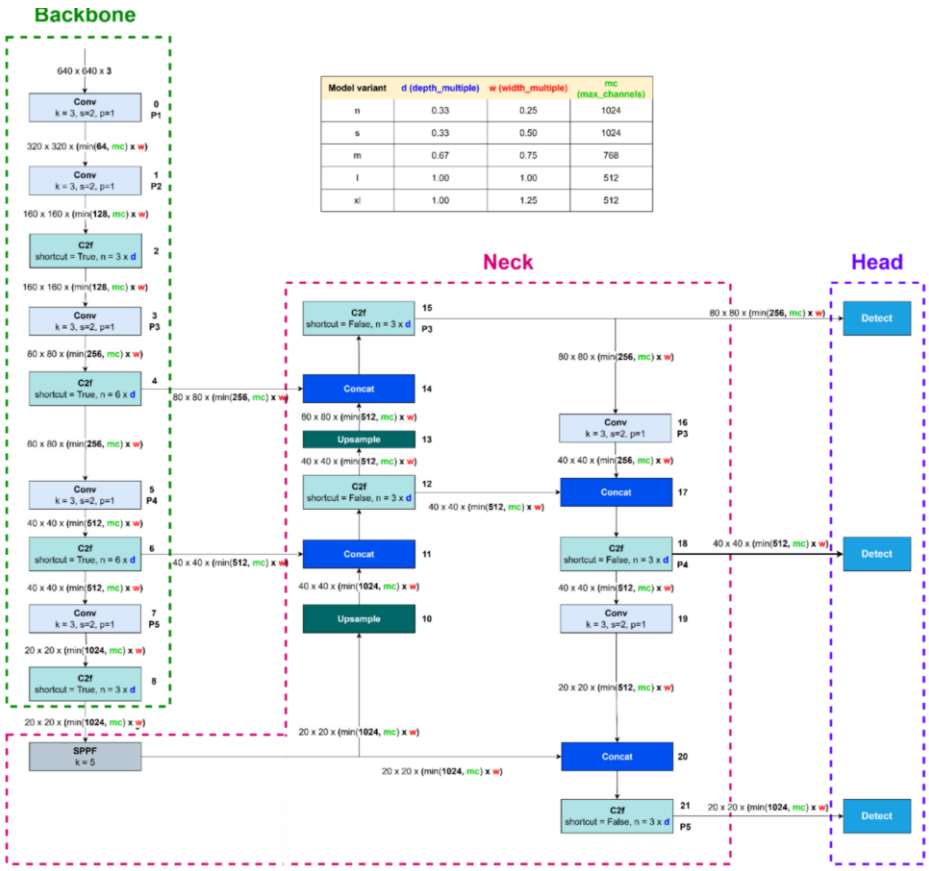
Cabe señalar que Ultralytics no ofrece documentación técnica detallada sobre las arquitecturas internas de sus modelos, ya que prioriza la rápida iteración y el lanzamiento de nuevas versiones sobre la publicación formal de dicha información (Ultralytics, 2023, 2024).

Arquitectura YOLOv8

YOLOv8, desarrollado por Ultralytics, destaca por su rapidez y precisión en detección de objetos en tiempo real. Hidayatullah et al. (2025) señalan que su módulo SPPF permite captar detalles a diferentes escalas, mejorando el reconocimiento de objetos de distintos tamaños. Su diseño ligero incluye un head desacoplado y arquitectura sin anchors, permitiendo su ejecución en dispositivos con recursos limitados y adaptación a

distintas necesidades de velocidad y precisión (Murat & Kiran, 2025). Ramos y Sappa (2025) destacan el módulo C2F y técnicas de entrenamiento optimizadas que aumentan la capacidad de generalización en escenarios complejos. Baglat et al. (2025) resaltan que el C2f proporciona fusión contextual enriquecida, mejorando la separación de objetos superpuestos en entornos reales. La Figura 2 presenta el diseño arquitectónico de YOLOv8.

Figura 2. Arquitectura de YOLOv8

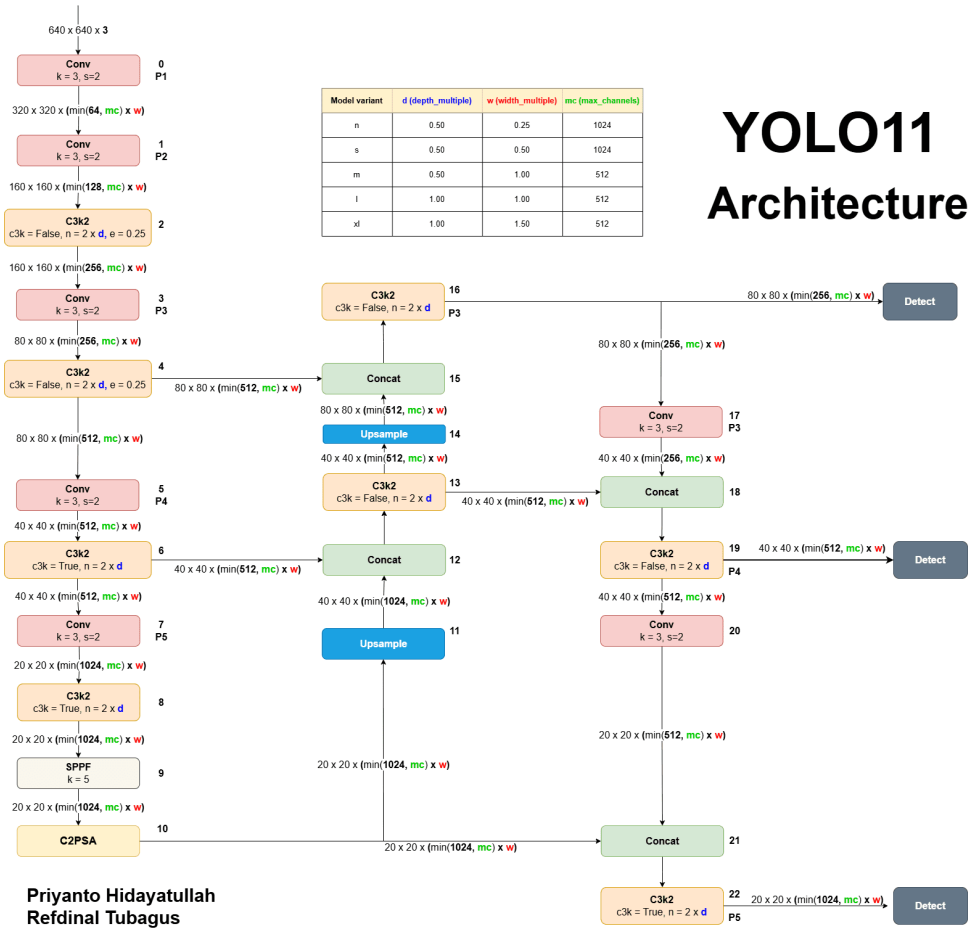


Nota. La figura ilustra la arquitectura empleada por la versión 8 del modelo YOLO. Fuente: Hidayatullah (2025)

Arquitectura YOLOv11

YOLOv11 es una de las versiones más reciente de la serie YOLO, manteniendo la estructura backbone-neck-head con innovaciones significativas. Hidayatullah et al. (2025) señalan que el bloque C3k2 evoluciona del C2f de YOLOv8, utilizando menos parámetros para mayor eficiencia computacional, mientras que Baglat et al. (2025) destacan que el C2PSA mejora la detección de objetos pequeños o superpuestos sin sacrificar precisión. Murat y Kiran (2025) indican que YOLOv11 ha amplía sus capacidades incorporando tareas como detección de orientación y optimizando el desempeño en escenarios complejos como vehículos autónomos, imágenes satelitales y drones. Ramos y Sappa (2025), enfatizan que conserva la detección multiescala y la versatilidad de sus variantes. La Figura 3 muestra el diseño arquitectónico de YOLOv11, mientras que la Tabla 1 compara los bloques arquitectónicos de YOLOv8 y YOLOv11.

Figura 3. Arquitectura de YOLOv11



Nota. La figura ilustra la arquitectura empleada por la versión 11 del modelo YOLO. Fuente: Ghost (2024)

Tabla 1. Comparación de la arquitectura entre YOLOv8 y YOLOv11

Bloque	Función General	YOLOv8	YOLOv11
Backbone	Extrae características visuales de la imagen para crear un mapa de características.	Bloque C2f, versión optimizada del CSP Bottleneck para extracción eficiente con redundancia reducida (Jocher et al., 2023).	Bloque C3k2, evolución del C2f con kernel más pequeño (k=2) que captura detalles finos, mejorando eficiencia y precisión en objetos pequeños (Ultralytics, 2024).
Neck	Combina características de diferentes escalas para enriquecer la información.	PAN-FPN fusiona información bidireccional para mejorar localización de objetos de múltiples tamaños(Jocher et al., 2023).	PANet mejorada con módulo C2PSA y SPPF. El C2PSA enfoca características relevantes, optimizando detección en ambientes complejos con objetos pequeños (Ultralytics, 2024).
Head	Interpreta características enriquecidas para producir predicciones finales.	Anchor-free, simplificando el modelo y mejorando flexibilidad para detectar objetos de diversas formas y tamaños(Jocher et al., 2023).	Anchor-free optimizado para eficiencia y precisión en múltiples tareas: detección, segmentación y estimación de poses(Ultralytics, 2024).
Enfoque	Objetivo de diseño arquitectónico.	Balance entre velocidad y precisión para aplicaciones versátiles.	Máxima precisión con mayor eficiencia, especialmente en detección de objetos pequeños.

Nota. Datos tomados y adaptados de Jocher et al. (2023) y Ultralytics (2024).

1.2.4. Métricas de evaluación

Para garantizar una evaluación equitativa, se emplearon métricas rigurosas como la precisión promedio (AP), la precisión media promedio (mAP) y la puntuación F1. El cálculo de estas métricas se basa en conceptos claves: Verdaderos Positivos (VP) representan detecciones correctas de objetos, Falsos Positivos (FP) corresponden a detecciones incorrectas y los Falsos Negativos (FN) objetos no detectados. Los Verdaderos Negativos (VN) no se consideran debido a la cantidad de regiones sin objetos en una imagen (Padilla et al., 2020).

Principales métricas utilizadas.

Según Derczynski (2016):

- Precisión: Mide qué tan correctas son las predicciones positivas del modelo (De todo lo del sistema dijo que era correcto, ¿cuánto real lo es?).
- Recall: mide la proporción de elementos positivos correctamente identificados por el modelo respecto al total de elementos positivos reales.
- F1-Score: medida armónica entre precisión y recall, que equilibra ambas métricas (Balance entre precisión y Recall).

Según Ultralytics (s. f.-b):

- mAP: Métrica promedio de la precisión media en todas las clases, usado para evaluar el desempeño general de un modelo de detección de objetos.
- mAP50: Precisión media promedio calculada con un umbral de coincidencia del 50%, que refleja el rendimiento en casos más sencillos de detección.
- mAP50-95: Promedio de la precisión media en distintos umbrales de coincidencia entre 50% y 95%, lo que ofrece una visión más completa del rendimiento del modelo en detecciones fáciles y difíciles.

1.3. Marco de licenciamiento

Los proyectos de Ultralytics están licenciados bajo AGPL-3.0 (Affero General Public License v3), garantizando acceso al código fuente incluso en implementaciones como servicio en línea (Free Software Foundation, 2007; Ultralytics, s. f.-a). Los conjuntos de datos de Roboflow están licenciados bajo CC BY 4.0 (Creative Commons Attribution 4.0 International), permitiendo copiar, distribuir y adaptar el material con atribución (Creative Commons, s. f.).

1.4. Marco Tecnológico

El desarrollo experimental se realizó en entornos locales de cómputo de gama media-baja y mediante Google Colab, plataforma que ofrece acceso gratuito a GPUs y facilita el uso de datasets (Bisong, 2019). Para la implementación, se emplearon librerías oficiales de Ultralytics YOLO, que permiten entrenar y evaluar modelos como YOLOv8 y YOLOv11 de manera estandarizada, proporcionando un marco optimizado para detección de objetos (Jocher et al., 2022). Los modelos, distribuidos con pesos preentrenados en el dataset COCO (Tsung-Yi et al., s. f.), fueron ajustados a las necesidades del estudio. El procesamiento se aceleró mediante CUDA, plataforma de NVIDIA que reduce significativamente los tiempos de entrenamiento y hace viable la detección en tiempo real (NVIDIA Corporation, s. f.) La Tabla 2 detalla las especificaciones de hardware y software del entorno experimental.

Tabla 2. Parámetros de hardware y software del entorno experimental

Componente	Especificación
Equipo	Lenovo IdeaPad Gaming 3 15IAH7 (Tipo 82S9)
Sistema operativo	Windows 11 Home Single Language, versión 24H2 (compilación 26100.3775)
Memoria RAM	8 GB DDR4, 3200 MHz
Procesador	Intel Core i5-12450H, 12 núcleos (~2.0 GHz)
Almacenamiento	512 GB NVMe
GPU	NVIDIA GeForce RTX 3050 Laptop, 4 GB VRAM, 2048 núcleos CUDA
Lenguaje	Python 3.12.6
Aceleración GPU	CUDA 12.1
Framework	Ultralytics YOLO v8.3.151

Nota. La tabla presenta las especificaciones técnicas del entorno empleado para el entrenamiento, validación y prueba de YOLOv8 y YOLOv11

1.5. Estado del arte

Desde su creación, YOLO ha evolucionado de forma constante, orientada a mejorar la precisión, la velocidad de inferencia y la eficiencia computacional, destacándose sus últimas versiones YOLOv8 y YOLOv11.

Jegham et al. (2025) evaluaron YOLOv3 a YOLOv12 en tres conjuntos de datos diversos (señales de tránsito, fauna africana y barcos), con el fin de analizar su robustez frente a variaciones de tamaño, densidad y orientación de objetos. El estudio realizado en un entorno de alto rendimiento con dos GPUs NVIDIA RTX 4090, evidencio que YOLOv11 ofreció un equilibrio sobresaliente entre precisión y eficiencia, mientras que YOLOv9 destaco en datasets pequeños por su arquitectura ligera. En conclusión, los autores señalaron que la elección del modelo depende de las características del problema y las limitaciones del hardware disponible.

Raj et al. (2025) evaluaron el rendimiento de distintas versiones de YOLO (v5 a v11) y YOLO-NAS en el reconocimiento de Lenguaje de Señas de la India, utilizando un conjunto de datos con 115 clases capturadas en condiciones controladas. Los modelos fueron entrenados en una GPU NVIDIA RTX 4090, aplicando 50 épocas para YOLO NAS y 100 para YOLOv5-11. Los resultados revelaron que YOLOv11 obtuvo un mAP@0.50 de 0.978, mientras que YOLO-NAS en su versión "m" lo superó ligeramente con un valor de mAP@0.50, destacándose ambos en precisión, F1-score y recall, lo que lo posiciona como alternativas prometedoras para aplicaciones en tiempo real.

Sharma et al. (2024) compararon YOLOv8 a YOLOv11 con Faster R-CNN en la detección de malas hierbas, empleando un conjunto de 2348 imágenes de campo y técnicas de aumento de datos. Los modelos se entrenaron en Google Colab con una GPU NVIDIA L4 durante 100 épocas. Los resultados mostraron que YOLOv9 obtuvo la mayor precisión promedio (mAP@0.5 = 0.935), mientras que YOLOv11 fue el modelo más rápido con un tiempo de inferencia de 13.5 milisegundos.

Jiang y Zhong (2025) compararon YOLOv5 a YOLOv11 por medio de 33 conjuntos de datos de 11 dominios distintos, entrenados bajo condiciones estandarizadas en una GPU NVIDIA A100. Sus resultados mostraron que

las versiones más recientes no siempre superan a las anteriores en todos los contextos; sin embargo, YOLOv11 destacó por lograr un equilibrio entre velocidad y precisión, especialmente en la detección de objetos pequeños.

Durante el análisis realizado, se logró observar un patrón similar durante el proceso de entrenamiento, en el que los diferentes investigadores utilizaron laboratorios especializados, contando con un hardware potente para la realización de los entrenos.

1.6. Justificación

El avance de los algoritmos de detección de objetos, especialmente YOLO, ha impulsado aplicaciones destacadas por su precisión y velocidad (Redmon et al., 2016). Con el surgimiento de YOLOv8 y YOLOv11, se hace necesario un análisis sistemático que recopile y contraste evidencia científica sobre su desempeño, complementado con experimentos controlados en datasets representativos. Este enfoque permite identificar rigurosamente si las mejoras reportadas en cada versión representan ventajas reales en precisión y velocidad bajo hardware accesible. Además de su aporte técnico, el artículo promueve la formación en inteligencia artificial aplicada y el uso de tecnologías de aprendizaje profundo (IBM, 2024), fortaleciendo la producción académica en la Facultad de Ingeniería de la Universidad Santiago de Cali mediante un marco metodológico reproducible para futuras investigaciones en visión por computador.

1.7. Planteamiento del problema

La inteligencia artificial ha revolucionado diversos sectores mediante algoritmos de visión por computador para detección de objetos. Entre ellos, la familia YOLO destaca por su equilibrio entre precisión y velocidad (Redmon et al., 2016; Ultralytics, s. f.-a). Las versiones recientes, YOLOv8 y YOLOv11, han incorporado mejoras en generalización, eficiencia de entrenamiento y rendimiento en tiempo real. Sin embargo, existe escasa evidencia comparativa sistemática que evalué ambas versiones bajo condiciones experimentales homogéneas y hardware accesible, dificultando la selección del modelo más adecuado en contextos de recursos limitados.

Para estructurar la investigación, se adoptó la metodología PICO (Richardson et al., 1995), formulando los siguientes componentes:

- P (Población): Modelos de detección de objetos en entornos de hardware accesible con datos heterogéneos
- I (Intervención): YOLOv11 en variantes n, s, m
- C (Comparación): YOLOv8 bajo condiciones experimentales equivalentes
- O (Outcome): Precisión, tiempo de entrenamiento y eficiencia computacional

Con base a esto, la pregunta de investigación es:

¿La implementación de YOLOv11 en sus variantes (n, s, m) mejora la precisión, tiempo de entrenamiento y eficiencia computacional en comparación con YOLOv8, bajo condiciones experimentales equivalentes en entornos de hardware accesible con datos heterogéneos?

La Figura 4 muestra las causas principales que explica por qué se dificulta seleccionar entre YOLOv8 y YOLOv11 en entornos de recursos limitados.

Figura 4. Diagrama de espina de pescado

Nota. La figura representa la relación causa-efecto del problema de investigación. Fuente: Ishikawa (1986)

1.8. Objetivos

1.8.4. Objetivo General: Analizar el rendimiento de las versiones YOLOv8 y YOLOv11 en tareas de detección de objetos mediante una revisión sistemática de literatura y validación experimental, considerando el equilibrio entre precisión y velocidad en entornos computacionales de recursos limitados.

1.8.5. Objetivos Específicos:

- Identificar la literatura reciente sobre el rendimiento de YOLOv8 y YOLOv11 en tareas de detección de objetos mediante una revisión sistemática.
- Comparar los hallazgos de la literatura sobre el desempeño de YOLOv8 y YOLOv11 en distintas áreas de aplicación, considerando métricas de precisión, recall, mAP y velocidad.
- Validar experimentalmente el rendimiento de YOLOv8 y YOLOv11 en un entorno de recursos limitados, evaluando métricas de precisión (mAP), tiempo de entrenamiento y eficiencia computacional.

1.9. Estructura del articulo

La estructura de este artículo explora, en primer lugar, marco teórico, antecedentes y el contexto de la detección de objetos, lo que permite ubicar la problemática y justificar la pertinencia del estudio. Se revisa la literatura existente con el propósito de identificar trabajos previos de comparación, así como sus principales limitaciones. Con base en ello, se propone una metodología orientada a la solución del problema planteado. Finalmente, a partir de los resultados obtenidos en las comparaciones, se discuten los hallazgos y se analizan sus implicaciones prácticas.

2. METODOLOGÍA

2.1. Metodología de investigación

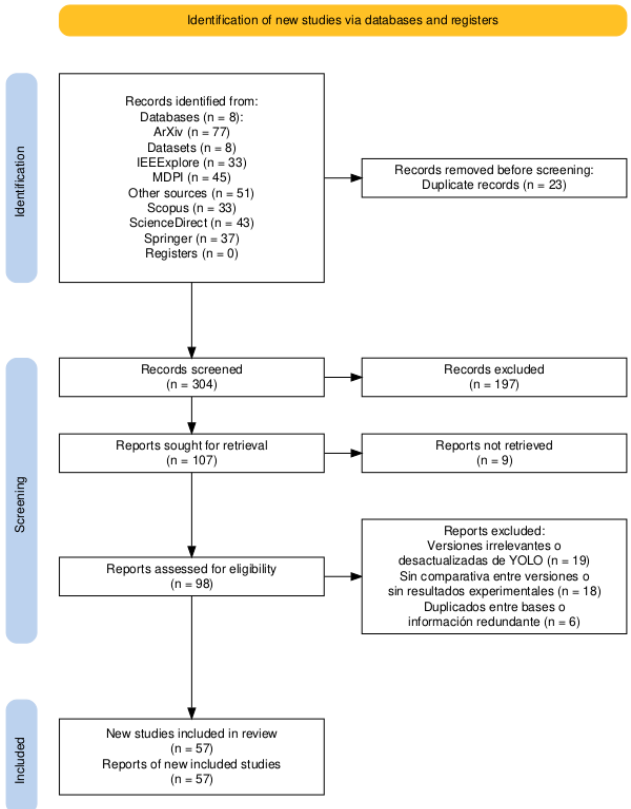
Este estudio adoptó un diseño mixto que integra revisión sistemática de literatura y validación experimental. El enfoque cuantitativo (Hernández Sampieri et al., 2006) permitió evaluar métricas estandarizadas de precisión, tiempo de entrenamiento y eficiencia computacional. Asimismo, se enmarca en una investigación de tipo comparativa (de Barrera, 2005), orientada a identificar similitudes y diferencias entre las versiones YOLOv8 y YOLOv11. Se emplearon los modelos de análisis y síntesis (Méndez Álvarez, 1987) para descomponer el problema y establecer relaciones entre ambos modelos.

La búsqueda sistemática siguió los lineamientos PRISMA (Moher et al., 2009; Page et al., 2021), estructurando el procedimiento en cuatro fases: identificación, cribado, elegibilidad e inclusión. La búsqueda se realizó en ocho fuentes: arXiv, IEEE Xplore, MDPI, Scopus, ScienceDirect, Springer, repositorios de datasets y documentación de Ultralytics, utilizando las palabras clave YOLO, YOLOv8, YOLOv11, object detection, computer vision, accuracy, speed y limited resources.

Los criterios de inclusión establecidos fueron: publicaciones entre 2016 y 2025, en inglés o español, enfocadas en detección de objetos mediante YOLOv8/YOLOv11, con datos empíricos o comparativas. Se excluyeron duplicados, textos no recuperables y artículos sin comparativas directas.

El proceso de selección (Figura 5) inició con 327 registros identificados. Tras eliminar 23 duplicados (7 automáticos y 16 manuales), se cribaron 304 registros por título y resumen, excluyéndose 197 por no cumplir los criterios de inclusión. Se seleccionaron 107 textos completos, de los cuales 9 no fueron recuperables. Los 98 artículos restantes fueron evaluados en detalle, excluyéndose 43 adicionales (19 versiones irrelevantes, 18 sin comparativas y 6 duplicados entre bases). Finalmente, 57 estudios fueron incluidos en la síntesis final, garantizando la validez, actualidad y replicabilidad del proceso metodológico.

Figura 5. Diagrama de flujo de la metodología PRISMA



Nota. La figura presenta el diagrama de flujo correspondiente a la metodología PRISMA. Fuente: Page (2021)

2.2. Metodología disciplinar

Para la fase práctica del artículo se utilizó la metodología del ciclo de vida de la ciencia de datos basada en el modelo OSEMN, por su adecuación a estudios experimentales y comparativos. Su eficacia ha sido comprobada en investigaciones recientes, como la de Andalusia et al. (2024), donde se aplicó OSEMN para entrenar un modelo YOLOv8 con datos médicos etiquetados con la plataforma Roboflow.

Figura 6. Metodología OSEMN

Nota. La figura muestra las etapas que componen la metodología disciplinar OSEMN. Fuente: Data Science PM (2022)

2.3. Etapa de la metodología

2.3.1. Obtain (Obtener)

La fase obtener, se compuso por una recolección de datasets de imágenes que se usarán en la comparación de la investigación. Se seleccionarán algunos repositorios públicos de la plataforma Roboflow y Ultralytics, cada uno con características distintas para evaluar los modelos en sus diversas versiones.

2.3.1.1. Conjunto de datos

Se emplearon distintos conjuntos de datos para comparar las versiones de YOLO. Aunque la mayoría presentan una estructura balanceada, algunos muestran variaciones en las anotaciones por clase y casos de superposición, lo que supone desafíos adicionales para la localización y clasificación precisa de los objetos.

2.3.1.1.1. PI3final Computer Vision Project

El conjunto de datos PI3final corresponde a un dataset de código abierto disponible en la plataforma Roboflow, orientado a tareas de detección de alimentos pertenecientes a las categorías de frutas, verduras, lácteos, proteínas y granos (teste, 2024), en la Tabla 3 y 4, se observa su configuración tanto en división de entrenamiento y anotaciones por cada objeto.

2.3.1.1.2. Conjunto de datos African Wildlife

El conjunto de datos African Wildlife es un dataset de código abierto disponible en la plataforma de Ultralytics, diseñado para tareas de detección de animales característicos de la fauna africana (Ultralytics, s. f.), en la Tabla 5 y 6, se observa su respectiva configuración tanto en división de entrenamiento y anotaciones por cada objeto.

2.3.2. Scrub (Limpiar)

En la fase de limpiar, se realizó una revisión de los conjuntos de datos para asegurar su calidad. Se verifico las anotaciones, se corrigieron posibles errores y se organizará la estructura de los archivos para garantizar que

los datos sean confiables y estuvieran listos para ser procesados por los modelos.

2.3.3. Explore (Explorar)

Se realizó un análisis cuantitativo de los datasets, identificando clases, número de instancias e imágenes por partición (entrenamiento, validación y prueba). Esta exploración permitió comprender las características de los datos y facilitar la interpretación de los resultados. Del mismo modo, se reporta la cantidad de instancias correspondientes a cada una de las clases de objetos incluidas.

Tabla 3. Distribución de imágenes del dataset PI3final Computer Vision Project

Tipo de Imagen	Cantidad de Imágenes
Entrenamiento	3074
Validación	286
Pruebas	0

Nota. Elaboración propia a partir del Dataset PI3final (2024)

Tabla 4. Número de instancias por clase del dataset PI3final Computer Vision Project

Clase	Numero de Instancias
0-lettuce	399
1-rice	279
2-banana	357
3-potato	771
4-meat	449
5-onion	583
6-beans	1442
7-chicken	511
8-orange	549
9-milk	306
10-apple	441
11-wartermelon	504
12-strawberry	1257
13-tomato	558
14-egg	282

Nota. Elaboración propia a partir del Dataset PI3final (2024)

Tabla 5. Distribución de imágenes del dataset African Wildlife

Tipo de Imagen	Cantidad de Imágenes
Entrenamiento	1049
Validación	225
Pruebas	227

Nota. Elaboración propia a partir del dataset African Wildlife (s. f.)

Tabla 6. Número de instancias por clase del dataset African Wildlife

Clase	Numero de Instancias
elephant	558
Rhino	399
buffalo	399
zebra	575

Nota. Elaboración propia a partir del dataset African Wildlife (s. f.)

2.3.4. Model (Modelar)

En la fase de modelar, se realizó el entramiento de los modelos YOLOv8 y YOLOv11 en sus variantes "n", "s", y "m". El entrenamiento se realizará bajo condiciones controladas y estandarizadas (mismo número de épocas y resolución de imagen) para garantizar que la comparación de los resultados sea justa y refleje las diferencias entre las arquitecturas de los modelos.

2.3.4.1. Proceso de entrenamiento

El entrenamiento se llevó a cabo con model.train() de YOLO, utilizando imágenes de 640 píxeles, 30 épocas y un batch size de 4, parámetros definidos según la capacidad del equipo para optimizar recursos y evitar sobreajuste. La carga de datos se configuró con workers=0 por compatibilidad en Windows, y se empleó CUDA para aprovechar la aceleración de la GPU (Figura 7).

Figura 7. Configuración del entrenamiento

```
model.train(
        data=dataset_path,
        epochs=30,
        imgsz=640,
        device='cuda',
        batch=4,
        workers=0
    )
```

Nota. La figura muestra la configuración del modelo durante el proceso de entrenamiento. Fuente: Elaboración propia (VS Code, editor de código)

2.3.5. Interpret (Interpretar)

En la fase final se analizaron y compararon los resultados de cada modelo usando métricas como mAP, matrices de confusión, Recall y tiempo de entrenamiento. El objetivo fue identificar cuál modelo ofrece el mejor equilibrio entre precisión y velocidad según el escenario evaluado.

3. RESULTADOS

3.1. Resultados comparación del desempeño

La Tabla 7 detalla el rendimiento comparativo de los modelos YOLOv8 y YOLOv11 a través de una serie de aplicaciones en el mundo real, recopiladas de diversas publicaciones científicas. En ella se especifica, para cada área de aplicación, qué modelo de YOLO ofrece el mejor desempeño y cuál es la métrica principal que lo respalda. Este análisis subraya la importancia de seleccionar la variante de YOLO más adecuada para optimizar resultados, ya sea buscando máxima precisión (mAP), alto Recall, o mejor balance en tareas específicas.

Tabla 7. Comparación del desempeño de los modelos YOLOv8 y YOLOv11 en distintas áreas de aplicación.

Área de Aplicación	Modelo recomendado	Métrica
Detección y segmentación de frutos en huertos complejos (Sapkota & Karkee, 2025).	YOLOv11m	Mayor mAP@50 tanto en detección de cajas como en segmentación de máscaras, incluyendo el desempeño en frutos parcialmente ocultos.
Detección de enfermedades de arroz (Teng et al., 2025).	YOLOv11n	Mejor precisión, recall y mAP; menor complejidad computacional.
Visión subacuática (Hung & Rodriguez, 2025).	YOLOv8s	Mayor precisión y recall en datasets de peces, mejor rendimiento en entornos marinos.
Detección de tumores cerebrales (Taha et al., 2025).	YOLOv11*	Mayor exactitud, precisión y recall, crítico para aplicaciones médicas.
Monitoreo automático del cumplimiento del uso de equipos de protección personal en obras de construcción (Sivanraj et al., 2025)	YOLOv11s	Mayor exactitud en detección y desempeño general según mAP y curvas Precision-Recall.
Monitoreo de pollos muertos en granjas (Bumbálek et al., 2025)	YOLOv8n	Mejor precisión por clase, recall, y mAP50-95; YOLOv11n más rápido.
Análisis de layout de documentos históricos (Santos Júnior et al., 2025)	YOLOv8s	Mejor mAP y recall para detectar correctamente las regiones.
Detección de defectos en paneles solares (Khanam et al., 2025).	YOLOv11m	Mejor equilibrio entre precisión, recall y velocidad
Detección de fracturas de radio distal (Selcuk et al., 2026).	YOLOv11m	Mayor precisión y exactitud, mejor localización de bounding boxes
Detección de cáncer de pulmón (D. K. Sharma et al., 2025).	YOLOv11*	Mejor balance entre precisión y recall, minimiza falsos positivos
Detección de leucemia (Awad & Aly, 2025)	YOLOv11s	Mejor exactitud, F1, precisión, Recall y especificidad; más confiable para células normales
Detección de enfermedades y plagas en hojas de café (Fragoso et al., 2025).	YOLOv8s	Mejor mAP, F1-Score y Recall general; balance entre cobertura y precisión

Nota. Los datos provienen de diversas investigaciones recientes.

El asterisco indica que no se especifica la variante exacta (n, s, m); sin embargo, en ausencia de esta información, se asume que corresponde a la versión medium, considerada la variante intermedia estándar de YOLO.

3.2. Resultados Dataset PI3final

La Tabla 8 presenta los resultados de entrenamiento, donde YOLOv8m destaca en precisión y recall, superando levemente a YOLOv11s en mAP y mAP50-95. Sin embargo, YOLOv11s muestra una mejor eficiencia computacional al requerir menos tiempo y peso de modelo.

Las matrices de confusión (Figuras 8, 9 y 10) evidencian un buen nivel de aciertos con pocos errores. La clase apple presenta confusiones con otros objetos, mientras que YOLOv8 muestra diagonales bien definidas, coherentes con los resultados de la Tabla 8.

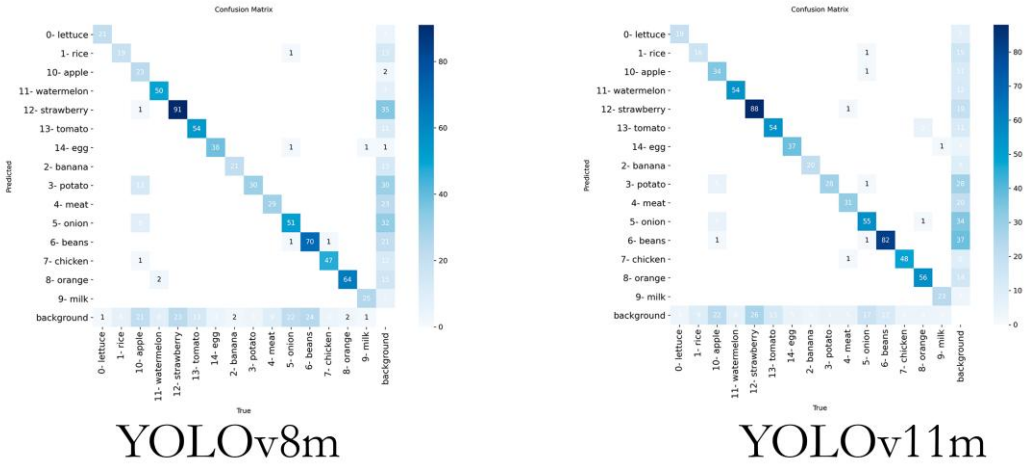
El análisis visual (Figura 11) revela que las variantes m detectan carne con mayor precisión; las versiones s y 8n identifican cebolla, aunque YOLOv8n confunde una manzana con un tomate. En general, los modelos detectan correctamente los alimentos, aunque persisten errores en la clasificación de ajos como pollo.

Tabla 8. Resultados de desempeño en el dataset PI3final.

VERSIONES	Precisión	Recall	F1-Score	mAP50	mAP50-95	Tiempo(m)	Peso (KB)
YOLOv8n	0.7499	**0.7767**	0.7630	0.7786	0.5838	42.20	6104.97
YOLOv8s	0.7934	0.7582	0.7754	0.7964	0.5958	86.34	21998.23
YOLOv8m	**0.8228**	0.7658	**0.7933**	0.8152	0.6190	**96.42**	**50823.07**
YOLOv11n	0.7795	0.7518	0.7654	0.7832	0.5730	58.34	5346.15
YOLOv11s	0.8060	0.7498	0.7769	**0.8213**	**0.6211**	58.20	18735.40
YOLOv11m	0.8091	0.7432	0.7748	0.8139	0.6143	95.26	39582.85

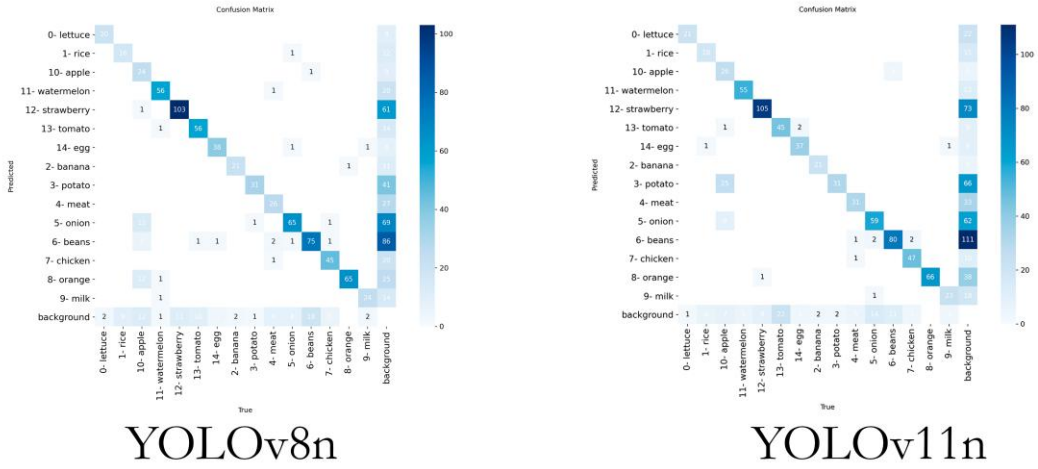
Nota. Elaboración propia con base en las métricas de desempeño de los modelos

Figura 8. Matriz de confusión Yolov8-11m en el dataset PI3final



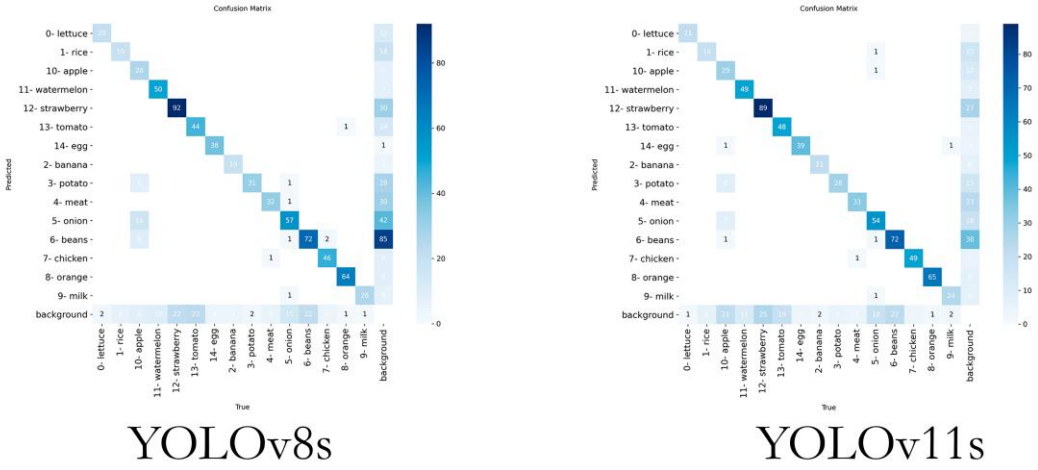
Nota: La diagonal principal indica el número de aciertos por clase. Fuente: Elaboración propia.

Figura 9. Matriz de confusión Yolov8-11n en el dataset PI3final



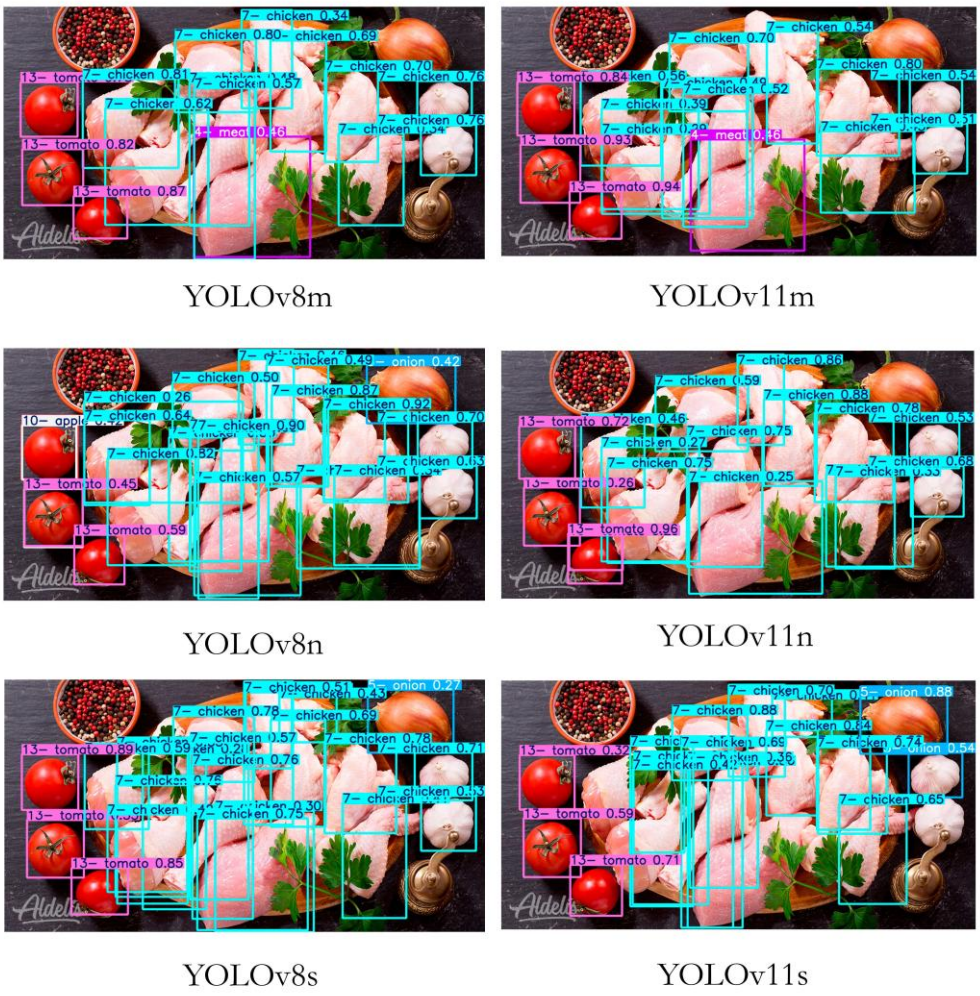
Nota: La diagonal principal indica el número de aciertos por clase. Fuente: Elaboración propia.

Figura 10. Matriz de confusión Yolov8-11s en el dataset PI3final



Nota: La diagonal principal indica el número de aciertos por clase. Fuente: Elaboración propia.

Figura 11. Resultado del análisis en imagen del dataset PI3final



Nota. La figura muestra la salida generada por el modelo al aplicarse sobre una imagen del dataset. Fuente: Elaboración propia.

3.3. Resultados Dataset African Wildlife

La Tabla 9 presenta los resultados obtenidos en el proceso de entrenamiento, en él se destaca YOLOv8s obteniendo el mayor F1-Score y mAP50-95, sobresaliendo por su equilibrio entre exactitud y eficiencia. Le sigue YOLOv8m, con resultados similares pero mayor peso y tiempo de entrenamiento. En términos de eficiencia computacional, los modelos livianos (YOLOv8n y YOLOv11n) sobresalen por sus menores tiempos y tamaños, siendo más prácticos para hardware limitado.

Las matrices de confusión (Figuras 12, 13 y 14) evidencian un buen desempeño general de YOLOv8 y YOLOv11 en la detección de especies, con altos aciertos y pocos errores; la clase rinoceronte muestra menor número de errores, lo que hace referencia a los altos valores de precisión de la Tabla 11.

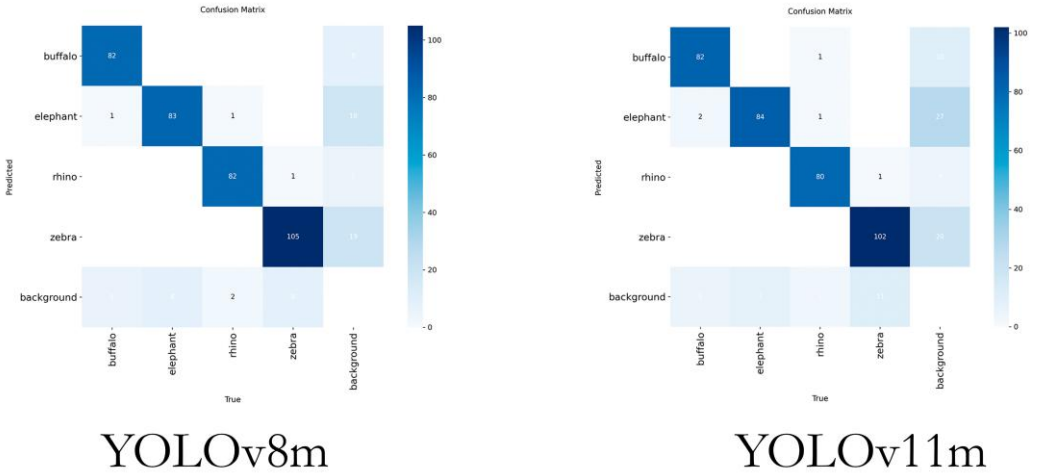
En el análisis visual (Figura 15), se observan errores en YOLOv11m (elefantes confundidos con búfalos), en YOLOv11n (jirafas como elefantes) y en YOLOv11s (detección incompleta de elefantes). En conjunto, las variantes de YOLOv8 demostraron un rendimiento más confiable frente a las de YOLOv11.

Tabla 9. Resultados de desempeño en el dataset African Wildlife

VERSIONES	Precisión	Recall	F1-Score	mAP50	mAP50-95	Tiempo(m)	Peso (KB)
YOLOv8n	0.9526	0.8977	0.9244	0.9580	0.7931	16.04	6082.47
YOLOv8s	**0.9590**	**0.9027**	**0.9300**	**0.9586**	**0.8044**	20.52	21971.28
YOLOv8m	0.9411	0.8938	0.9168	0.9492	0.7976	**34.58**	**50792.39**
YOLOv11n	0.9371	0.8763	0.9057	0.9428	0.7820	20.28	5323.27
YOLOv11s	0.9309	0.8890	0.9095	0.9450	0.7904	21.49	18708.39
YOLOv11m	0.9488	0.8607	0.9026	0.9413	0.7655	34.26	39547.97

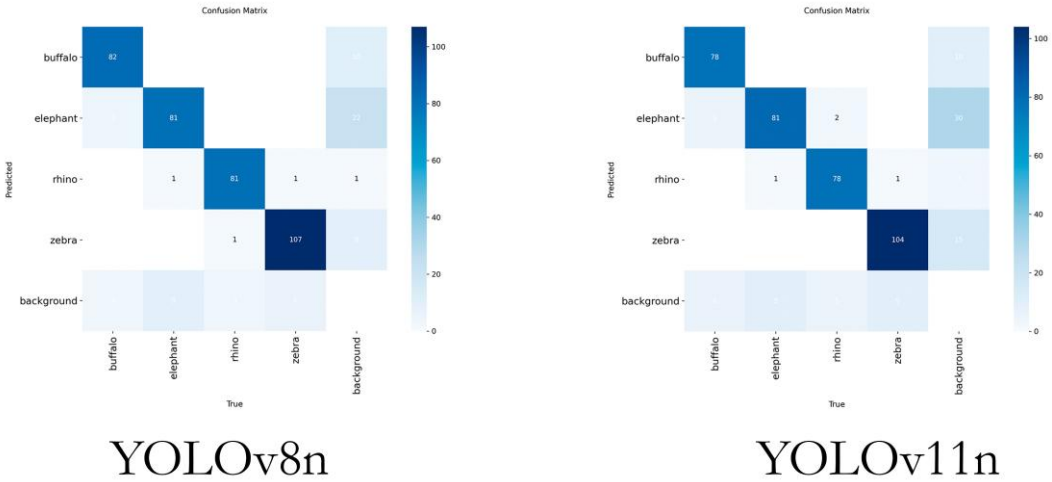
Nota. Elaboración propia con base en las métricas de desempeño de los modelos

Figura 12. Matriz de confusión Yolov8-11m en el dataset African Wildlife



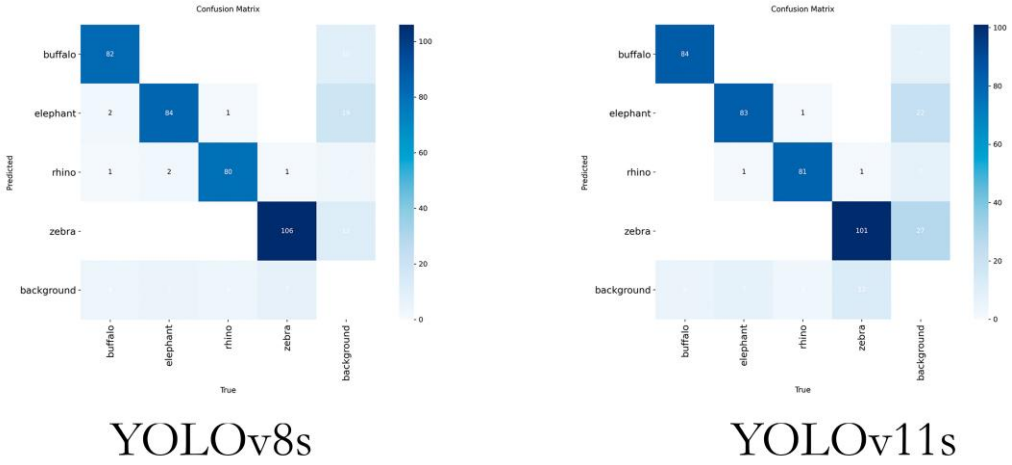
Nota: La diagonal principal indica el número de aciertos por clase. Fuente: Elaboración propia.

Figura 13. Matriz de confusión Yolov8-11n en el dataset African Wildlife.



***Nota*: La diagonal principal indica el número de aciertos por clase. Fuente: Elaboración propia.**

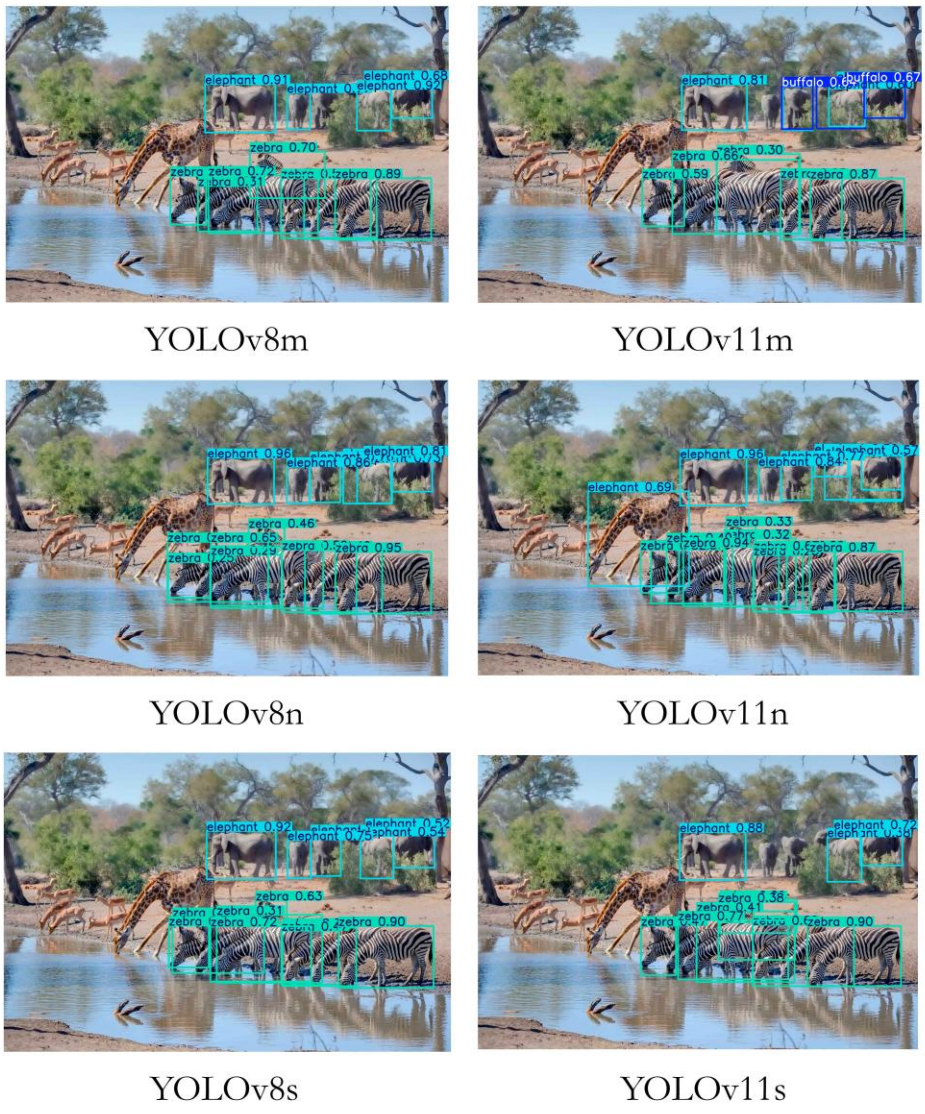
Figura 14. Matriz de confusión Yolov8-11s en el dataset African Wildlife.



YOLOv8s YOLOv11s

Nota: La diagonal principal indica el número de aciertos por clase. Fuente: Elaboración propia.

Figura 15. Resultado del análisis en imagen del dataset African Wildlife,



YOLOv8m YOLOv11m

YOLOv8n YOLOv11n

YOLOv8s YOLOv11s

Nota. La figura muestra la salida generada por el modelo al aplicarse sobre una imagen del dataset. Fuente: Elaboración propia.

4. CONCLUSIONES

En este artículo se desarrolló un análisis comparativo entre YOLOv8 y YOLOv11, con el objetivo de evaluar su rendimiento en términos de precisión, velocidad de entrenamiento bajo condiciones experimentales controladas.

Los resultados evidenciaron que YOLOv8 mantiene una mayor precisión y estabilidad tanto en escenarios complejos de fauna (African Wildlife) como en contextos de objetos alimenticios (PI3final). En contraste, YOLOv11 ofrece mayor eficiencia computacional y de almacenamiento, sin comprometer significativamente la exactitud. Estas diferencias confirman que la selección del modelo debe basarse según el contexto de uso: YOLOv8 es ideal cuando se prioriza la precisión, mientras que YOLOv11 resulta más adecuado para entornos con limitaciones de hardware.

A pesar de estos avances, se reconoce la necesidad de ampliar futuros estudios mediante la incorporación de hardware diversos, así como la integración de métricas adicionales como el consumo energético, latencia, velocidad en tiempo real o con un diferente sistema operativo, para complementar la comparación y fortalecer la valides de los hallazgos.

En conjunto, este trabajo contribuye al entendimiento de las diferencias arquitecturas y funciones entre ambas versiones de YOLO, proporcionando un marco metodológico reproducible y comparativo que puede servir de guía a la comunidad académica y profesional en proyectos de visión por computador, aprendizaje profundo y detección de objetos en contextos reales.

5. REFERENCIAS

Andalusia, F., Suakanto, S., Hamami, F., Mat Raffei, A. F., & Nuryatno, E. (2024). Real-Time Object Detection System for Hospital Assets Using YOLOv8. *2024 4th International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, 403-408. https://doi.org/10.1109/ICE3IS62977.2024.10775948

Awad, A., & Aly, S. A. (2025). *Early Diagnosis of Acute Lymphoblastic Leukemia Using YOLOv8 and YOLOv11 Deep Learning Models* (No. arXiv:2410.10701). arXiv. https://doi.org/10.48550/arXiv.2410.10701

Baglat, P., Hayat, A., Mostafa, S. S., Mendonça, F., & Morgado-Dias, F. (2025). Comparative analysis and evaluation of YOLO generations for banana bunch detection. *Smart Agricultural Technology*, *12*, 101100. https://doi.org/10.1016/j.atech.2025.101100

Bayoudh, K., Knani, R., Hamdaoui, F., & Mtibaa, A. (2022). A survey on deep multimodal learning for computer vision: Advances, trends, applications, and datasets. *The Visual Computer*, *38*(8), 2939-2970. https://doi.org/10.1007/s00371-021-02166-7

Bisong, E. (2019). Google Colaboratory. En *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 59-64). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7

Bradski, G. R., & Kaehler, A. (2011). *Learning OpenCV: Computer vision with the OpenCV library* (1. ed., [Nachdr.]). O'Reilly.

Bumbálek, R., Umurungi, S. N., Ufitikirezi, J. D. D. M., Zoubek, T., Kuneš, R., Stehlík, R., Lin, H.-I., & Bartoš, P. (2025).

Deep learning in poultry farming: Comparative analysis of Yolov8, Yolov9, Yolov10, and Yolov11 for dead chickens detection. *Poultry Science*, *104*(9), 105440. https://doi.org/10.1016/j.psj.2025.105440

Computer Vision Market Size, Trends | Forecast Analysis [2032]. (2025). https://www.fortunebusinessinsights.com/computer-vision-market-108827

Creative Commons. (s. f.). *Atribución 4.0 Internacional (CC BY 4.0)*. https://creativecommons.org/licenses/by/4.0/deed.es

de Barrera, J. H. (2005). *Cómo formular objetivos de investigación*. Instituto Universitario de Tecnología "José Antonio Anzoátegui"; Quirón Ediciones; Fundación Sypal.

Derczynski, L. (2016). Complementarity, F-score, and NLP Evaluation. En N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 261-266). European Language Resources Association (ELRA). https://aclanthology.org/L16-1040/

Fragoso, J., Silva, C., Paixão, T., Alvarez, A. B., Júnior, O. C., Florez, R., Palomino-Quispe, F., Savian, L. G., & Trazzi, P. A. (2025). Coffee-Leaf Diseases and Pests Detection Based on YOLO Models. *Applied Sciences*, *15*(9), 5040. https://doi.org/10.3390/app15095040

Free Software Foundation. (2007). *GNU Affero General Public License version 3*. https://www.gnu.org/licenses/agpl-3.0.html

Ghosh, A. (2024). *YOLO11: Redefining Real-Time Object Detection - Tutorial*. https://learnopencv.com/yolo11/

Gionfrida, L., Wang, C., Gan, L., Chli, M., & Carlone, L. (2024). Computer and Robot Vision: Past, Present, and Future [TC Spotlight]. *IEEE Robotics & Automation Magazine*, *31*(3), 211-215. https://doi.org/10.1109/MRA.2024.3428780

Hernández Sampieri, R., Mendoza Torres, C. P., & Fernández Collado, L. (2006). *Metodología de la investigación* (4.ª ed.). McGraw-Hill Education.

Hidayatullah, P., Syakrani, N., Sholahuddin, M. R., Gelar, T., & Tubagus, R. (2025). *YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review* (No. arXiv:2501.13400). arXiv. https://doi.org/10.48550/arXiv.2501.13400

Hotz, N. (2022, diciembre 31). OSEMN Data Science Life Cycle. *Data Science PM*. https://www.datascience-pm.com/osemn/

Hung, G., & Rodriguez, I. F. (2025). *A Comparative Study of YOLOv8 to YOLOv11 Performance in Underwater Vision Tasks* (No. arXiv:2509.12682). arXiv. https://doi.org/10.48550/arXiv.2509.12682

IBM. (2024, junio 17). *¿Qué es el deep learning?* https://www.ibm.com/es-es/think/topics/deep-learning

Ishikawa, K. (1986). *Guide to quality control* (Revised ed.). Asian Productivity Organization. https://archive.org/details/guidetoqualityco00ishi

Jegham, N., Koh, C. Y., Abdelatti, M., & Hendawi, A. (2025). *YOLO evolution: A comprehensive benchmark and architectural review of YOLOv12, YOLO11, and their previous versions*. arXiv. https://doi.org/10.48550/arXiv.2411.00201

Jiang, T., & Zhong, Y. (2025). *ODverse33: Is the new YOLO version always better? A multi-domain benchmark from YOLOv5 to v11*. arXiv preprint arXiv:2502.14314. https://doi.org/10.48550/arXiv.2502.14314

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J., imyhxy, Lorna, Yifu), 曾逸夫(Zeng, Wong, C., V, A., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, … Jain, M. (2022). *ultralytics/yolov5: V7.0 - YOLOv5 SOTA Realtime Instance Segmentation* [Software]. Zenodo. https://doi.org/10.5281/zenodo.7347926

Jocher, G., Qiu, J., & Chaurasia, A. (2023). *Ultralytics YOLO* (Versión 8.0.0) [Python]. https://github.com/ultralytics/ultralytics

Journal of Network and Computer Applications. (s. f.). https://www.sciencedirect.com/topics/computer-science/optimization-theory

Kang, S., Hu, Z., Liu, L., Zhang, K., & Cao, Z. (2025). Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis. *Electronics*, *14*(6), 1104. https://doi.org/10.3390/electronics14061104

Khanam, R., Asghar, T., & Hussain, M. (2025). Comparative Performance Evaluation of YOLOv5, YOLOv8, and YOLOv11 for Solar Panel Defect Detection. *Solar*, *5*(1), 6. https://doi.org/10.3390/solar5010006

Lessig, L. (2002). *The future of ideas: The fate of the commons in a connected world*. Random House.

Méndez Álvarez, C. E. (1987). *Metodología: Guía para elaborar diseños de investigación en ciencias económicas, contables y administrativas*. McGraw-Hill.

Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., & The PRISMA Group. (2009). Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement. *PLoS Medicine*, *6*(7), e1000097. https://doi.org/10.1371/journal.pmed.1000097

Murat, A. A., & Kiran, M. S. (2025). A comprehensive review on YOLO versions for object detection. *Engineering Science and Technology, an International Journal*, *70*, 102161. https://doi.org/10.1016/j.jestch.2025.102161

NVIDIA Corporation. (s. f.). *CUDA*. NVIDIA Developer. Recuperado 1 de octubre de 2025, de https://developer.nvidia.com/about-cuda

Padilla, R., Netto, S. L., & da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 237-242. https://doi.org/10.1109/IWSSIP48289.2020.9145130

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, n71. https://doi.org/10.1136/bmj.n71

Palaniappan, D., Jain, R., Premavathi, T., Parmar, K., Ghribi, W., Ahmed, A. M., & Ahmad, N. (2025). YOLO in Healthcare: A Comprehensive Review of Detection Architectures, Domain Applications, and Future Innovations. *IEEE Access*, *13*, 145714-145735. https://doi.org/10.1109/ACCESS.2025.3599358

Raj, R., Sreemathy, R., Turuk, M., Jagdale, J., & Anish, M. (2025). Performance comparison of different versions of YOLO for Indian Sign Language captioning in real time of multiple signers. *Procedia Computer Science*, *259*, 991-1000. https://doi.org/10.1016/j.procs.2025.04.053

Ramos, L. T., & Sappa, A. D. (2025). A comprehensive analysis of YOLO architectures for tomato leaf disease identification. *Scientific Reports*, *15*(1), 26890. https://doi.org/10.1038/s41598-025-11064-0

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788. https://doi.org/10.1109/CVPR.2016.91

Richardson, W. S., Wilson, M. C., Nishikawa, J., & Hayward, R. S. (1995). The well-built clinical question: A key to evidence-based decisions. *ACP Journal Club*, *123*(3), A12-A13.

Santos Júnior, E. S. dos, Paixão, T., & Alvarez, A. B. (2025). Comparative Performance of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for Layout Analysis of Historical Documents Images. *Applied Sciences*, *15*(6), 3164. https://doi.org/10.3390/app15063164

Sapkota, R., & Karkee, M. (2025). *Comparing YOLOv11 and YOLOv8 for instance segmentation of occluded and non-occluded immature green fruits in complex orchard environment* (No. arXiv:2410.19869). arXiv. https://doi.org/10.48550/arXiv.2410.19869

Selcuk, B., Serif, S., & Serif, T. (2026). A Comparative Study on Distal Radius Fracture Detection: YOLOv8 and YOLOv11 Versus Faster R-CNN. En M. Younas, I. Awan, L. Martin, & H. Wu (Eds.), *Mobile Web and Intelligent Information Systems* (pp. 229-242). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-02060-4_16

Sharma, A., Kumar, V., & Longchamps, L. (2024). Comparative performance of YOLOv8, YOLOv9, YOLOv10, YOLOv11 and Faster R-CNN models for detection of multiple weed species. *Smart Agricultural Technolo*, *9*, 100648. https://doi.org/10.1016/j.atech.2024.100648

Sharma, D. K., Pal, M. K., & Singh, A. K. (2025). A comparative analysis of YOLO models for efficient lung tumor detection using CT images. *Health and Technology*, *15*(4), 787-800. https://doi.org/10.1007/s12553-025-00989-1

Sivanraj, S., Uduwage, D., & Tripathi, M. (2025). *Comparison of YOLO algorithms for PPE compliance monitoring at construction sites*. https://dl.lib.uom.lk/handle/123/24190

Stallman, R. M., & Lessig, L. (2004). *Software libre para una sociedad libre* (2.ª). Traficantes de Sueños.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*.

Taha, A. M., Aly, S. A., & Darwish, M. F. (2025). *Detecting Glioma, Meningioma, and Pituitary Tumors, and Normal Brain Tissues based on Yolov11 and Yolov8 Deep Learning Models* (No. arXiv:2504.00189). arXiv. https://doi.org/10.48550/arXiv.2504.00189

Teng, H., Wang, Y., Li, W., Chen, T., & Liu, Q. (2025). Advancing Rice Disease Detection in Farmland with an Enhanced YOLOv11 Algorithm. *Sensors*, *25*(10), 3056. https://doi.org/10.3390/s25103056

teste. (2024). *PI3final* [Conjunto de datos]. Roboflow Universe. https://universe.roboflow.com/teste-umz06/pi3final

Tsung-Yi, L., Michael, M., Serge, B., James, H., Pietro, P., Deva, R., Piotr, D., & C. Lawrence, Z. (s. f.). *COCO - Common Objects in Context*. Recuperado 14 de septiembre de 2025, de https://cocodataset.org/#home

Ultralytics. (s. f.-a). *Documentación de los modelos YOLO de Ultralytics*. https://docs.ultralytics.com/es/

Ultralytics. (s. f.-b). *Métricas de Rendimiento de YOLO*. https://docs.ultralytics.com/es/guides/yolo-performance-metrics

Ultralytics. (2023). *YOLOv8: Documentación técnica*. https://docs.ultralytics.com/es/models/yolov8

Ultralytics. (2024). *YOLOv11: Documentación técnica*. https://docs.ultralytics.com/es/models/yolo11

Ultralytics. (s. f.). *African Wildlife* [Conjunto de datos]. https://docs.ultralytics.com/es/datasets/detect/african-wildlife/