UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

Facultad de Ingeniería

Escuela Profesional de Ingeniería en Informática y Sistemas

ANÁLISIS COMPARATIVO DE MODELOS DE INTELIGENCIA ARTIFICIAL BASADOS EN APRENDIZAJE PROFUNDO APLICADOS A LA DETECCIÓN DE FRUTOS DE OLIVO EN LA YARADA LOS PALOS - TACNA

TESIS

Presentada por:

Bach. ALEJANDRO DANIEL YANAPA CHICALLA

Para optar el Título Profesional de:

INGENIERO EN INFORMÁTICA Y SISTEMAS

TACNA – PERÚ

2024

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS

"ANÁLISIS COMPARATIVO DE MODELOS DE INTELIGENCIA ARTIFICIAL BASADOS EN APRENDIZAJE PROFUNDO APLICADOS A LA DETECCIÓN DE FRUTOS DE OLIVO EN LA YARADA LOS PALOS – TACNA"

TESIS PRESENTADA Y APROBADA EL 23 DE AGOSTO DEL 2024
ESTANDO EL JURADO CALIFICADOR INTEGRADO POR:

Presidente : Dr. Edwin Antonio Hinojosa Ramos

Secretario : Mtro. Hugo Manuel Barraza Vizcarra

Vocal : Dr. Erbert Francisco Osco Mamani

Asesor : Dr. Erbert Francisco Osco Mamani

Universidad Nacional Jorge Basadre Grohmann

Facultad de Ingeniería

Escuela Profesional de Ingeniería en Informática y Sistemas

Acta de sustentación de Título Profesional

En Laboratorio "A" de la Escuela Profesional de Ingeniería en Informática y Sistemas, siendo las 09:35 horas del día 23 de agosto del 2024, y cumpliendo lo señalado en el Reglamento de Grados y Títulos de la Facultad de Ingeniería, se reunió el Jurado Calificador integrado por los docentes:

- Dr. Edwin Antonio Hinojosa Ramos — Presidente
- Mtro. Hugo Manuel Barraza Vizcarra — Secretario
- Dr. Erbert Francisco Osco Mamani — Vocal

Designados mediante R.F. Nº 9062-2024-FAIN/UNJBG, para evaluar la Tesis: 'Análisis comparativo de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada, Los Palos - Tacna', presentada por el bachiller Daniel Yanapa Chicalla, para optar el Título Profesional Ingeniero en Informática y Sistemas. Le asesoró, Dr. Erbert Francisco Osco Mamani (R.F. Nº 8315-2023-FAIN/UNJBG).

Dicho acto de sustentación se desarrolló en dos etapas: exposición y absolución de preguntas; procediéndose luego a la evaluación por parte de los miembros del Jurado.

Habiendo absuelto las preguntas que le fueron formuladas por los miembros del Jurado Calificador, y de conformidad con las respectivas disposiciones reglamentarias, procedieron a deliberar y calificar, declarándolo APROBADO por unanimidad, con el calificativo numérico de 16 (DIECISÉIS) y cualitativo de bueno.

Siendo las 10:30 horas del día 23 de agosto del 2024, los miembros del Jurado Calificador firman la presente Acta en señal de conformidad.

Dr. Edwin Antonio Hinojosa Ramos
Miembro del Jurado Calificador

Mtro. Hugo Manuel Barraza Vizcarra
Miembro del Jurado Calificador

Dr. Erbert Francisco Osco Mamani
Miembro del Jurado Calificador

Fe de erratas:
* En la línea 10 dice: La Yarada, Los Palos
Debe decir: La Yarada Los Palos

* En la línea 11 dice: Daniel Yanapa
Debe decir: Alejandro Daniel Yanapa

Universidad Nacional Jorge Basadre Grohmann

Facultad de Ingeniería

Escuela Profesional de Ingeniería en Informática y Sistemas

Certificado de similitud

Yo, Dr. Erbert Francisco Osco Mamani, en mi condición de asesor acreditado por la Resolución de Facultad Nº 8260-2023-FAIN/UNJBG, de la Tesis: 'Análisis comparativo de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna', presentado por el bachiller ALEJANDRO DANIEL YANAPA CHICALLA, para optar el Título Profesional Ingeniero en Informática y Sistemas.

Habiendo cumplido con lo establecido en el reglamento de originalidad y de similitud de trabajos de investigación y producción intelectual, considerando que según la revisión, evaluación y análisis realizado a través del software de similitud textual Turnitin, cuenta con el NIVEL DE SIMILITUD de 8%. Por lo que certifico la similaridad de la tesis, que está de acuerdo al NIVEL PERMITIDO, para continuar con los trámites correspondientes y para su publicación en el repositorio Institucional.
Se emite el presente certificado con fines de continuar con los trámites respectivos para su obtención del título profesional.

Dr. Erbert Francisco Osco Mamani
DNI. 00409196

Bach. Alejandro Daniel Yanapa Chicalla
DNI. 71266919

DEDICATORIA

A mi amada familia,
quienes han sido mi
fuente de apoyo,
amor y fortaleza a lo
largo de mi vida.

AGRADECIMIENTOS

A Dios y a mi familia por su amor incondicional y su eterno apoyo en cada paso de mi camino.

A mis estimados docentes de la Escuela Profesional de Ingeniería en Informática y Sistemas de la Universidad Nacional Jorge Basadre Grohmann por su orientación, sabiduría y apoyo constante a lo largo de mi trayectoria académica.

Al proyecto “Estimación de productividad de olivares usando técnicas de visión artificial y aprendizaje profundo en la Región de Tacna” de la Universidad Nacional Jorge Basadre Grohmann, Tacna - Perú por proporcionar el valioso conjunto de imágenes que hizo posible este trabajo.

ÍNDICE TEMÁTICO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La actividad de producción agrícola relacionada a la producción de olivo representa un pilar clave para el crecimiento económico en el sur del Perú. Tacna encabeza la producción (77%) y exportación de aceitunas (64%).

El sector agrícola en Tacna ha dependido de técnicas de producción tradicionales. Sin embargo, la falta de adopción de nuevas tecnologías, como la detección automática de frutos, ha limitado el potencial de mejora y eficiencia en la producción agrícola.

Por ello, en este trabajo se aborda el desafío de la detección automática de frutos de olivo en la región de La Yarada Los Palos - Tacna. Se concentró en evaluar y comparar el rendimiento de modelos de detección de objetos con tres arquitecturas ampliamente usadas: YOLOv8, Faster R-CNN y RetinaNet, basándonos en los indicadores de mAP50 y mAP50-95.

Se concluyó que el modelo con mejor desempeño en el indicador mAP50 fue el que tuvo como arquitectura la variante YOLOv8s (YOLOv8 Small), con un valor máximo de 94.751% y un valor promedio de 94.7349%. Asimismo, para el indicador mAP50-95 el modelo con mejor resultado fue YOLOv8s, con un valor máximo de 77.562% y un valor promedio de 77.5076%. Finalmente, mediante pruebas estadísticas mencionamos que existe diferencia significativa en los resultados generados por los modelos con las tres arquitecturas tanto para el indicador mAP50 como para el indicador mAP50-95.

Palabras clave: Detección de objetos, aprendizaje profundo, inteligencia artificial, aceituna.

ABSTRACT

Agricultural production activity related to olive production represents a key pillar for economic growth in southern Peru. Tacna leads in production (77%) and export (64%) of olives.

The agricultural sector in Tacna has relied on traditional production techniques. However, the lack of adoption of new technologies, such as automatic fruit detection, has limited the potential for improvement and efficiency in agricultural production.

Therefore, this work addresses the challenge of automatic olive fruit detection in the region of La Yarada Los Palos - Tacna. It focused on evaluating and comparing the performance of object detection models using three widely used architectures: YOLOv8, Faster R-CNN, and RetinaNet, based on mAP50 and mAP50-95 indicators.

It was concluded that the model with the best performance in the mAP50 indicator was the one with the YOLOv8s (YOLOv8 Small) architecture, with a maximum value of 94.751% and an average value of 94.7349%. Likewise, for the mAP50-95 indicator, the model with the best result was YOLOv8s, with a maximum value of 77.562% and an average value of 77.5076%. Finally, through statistical tests, we mentioned that there is a significant difference in the results generated by the models with the three architectures for both the mAP50 and mAP50-95 indicators.

Keywords: Object detection, deep learning, artificial intelligence, olive.

INTRODUCCIÓN

La aceituna desempeña un papel importante en la industria agrícola de Tacna. Por ello, surge la necesidad de aplicar nuevas tecnologías en procesos como la producción y recolección de olivos.

En este contexto, la aplicación de técnicas de inteligencia artificial, en particular los modelos basados en aprendizaje profundo, ofrece nuevas oportunidades para mejorar la detección automática de frutos de olivo en el campo. Estos modelos pueden procesar grandes volúmenes de datos de imágenes con rapidez y precisión, lo que los convierte en herramientas valiosas para la agricultura de precisión y la gestión de cultivos.

El presente trabajo se centra en el análisis comparativo de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en la región de La Yarada Los Palos - Tacna, Perú. Se exploró el rendimiento de diferentes arquitecturas de modelos, como YOLOv8, Faster R-CNN y RetinaNet, en la detección de frutos de olivo en imágenes capturadas directamente en el campo. Esta investigación está organizada en cinco capítulos de la siguiente manera:

El capítulo I trata la descripción de la problemática de la investigación, la formulación del problema, los objetivos planteados, además de la justificación y las limitaciones del estudio.

El capítulo II recopila antecedentes relacionados al ámbito de la investigación, así como las bases teóricas y definiciones.

En el capítulo III se detalla la metodología, procedimientos y técnicas usadas en este trabajo.

En el capítulo IV se muestran los resultados, así como la contrastación estadística de las hipótesis.

El capítulo V muestra las discusiones. Finalmente, las conclusiones, las recomendaciones, las referencias bibliográficas y los anexos.

CAPÍTULO I
PLANTEAMIENTO DEL PROBLEMA

1.1 Antecedentes del problema

La idea de esta investigación surgió a partir de la observación de los desafíos que enfrentan los agricultores en la región de Tacna, particularmente en La Yarada Los Palos. La producción de aceitunas, una actividad vital para la economía local, se realiza utilizando métodos tradicionales de cultivo y cosecha. Estos métodos limitan la capacidad de los agricultores para optimizar tanto la producción como la calidad del producto.

Para enfrentar estos retos, se ha considerado la aplicación de técnicas de inteligencia artificial y visión por computadora. La investigación fue inspirada por varios estudios que destacan el potencial de estas tecnologías en la agricultura y se consideró también las problemáticas que enfrentaron estos estudios. Aquino et al. (2019) señalaron que la olivicultura se enfrenta a problemas de producción y sostenibilidad ambiental. En su artículo, abordan la necesidad de una estimación temprana y precisa de la producción de aceitunas en España, proponiendo la detección y el conteo de frutos mediante visión artificial y redes neuronales convolucionales. Resaltan la importancia de aplicar técnicas de morfología matemática y filtrado estadístico para preprocesar las imágenes.

Bellocchio et al. (2019) destacaron que una estimación precisa del rendimiento del cultivo facilita la planificación de las operaciones de recolección y venta de la cosecha. Enfrentaron el problema del conteo de frutas, señalando la necesidad de una supervisión intensiva y detallada, la cual es costosa y propensa a errores en escenarios agrícolas. También mencionaron los desafíos relacionados con la consistencia espacial de las imágenes, así como la variabilidad en las escalas y ubicaciones de los frutos dentro de las imágenes.

El artículo de Cano et al. (2021) se centra en la detección de defectos en los frutos de olivo durante el proceso de producción de aceite de oliva extra virgen. Abordaron la variabilidad y limitaciones de los métodos manuales, destacando la necesidad de mantener alta precisión en condiciones no controladas mediante técnicas automáticas y avanzadas de procesamiento de imágenes.

Por otro lado, en la investigación de Zhang et al. (2022) se aborda el problema del conteo de frutos de acebo mediante visión computacional. Mencionan que el conteo de frutos en un árbol completo apenas se ha estudiado y se carece de técnicas precisas y robustas. Describen dificultades como la cobertura de los frutos, sombras, desenfoque en las imágenes y la agrupación de los frutos.

Estos antecedentes resaltan la viabilidad de aplicar modelos de inteligencia artificial para la detección de frutos de olivo en la región de Tacna, específicamente mediante el uso de visión por computadora y redes neuronales convolucionales. Este estudio busca llenar el vacío existente al evaluar modelos en un contexto local, proporcionando soluciones adaptadas a las características específicas de los cultivos y el entorno regional.

1.2 Descripción del problema

En la región de Tacna, la producción de aceitunas constituye una porción importante dentro del sector agrícola. En el Perú se cuentan con 45,000 hectáreas de cultivo destinadas al olivo, Tacna abarca el 86% a nivel nacional con 35,000 hectáreas aproximadamente, estas se encuentran ubicadas principalmente en el distrito de La Yarada Los Palos (Agraria.pe, 2024).

Tacna se destaca por exportar principalmente aceitunas, que constituyen el producto agrícola principal en sus exportaciones, representando el 46% del total de exportaciones agropecuarias de la región. (Ministerio de Comercio Exterior y Turismo [MINCETUR], 2024).

La mayor parte de las exportaciones de aceituna de Tacna tienen como destinos principales a Brasil, que recibe el 62%, seguido por Chile, con el 36%. Tacna lidera tanto en la producción (77%) como en la exportación (64%) de aceituna, según las cifras más recientes. Sin embargo, en el primer semestre de 2023 la producción de aceituna en Tacna experimentó una caída del 16% (MINCETUR, 2024).

A pesar de su importancia, la industria de la aceituna en Tacna se enfrenta a desafíos cruciales relacionados con la gestión eficiente de la producción, la calidad del producto y las exportaciones. La producción de aceituna en el país se distingue por la predominancia de pequeños agricultores, quienes emplean un bajo nivel tecnológico (Exportemos.pe, s.f.).

En este contexto, surge la necesidad de optimizar los procesos agrícolas mediante la implementación de tecnologías avanzadas. El aprendizaje profundo ha emergido como

el enfoque destacado en la inteligencia artificial para abordar una amplia gama de desafíos, incluidos aquellos asociados con la visión computacional (Ekman, 2021).

La aplicación de inteligencia artificial, en particular la utilización de modelos basados en aprendizaje profundo, se presenta como una solución potencial para mejorar la gestión y eficiencia en la detección de olivos. La detección automática de frutos en la industria agrícola, particularmente en cultivos como el olivo, es crucial para optimizar la eficiencia de la producción y reducir los costos asociados con la mano de obra. Pero debido a la variedad de arquitecturas y técnicas de modelos para la detección de objetos, es fundamental realizar una evaluación comparativa exhaustiva para determinar cuál de estos enfoques es más adecuado y si los rendimientos de estos modelos para la detección de frutos de olivo en datos obtenidos localmente son estadísticamente diferentes.

1.3 Formulación del problema

1.3.1 Problema general

¿Cómo es el rendimiento de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna?

1.3.2 Problemas específicos

a) ¿Cómo es el rendimiento en cuanto al indicador mAP50 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna?
b) ¿Cómo es el rendimiento en cuanto al indicador mAP50-95 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna?

1.4 Objetivos de la investigación

1.4.1 Objetivo general

Comparar el rendimiento de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

1.4.2 Objetivos específicos

a) Comparar el rendimiento en cuanto al indicador mAP50 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

b) Comparar el rendimiento en cuanto al indicador mAP50-95 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

1.5 Justificación e importancia de la investigación

Se hicieron trabajos relacionados a la comparación de modelos de inteligencia artificial aplicados a la detección de aceitunas como los de Aljaafreh et al. (2023) y Zhu et al. (2024), pero hay una escasez de estudios previos relacionados con la detección automática de frutos de olivo en nuestra localidad.

Además, se destaca la relevancia de utilizar imágenes capturadas en el entorno específico de la región, lo cual añade un valor contextual único al conjunto de datos generado. Este conjunto de datos no solo sirve como punto de partida para nuestra investigación, sino que también podría aplicarse en futuras investigaciones relacionadas a la visión artificial. Asimismo, las técnicas desarrolladas en este estudio podrían tener aplicaciones prácticas en la optimización de labores como el etiquetado de datos y el entrenamiento de modelos de inteligencia artificial.

La comparación de diferentes modelos de detección automática de frutos de olivo se justifica por la necesidad de identificar el enfoque más eficaz y preciso para esta tarea. Dada la variedad de arquitecturas de redes neuronales disponibles y la constante evolución de las técnicas de aprendizaje profundo, es crucial evaluar y comparar el rendimiento de varios modelos en condiciones específicas de cultivo de olivos.

1.6 Limitaciones

El conjunto de datos empleado para este estudio presenta las siguientes limitaciones:

La ausencia de fotografías a lo largo de múltiples temporadas: Esto puede restringir la capacidad de los modelos para adaptarse a variaciones estacionales y cambios a lo largo del tiempo en la producción de frutos de olivo.

Variabilidad en las condiciones de captura: La variabilidad limitada en las condiciones de iluminación, ángulos de visión, y diversidad en la apariencia de los frutos de olivo dentro del conjunto de datos podría afectar la capacidad de los modelos para generalizar a entornos diversos en condiciones reales de campo.

Estas limitaciones pueden impactar a los modelos en su capacidad de adaptación a escenarios cambiantes o poco predecibles.

1.7 Viabilidad del estudio

El estudio es viable ya que se dispone del consentimiento necesario para utilizar los datos en el entrenamiento de los modelos. Además, se cuenta con las herramientas y tecnologías necesarias para el aprendizaje profundo y los algoritmos de detección de objetos, así como con la infraestructura computacional requerida para su desarrollo. Finalmente, la investigación cumple con todas las normativas legales vigentes.

1.8 Formulación de hipótesis

1.8.1 Hipótesis general

Existe diferencia significativa en el rendimiento de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

1.8.2 Hipótesis específicas

a) Existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

b) Existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50-95 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

1.9 Variables

Según Hernández Sampieri et al. (2014), una variable es una propiedad que puede cambiar y cuya variación se puede medir u observar.

La única variable considerada en este trabajo es el rendimiento de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo, por lo que es univariable. Por lo tanto, no hay variables independientes ni dependientes.

1.9.1 Definición de la variable

Variable de estudio: Rendimiento de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo.

- Definición conceptual: Se refiere al indicador medible de qué tan efectivo es el modelo de inteligencia artificial en la detección de frutos de olivo.
- Definición operacional: Evalúa al modelo de inteligencia artificial en su capacidad para detectar frutos de olivo en un conjunto de imágenes.

1.10 Operacionalización de variables

Arias (2012) menciona que las variables simples son aquellas que se expresan directamente mediante un indicador o una unidad de medida y no se dividen en dimensiones. La variable usada en esta investigación es simple y tiene dos indicadores.

Tabla 1

Tabla de operacionalización de variables

Variable	Definición conceptual	Definición operacional	Indicadores	Escala
Rendimiento de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo	Evalúa al modelo de inteligencia artificial en su capacidad para detectar frutos de olivo en un conjunto de imágenes.	Evalúa al modelo de inteligencia artificial en su capacidad para detectar frutos de olivo en un conjunto de imágenes.	I1: mAP50	Decimal
			I2: mAP50-95	Decimal

CAPÍTULO II
MARCO TEÓRICO

2.1 Antecedentes del trabajo de investigación

Aljaafreh et al. (2023) en su artículo exploran el uso de modelos de redes neuronales profundas para la detección de aceitunas en árboles a partir de videos digitales. Utilizando variantes de YOLOv5 y YOLOR, se construye y evalúa un conjunto de datos de 1,200 imágenes. Los resultados indican que las variantes de YOLOv5, especialmente YOLOv5s, logran una alta precisión (más del 0.75 mAP0.5) en la detección en tiempo real de aceitunas, destacándose como el modelo más eficiente entre las variantes de YOLOv5 y YOLOR. La metodología se enfoca en la evaluación de algoritmos de detección de objetos mediante un conjunto de datos específicamente creado para aceitunas en el árbol.

Zhu et al. (2024) en su estudio presentan el modelo Olive-EfficientDet para la detección de la madurez de frutos de olivo en huertos. Se destaca por incorporar el módulo de atención CBAM y optimizar la estructura Bi-FPN para abordar desafíos de solapamiento y ocultamiento de frutos. Los resultados experimentales muestran un impresionante mAP en la detección de madurez, alcanzando un 94.60%, 95.45%, 93.75%, y 96.05% para las variedades 'Frantoio', 'Ezhi 8', 'Leccino' y 'Picholine', respectivamente. Con un tiempo de detección de 337 ms por imagen y un tamaño de modelo de solo 32.4 MB, Olive-EfficientDet supera a otros métodos, incluidos SSD, EfficientDet, YOLOv3 y Faster RCNN, proporcionando una base técnica prometedora para la detección de madurez en robots de cosecha de aceitunas.

En el trabajo de Li et al. (2021) se propone el método Lemon-YOLO (L-YOLO) para mejorar la detección precisa y en tiempo real de limones en entornos naturales. Se utiliza la red SE_ResGNet34 en lugar de DarkNet53 en YOLOv3, lo que aumenta la precisión y la velocidad de detección. Se agrega el módulo SE_ResNet para mejorar la calidad de las representaciones de la red. Los resultados experimentales muestran que L-YOLO tiene una precisión promedio del 96.28% y una velocidad de detección de 106

FPS, superando a YOLOv3. Esto ofrece soporte técnico para la recolección automatizada de limones y otros frutos.

En el estudio de Angelika et al. (2024) se busca mejorar el reconocimiento automático de matrículas mediante el uso de modelos de super-resolución. Se emplearon dos fases: detección de matrículas y reconocimiento de matrículas. Para la primera fase se emplearon modelos de detección de objetos, como YOLOv8 y Faster R-CNN, para mejorar el reconocimiento automático de matrículas. YOLOv8 alcanzó una precisión del 93% en AP50 y 70,8% en AP50-95 en su conjunto de validación, mientras que Faster R-CNN obtuvo un 71% en AP50 y 50% en AP50-95. Estos resultados destacan el desempeño superior de YOLOv8 en comparación con Faster R-CNN.

En el trabajo de Butt et al. (2023) se presenta un modelo de detección automática de puntuación en deportes de tiro utilizando técnicas de detección de objetos. Se comparó el rendimiento de siete modelos de detección de objetos, incluyendo tres variantes de YOLOv8. Cinco de los modelos son detectores de una etapa, mientras que dos pertenecen a la categoría de detectores de dos etapas. Se utilizaron dos variantes de Faster R-CNN y dos variantes de RetinaNet con Detectron2. Los modelos entrenados fueron evaluados en un conjunto de datos de prueba y comparados utilizando métricas como mAP50, mAP50-90, precisión y recall. Se encontró que los modelos YOLOv8 pueden detectar múltiples objetos con buenos puntajes de confianza. Entre estos modelos, YOLOv8s tuvo el mejor desempeño, con un mAP50 de 96.5% y un mAP50-95 de 64.6%. En comparación, Faster R-CNN 101 alcanzó un mAP50 de 83.554% y un mAP50-95 de 50.090%, mientras que RetinaNet 101 obtuvo un mAP50 de 66.416% y un mAP50-95 de 41.488%. Se sugiere que, para implementación en tiempo real, YOLOv8s es una mejor opción debido a su menor tiempo de inferencia y un competitivo mAP50.

2.2 Bases teóricas

2.2.1 Olea europea L.

Según Gómez y Vidal (2006), el olivo (Olea europea) perteneciente a la familia de las Oleáceas, puede alcanzar alturas de hasta quince metros si no se poda adecuadamente. Esta especie, dentro de la que se encuentra el olivo común, Olea europea L., es una de las seiscientas especies que conforman los veintinueve géneros en esta familia. Para distinguirlos de la variedad silvestre, conocida como acebuche (O. europea

var. Silvestris), los olivos cultivados a menudo se mencionan como O. europea var. Communis. Un olivar no solo consiste en los árboles de olivo, sino que incluye recursos adicionales como el suelo, las plantas espontáneas, el agua de lluvia y los insectos beneficiosos. Además, puede integrar otros elementos como plantas cultivadas dentro de su área o preservadas en sus límites, el agua de riego y animales domésticos.

2.2.1.1 Aceituna

Según Gómez y Vidal (2006), estos frutos llamados drupas, son carnosos y tienen un hueso en su interior, midiendo entre 0.5 y 2 cm de longitud. Contienen una semilla protegida por un endocarpo endurecido y cuando alcanzan su madurez se tornan negros. La evaluación de estas frutas se realiza considerando su peso, forma, forma del extremo y de la base, simetría, posición del diámetro máximo, presencia de características como el pezón y las lenticelas (regiones en la superficie para el intercambio de gases), así como su tamaño. Tanto las particularidades de la fruta como las del endocarpo son cruciales para distinguir entre las distintas variedades del olivo.

2.2.2 Inteligencia artificial

La inteligencia artificial según Callier y Sandel (2021), engloba investigaciones y análisis informáticos destinados a crear software o herramientas capaces de emular la inteligencia humana. Su objetivo es recrear aspectos como el razonamiento, el procesamiento de información y el aprendizaje, con la visión de alcanzar en el futuro una inteligencia comparable a la humana.

Según Hoyo Dorado (2021), la inteligencia artificial, referida como IA, se define como el conjunto de métodos y técnicas que capacitan a las computadoras para imitar capacidades humanas.

2.2.3 Aprendizaje automático

Peña Moliner (2020) indica que el concepto de aprendizaje automático, conocido como machine learning, cambió radicalmente la forma de pensar de la programación clásica a la hora de hacer un programa. En la clásica, alguien proporciona los datos y define las reglas del programa para obtener una respuesta. En el aprendizaje automático, en cambio, alguien da los datos con las respuestas y la máquina crea sus propias reglas. Estas reglas pueden aplicarse a nuevos datos para generar nuevas respuestas.

2.2.4 Aprendizaje profundo

Según Peña Moliner (2020), el aprendizaje profundo, conocido como deep learning, surge del campo del aprendizaje automático, ofreciendo una nueva perspectiva sobre cómo los datos pueden ser procesados para aprender. Se centra en la idea de aprender de manera progresiva, donde cada capa de procesamiento aprende representaciones cada vez más abstractas. Por ejemplo, las primeras capas pueden aprender a detectar características simples como líneas y curvas, mientras que las capas más profundas pueden reconocer patrones complejos como rostros o formas de animales. La profundidad de un modelo se refiere al número de capas que posee. En su mayoría, los modelos de deep learning se basan en redes neuronales.

2.2.5 Redes neuronales artificiales

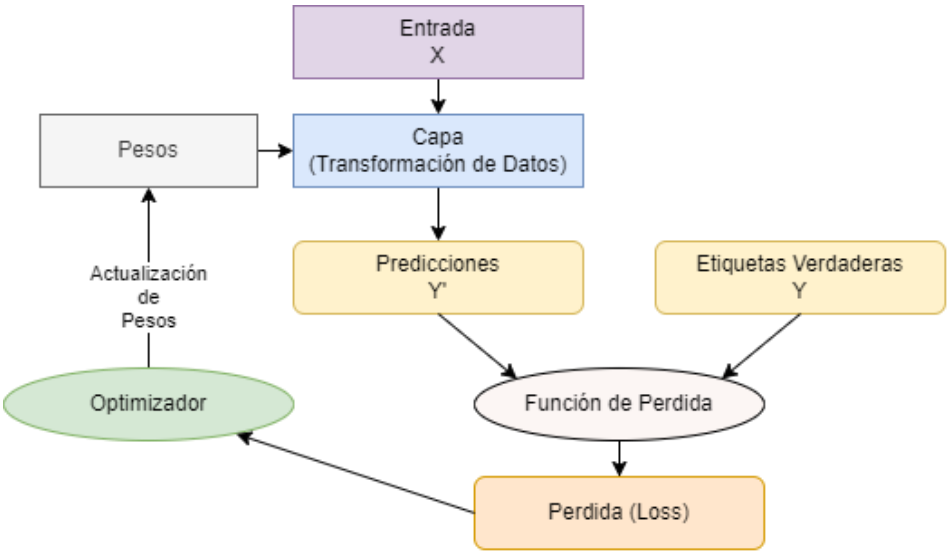
Según Janiesch et al. (2021), las redes neuronales artificiales (RNA), tienen la capacidad de emular conexiones similares al funcionamiento del cerebro humano. Estas redes tienen la habilidad de segmentar y categorizar conjuntos de datos, identificando correlaciones entre ellos. A medida que el sistema o la máquina adquieren nuevos conocimientos sin intervención humana, aplican esta información a otros conjuntos de datos. Cuanto más extenso sea el conjunto de datos con el que el sistema trabaja, mayor precisión tendrán sus predicciones.

Peña Moliner (2020) menciona que los elementos que componen una red neuronal para realizar su funcionamiento y aprendizaje son los siguientes:

- Capas: Que realizan las transformaciones de los datos indicadas por sus pesos.
- Datos de entrada: Para esta investigación, las imágenes con las etiquetas correspondientes en el formato adaptado al modelo de entrenamiento.
- Función de pérdida: Define el valor que se utiliza como retroalimentación para la ganancia.
- Optimizador: Define la forma en que se va a aprender.

Figura 1

Flujo de funcionamiento de una red neuronal



La Figura 1 muestra el flujo de funcionamiento de una red neuronal. En este diagrama, los datos de entrada atraviesan las distintas capas del modelo de deep learning utilizado. Cada capa realiza una transformación en los datos, la cual está determinada por los pesos asociados. En la salida de la red se generan las predicciones, las cuales se comparan con los datos reales. Este proceso se lleva a cabo utilizando una función de pérdida, cuyo objetivo es medir la discrepancia entre las predicciones y los datos reales. Esta discrepancia se representa como una puntuación de pérdida, que luego se utiliza en el proceso de optimización. El optimizador ajusta los valores de los pesos en todas las capas de la red con el fin de minimizar la puntuación de pérdida. Por lo tanto, el proceso de aprendizaje se basa en la adaptación de los pesos de la red para mejorar su capacidad predictiva. En resumen, el conocimiento se encuentra codificado en los pesos de la red neuronal.

2.2.6 Red neuronal convolucional

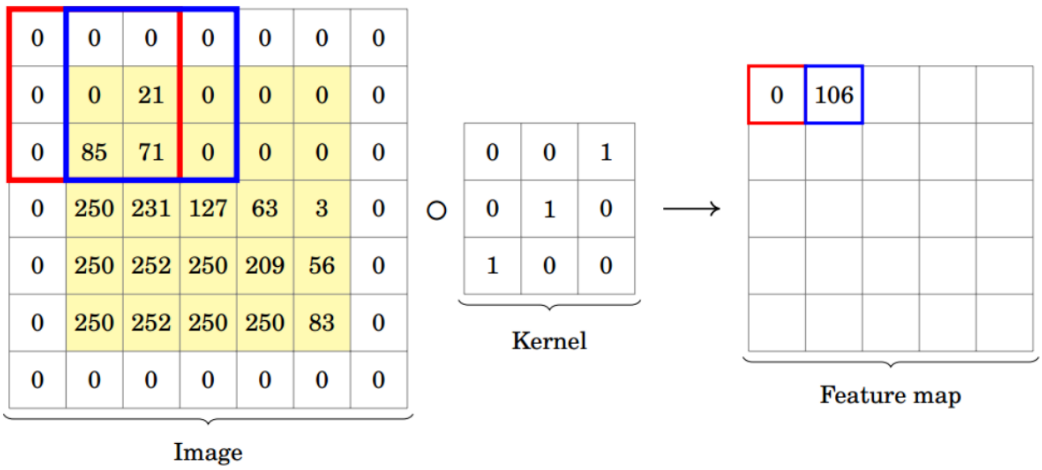
Según Peña Moliner (2020), cuando se analizan las imágenes, una de las soluciones con más éxito es la CNN (red neuronal convolucional) y la principal diferencia con las redes neuronales densamente conectadas es que la capa densa aprende patrones globales en su espacio de entrada global, mientras que las capas convolucionales aprenden patrones locales en pequeñas ventanas bidimensionales.

2.2.6.1 Capa convolucional

Según IBM (s.f.a), la capa convolucional constituye el componente fundamental de una convolutional neural network (CNN), donde se llevan a cabo la mayor parte de los cálculos. Esta capa requiere varios elementos, como los datos de entrada, un filtro y un mapa de características. Supongamos que la entrada es una imagen en color, que se representa mediante una matriz tridimensional de píxeles, donde cada dimensión corresponde a la altura, la anchura y la profundidad de la imagen, respectivamente, reflejando la composición RGB. El filtro, también denominado detector de características o kernel, se mueve a través de regiones de la imagen para verificar la presencia de ciertas características, en un proceso conocido como convolución. El detector de características consiste en una matriz bidimensional de pesos que representa una porción de la imagen. Aunque el tamaño del filtro puede variar, es común que tenga dimensiones de 3x3, lo que también determina el tamaño de la región receptiva. Posteriormente, el filtro se aplica a una región de la imagen y se realiza un producto escalar entre los valores de los píxeles de la entrada y los pesos del filtro. Este resultado se registra en una matriz de salida. Luego, el filtro se desplaza a lo largo de la imagen y se repite el proceso hasta que haya explorado toda la imagen. El resultado final de esta serie de productos escalares se conoce como mapa de características, mapa de activación o característica convolucionada. Un ejemplo de convolucional puede verse en la Figura 2.

Figura 2

Ejemplo de capa convolucional



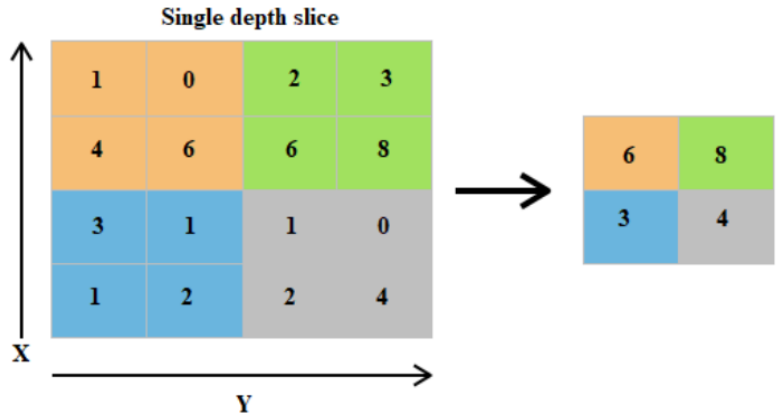
Nota. Adaptado de *Single Shot Multi Box Detector Approach to Autonomous Vision-Based Pick and Place Robotic Arm in the Presence of Uncertainties* (p. 07), por Chemelil, P. K., 2021.

2.2.6.2 Capa de agrupación

Peña Moliner (2020) indica que las capas de agrupación (pooling layers) suelen aplicarse después de las capas convolucionales y se utilizan para simplificar la información obtenida de las convoluciones generando una versión condensada de las mismas. Existen diferentes versiones para condensar la información, como el max-pooling que consiste en seleccionar el valor máximo de la ventana nxn. Otra alternativa podría ser el pooling promedio (average pooling). Un ejemplo de capa de agrupación se ve en la Figura 3.

Figura 3

Ejemplo de capa de agrupación (max-pooling)



Nota. Adaptado de *Single Shot Multi Box Detector Approach to Autonomous Vision-Based Pick and Place Robotic Arm in the Presence of Uncertainties* (p. 09), por Chemelil, P. K., 2021.

2.2.7 Descenso de gradiente estocástica

SGD por sus siglas en inglés, es un método de optimización ampliamente utilizado en el entrenamiento de redes neuronales. A diferencia del método de descenso de gradiente clásico, que calcula el gradiente de la función de pérdida con respecto a todos los datos de entrenamiento en cada actualización, el SGD realiza actualizaciones para cada muestra de datos de entrenamiento de manera individual. Esta característica hace que el SGD sea particularmente eficiente para grandes conjuntos de datos, ya que reduce significativamente el tiempo de cómputo por actualización (Goodfellow et al., 2016).

2.2.8 Detección de objetos

Según Zhao et al. (2019), la detección de objetos implica identificar un objeto en una imagen y determinar su posición.

2.2.8.1 Detección de objetos en dos etapas

Du et al. (2020) mencionan que el enfoque de dos etapas (two-stage), aunque más preciso, tiende a ser más complejo. Este método implica dos etapas distintas. En la primera etapa, se llevan a cabo pruebas iniciales para eliminar muestras negativas y se generan regiones de interés.

En la segunda etapa, el algoritmo realiza una clasificación regional y refina la localización basándose en las regiones de interés identificadas previamente. Entre los algoritmos de dos etapas más conocidos se encuentran Faster R-CNN, R-FCN, FPN, entre otros.

2.2.8.2 Detección de objetos en una etapa

Peña Moliner (2020) indica que el enfoque de una etapa (one-stage) se desarrolló para mejorar el rendimiento y acelerar el proceso de detección de múltiples objetos con un coste computacional inferior al de las arquitecturas de dos etapas. Estos sistemas conocidos también como one-shot realizan la detección de las diferentes regiones de la imagen original basándose en la idea del FASTer RCNN RPN, pero además añaden una nueva red paralela para obtener la clasificación de los objetos. Entre ellos están YOLO (You Only Look Once) y RetinaNet.

2.2.9 ResNet-101

Según Ghosal et al. (2019), ResNet-101 está inspirada en el modelo VGG-19 y es una de las arquitecturas más profundas utilizadas en el desafío ImageNet para detección de objetos y clasificación de imágenes. Esta red residual de 101 capas utiliza filtros de 3x3 y duplica el número de filtros cuando se reduce a la mitad el tamaño del mapa de características para mantener la complejidad temporal. Realiza la reducción de tamaño aplicando una convolución y termina con una capa de agrupación promedio global y una capa softmax. Las conexiones de atajo permiten evitar el problema de los gradientes que desaparecen, facilitando el entrenamiento y mejorando la precisión.

2.2.10 Feature pyramid network

Peña Moliner (2020) menciona que en redes como VGG o ResNet se ha visto que las capas más superficiales se encargan de calcular características de bajo nivel, como contornos, líneas, etc. Mientras que las capas más profundas, con menor resolución, contienen elementos semánticos de mayor nivel que permiten la identificación de objetos. Sin embargo, existe una correlación entre nivel semántico y resolución por la que un aumento del anterior disminuye el segundo.

La arquitectura FPN pretende resolver este problema introduciendo en la columna vertebral de una ResNet, una segunda estructura descendente en la que las características semánticas de alto nivel se propagan a las capas inferiores.

2.2.11 YOLOv8

Según Reis et al. (2024), YOLO es un algoritmo de detección de objetos en tiempo real, ha evolucionado desde su primera versión (YOLOv1) en 2015 hasta la más reciente (YOLOv8). La versión original se destacó por su enfoque simplificado y su capacidad para procesar imágenes rápidamente, logrando una alta precisión en la detección de objetos. YOLOv8 mantiene la arquitectura básica de sus predecesores, pero introduce mejoras importantes, como una nueva estructura de red neuronal que utiliza feature pyramid network (FPN) y path aggregation network (PAN). Estas mejoras permiten una detección más precisa de objetos de diferentes tamaños y formas al combinar características de múltiples escalas y resoluciones.

Sportelli et al. (2023) mencionan que YOLO realiza la detección de objetos dividiendo la imagen de entrada en una cuadrícula de $m \times m$ celdas de igual tamaño.

Una ilustración de la arquitectura puede observarse en la Figura 4 y la función de pérdida de YOLOv8 se muestra en la Ecuación 1.

$$L = \frac{\lambda box}{Npos}\sum_{x,y} \mathbb{1}c^*_{x,y}\left[1 - q_{x,y} + \frac{\|b_{x,y} - \hat{b}_{x,y}\|_2^2}{p^2} + a_{x,y}v_{x,y}\right] +$$

$$\frac{\lambda cls}{Npos}\sum_{x,y}\sum_{c\epsilon classes} y_c \log(\hat{y}_c) + (1 - y_c)\log(1 - \hat{y}_c) + \frac{\lambda dfl}{Npos}\sum_{x,y}\mathbb{1}c^*_{x,y}\left[-\left(q_{x,y+1} - q_{x,y}\right)\log\left(\hat{q}_{x,y}\right) + \left(q_{x,y} - q_{(x,y)-1}\right)\log\left(\hat{q}_{(x,y)+1}\right)\right] \quad (1)$$

Donde:

$q_{x,y} = IoU_{x,y}$

$$v_{x,y} = \frac{4}{\pi^2}(\arctan\left(\frac{w_{x,y}}{h_{x,y}}\right) - \arctan\left(\frac{\hat{w}_{x,y}}{\hat{h}_{x,y}}\right))^2$$

$$a_{x,y} = \frac{v}{1-q_{x,y}}$$

$$\hat{y}_c = \sigma(\cdot)$$

$$\hat{q}_{x,y} = softmax(\cdot)$$

$Npos$ es el número total de celdas que contienen un objeto.

$\mathbb{1}c^*_{x,y}$ es una función indicadora de las celdas que contienen un objeto.

$b_{x,y}$ es una tupla que representa el punto central de caja delimitadora (bounding box) real.

y_c es la etiqueta real para la clase c para cada celda de cuadrícula individual (x,y) en la entrada, independientemente de si hay un objeto presente.

$q_{(x,y)+/-1}$ son IoU de las cajas predichas más cercanas (izquierda y derecha).

$w_{x,y}$ y $h_{x,y}$son la anchura y la altura de las cajas.

p es la longitud diagonal de la caja más pequeña que cubre las cajas predichas y las reales.

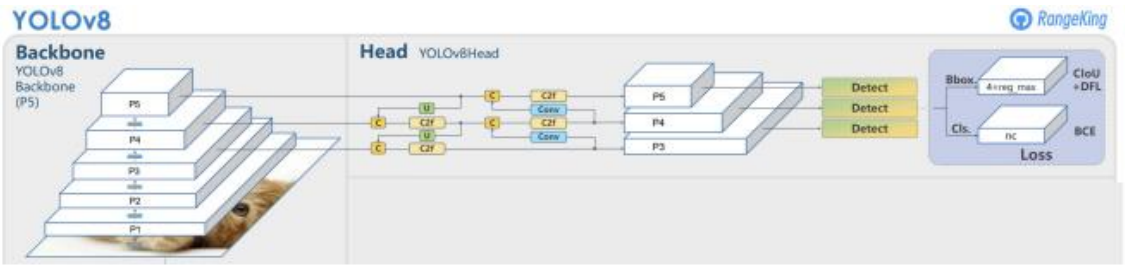
λbox se refiere al peso asignado a la pérdida de las cajas delimitadoras.

λcls se refiere al peso asignado a la pérdida de clasificación.

λdfl se refiere al peso "distribution focal loss" asignado.

Figura 4

Arquitectura de YOLOv8



Nota. Adaptado de "Real-Time Flying Object Detection with YOLOv8" (p. 08), por D. Reis et al., 2024, *arXiv*.

Jocher et al. (2023) mencionan que YOLOv8 fue lanzado por Ultralytics en enero de 2023. Soporta múltiples tareas de visión computacional tales como detección de objetos, segmentación semántica, estimación de poses, seguimiento y clasificación. YOLOv8 cuenta 5 versiones de diferentes escalas: nano (YOLOv8n), small (YOLOv8s), medium (YOLOv8m), large (YOLOv8l), extra large (YOLOv8x). Estas versiones

comparten un enfoque arquitectónico común, pero difieren en la profundidad y anchura de sus capas.

2.2.11.1 YOLOv8s

La versión YOLOv8s presenta las características vistas en la Tabla 2.

Tabla 2

Características de arquitectura YOLOv8s

depth	width	max_channels	layers	parameters	gradients	GFLOPs
0.33	0.50	1024	225	11166560	11166544	28.8

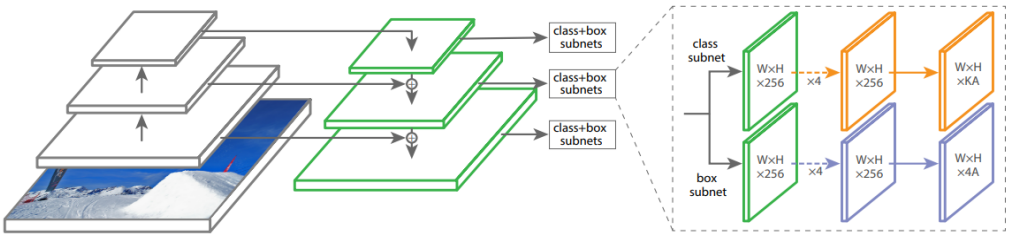
Nota. Datos obtenidos de Jocher et al. (2023).

2.2.12 RetinaNet

Según Lin et al. (2017), RetinaNet es un modelo de red neuronal único y cohesivo que consta de una red base y dos subredes específicas para tareas particulares. La red base se encarga de generar un mapa de características convolucionales para toda la imagen de entrada y se basa en una red convolucional preexistente. Por otro lado, la primera subred realiza la clasificación convolucional de objetos utilizando la salida de la red base, mientras que la segunda subred se encarga de la regresión convolucional de los cuadros delimitadores. Ambas subredes están diseñadas de manera sencilla, especialmente orientadas para la detección densa en una sola etapa. A pesar de que hay muchas opciones disponibles para los detalles de estos componentes, la mayoría de los parámetros de diseño no son particularmente sensibles a valores específicos. La arquitectura de RetinaNet se muestra en la Figura 5.

Figura 5

Arquitectura RetinaNet



Nota: Adaptado de “Focal Loss for Dense Object Detection” (p. 05), por T. Lin et al., 2017, *arXiv.*

2.2.12.1 RetinaNet 101

Según W. (2020), la arquitectura del modelo se fundamenta en una "feature pyramid network" sobre una red ResNet-101 de alimentación directa. Durante el entrenamiento del modelo, se ha implementado una nueva función de pérdida denominada "pérdida focal" mostrada en la Ecuación 2, diseñada para mitigar el desequilibrio entre las clases de primer plano y de fondo que comúnmente se presenta en los detectores de una sola etapa.

$$L = -\sum_{i=1}^{k}\left(y_i \log(p_i)\,(1-p_i)^{\gamma} a_i + (1-y_i)\log(1-p_i) p_i^{\gamma}(1-a_i)\right) \quad (2)$$

Donde:

k representa el número de clases.

y_i toma el valor unitario cuando el objeto se corresponde con la clase.

p_i representa la probabilidad del objeto i.

γ es el parámetro denominado pérdida focal, se utiliza para dar menos importancia a las clases más fáciles de clasificar, como es el caso del fondo. Es decir, las más frecuentes.

a_i es una forma alternativa de equilibrar los datos. Se asigna un valor a las clases que aparecen con más frecuencia.

2.2.13 Faster R-CNN

Según Ren et al. (2016), Faster R-CNN es una arquitectura diseñada para abordar la generación de propuestas de regiones de interés (ROIs) de manera eficiente y precisa en la detección de objetos. Se compone de tres elementos principales:

- Red convolucional compartida: Emplea una red convolucional profunda, como VGG o ResNet, para la extracción de características de la imagen usada como entrada. Esta red convolucional se comparte entre los módulos de generación de regiones propuestas (RNP) y el de detección.
- Red de propuesta de regiones (RPN): Introduce una red especializada que opera sobre la característica convolucional compartida para generar propuestas de regiones de interés (ROIs). la RPN utiliza "anchors" de diferentes aspectos y escalas para proponer ROIs y predice puntajes de objetividad para cada "anchor" y cuadros delimitadores.

- Módulo de detección: Utiliza las propuestas que son generadas por la RPN para clasificar y detectar objetos en categorías específicas. En esta etapa se utilizan las características que fueron extraídas por la red convolucional compartida y las ROIs que son propuestas por la RPN para ajustar los cuadros delimitadores y predecir las clases de objetos para cada objeto detectado.

La función de pérdida multitarea de Faster R-CNN está definida en la Ecuación 3 y su arquitectura está representada en la Figura 6.

$$L(\{p_i\},\{t_i\}) = \frac{1}{Ncls}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{Nreg}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (3)$$

Donde:

i es el índice de un "anchor" en un lote

p_i es la probabilidad prevista de que el "anchor" i sea un objeto

p_i^* es la etiqueta verdadera, vale 1 si el "anchor" es positiva y 0 si es negativa.

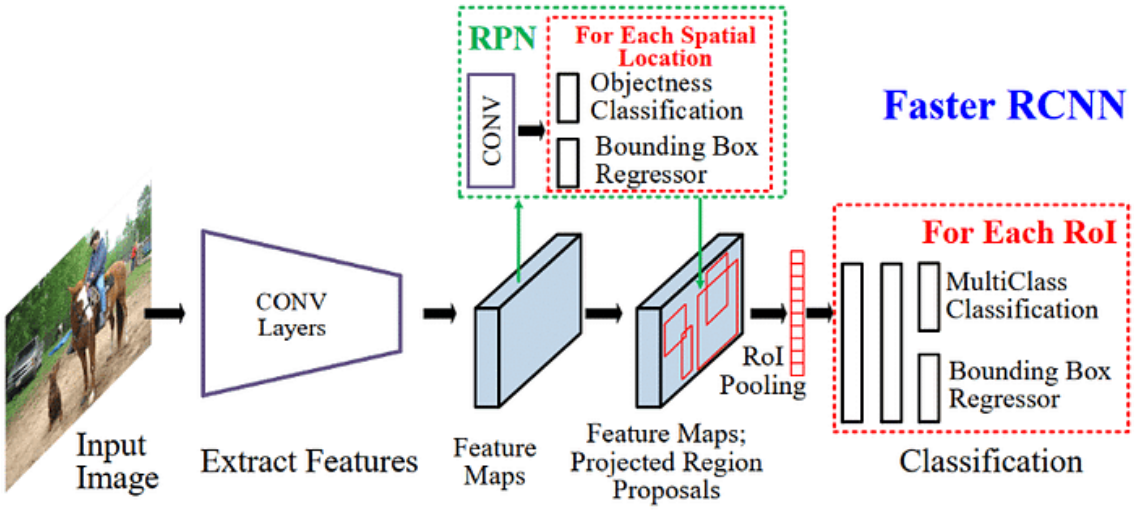
t_i es un vector que representa las 4 coordenadas parametrizadas de la caja delimitadora predicha.

t_i^* es la caja delimitadora real asociada a un "anchor".

λ es un peso asignado para equilibrar los datos.

Figura 6

Arquitectura de Faster R-CNN 101



Nota: Adaptado de "Real-Time Diseases Detection of Grape and Grape Leaves using Faster R-CNN and SSD MobileNet Architectures" (p.41), por S. Ghoury et al., 2019, *International conference on advanced technologies, computer engineering and science*, 2019.

2.2.13.1 Faster R-CNN 101

Zhou et al. (2020) mencionan que en Faster R-CNN 101 la extracción de características se lleva a cabo inicialmente mediante una ResNet-101. Posteriormente, se emplea una red de propuesta de regiones para generar un cuadro delimitador. Este cuadro de propuesta es mapeado en el mapa de características de la última capa de la red neuronal convolucional (CNN). Luego, la capa de agrupación de regiones de interés produce un mapa de características de tamaño fijo. Su arquitectura se observa en la Figura 6.

2.2.14 COCO "Common objects in context"

Lin et al. (2014) mencionan que COCO (Common Objects in Context) es un conjunto de datos diseñado para la detección, segmentación y clasificación de objetos en imágenes. Contiene imágenes de escenas cotidianas con anotaciones precisas y detalladas de objetos en 80 categorías diferentes. Las imágenes de COCO son diversas en términos de contexto y complejidad, lo que permite a los investigadores evaluar algoritmos en una variedad de situaciones del mundo real. Además, el conjunto de datos COCO proporciona una evaluación exhaustiva y puntos de referencia para diversas tareas de visión por computadora, lo que lo convierte en un recurso invaluable para el desarrollo y la evaluación de algoritmos en este campo.

2.2.15 Métricas

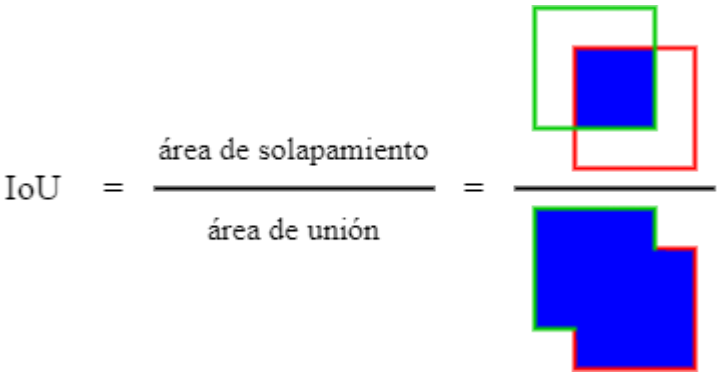
2.2.15.1 IoU

Según Padilla et al. (2020), el IoU mide el área de solapamiento entre el cuadro delimitador previsto y el cuadro delimitador real dividido por el área de unión entre ambos.

Comparando el IoU con un umbral "t" determinado, podemos clasificar una detección como correcta o incorrecta. Si el valor del IoU es mayor o igual a t, la detección se considera correcta; si el valor de IoU es menor a t, se considera incorrecta. La Figura 7 ilustra esta operación.

Figura 7

Cálculo de IoU



2.2.15.2 Precisión (Precision)

Se refiere a la habilidad de un modelo para reconocer exclusivamente los objetos pertinentes en una escena **(**Padilla et al., 2020). Este indicador está determinado por la Ecuación 4.

$$P = \frac{TP}{TP+FP} = \frac{TP}{Todas\ las\ detecciones} \quad (4)$$

Donde:

P: Precisión

TP: Verdaderos positivos, detección correcta para un recuadro de delimitación

FP: Falsos positivos, detección incorrecta de un objeto que no existe o detección incorrecta de un objeto existente.

2.2.15.3 Recuperación (Recall)

Recuperación se refiere a la habilidad de un modelo para detectar todos los casos pertinentes en una situación dada (Padilla et al., 2020). Este indicador está determinado por la Ecuación 5.

$$R = \frac{TP}{TP+FN} = \frac{TP}{Detecciones\ reales} \quad (5)$$

Donde:

R: Recuperación (Recall)

TP: Verdaderos positivos, detección correcta para un recuadro de delimitación

FN: Falsos negativos, un cuadro delimitador que no fue detectado

2.2.15.4 Curva de precisión-recuperación

La curva precisión-recuperación puede considerarse como un balance entre precisión y recuperación para distintos valores de confianza asociados a los recuadros delimitadores generados por un detector (Padilla et al., 2020).

2.2.15.5 Precisión media (Average precision)

La precisión media (AP) es el área bajo la curva de precisión-recuperación, este indicador está entre 0 y 1 para agrupar los distintos valores de precisión en un único valor que representa la media de todas las precisiones (Butt et al., 2023).

Según Wang (2022), se pueden utilizar todos los puntos de la curva de precisión-recuperación para calcular el AP o bien puntos específicos que correspondan a una secuencia de niveles de recuerdo igualmente espaciados que vayan de 0 a 1.

Jayawardena y Jayarathna (2020) mencionan que en COCO se calcula un promedio de AP con 101 puntos de interpolación. Este indicador se describe en la Ecuación 6.

$$AP = \sum(R_{n+1} - R_n) \times P_{interp}(R_{n+1}) \quad (6)$$

Donde:

AP: Precisión media

n: El número del par de precisión-recuperación

R: Recuperación

P: Precisión

$P_{interp}(R_{n+1})$: La máxima precisión cuyo valor de recuperación sea mayor o igual que (R_{n+1}).

El número de pares estará determinado por el número de detecciones.

Todas las métricas se calculan teniendo en cuenta un máximo de 100 detecciones por imagen (COCOdataset, s.f.).

2.2.16 Indicadores

2.2.16.1 Mean average precision (mAP)

Según Wang (2022), mAP ha sido ampliamente utilizada para evaluar la calidad de modelos de detección de objetos. Competiciones destacadas de detección de objetos como Pascal VOC y Microsoft COCO adoptan mAP como métrica de evaluación estándar.

El valor de Mean Average Precision (mAP) se calcula tomando la media de los valores AP de todas las clases (Butt et al., 2023), está determinado por la Ecuación 7.

$$mAP = \frac{1}{n}\sum_{i=1}^{n}[AP_i] \quad (7)$$

Donde:

mAP: Mean Average Precision

AP: El valor de precisión media

n: El número total de clases

Cabe mencionar que en el contexto del conjunto de datos COCO, ambas denominaciones, AP y mAP, se refieren a la misma métrica "mean average precision". La página de COCO menciona: No distinguimos entre AP y mAP y suponemos que la diferencia se deduce claramente del contexto (COCOdataset, s.f.).

Para este trabajo se usarán dos versiones de este indicador: mAP50 y mAP50-95.

2.2.16.2 mAP50

Ren et al. (2016) mencionan que mAP@0.5 (mAP50) es la métrica de la competencia PASCAL VOC.

Según Zhang et al. (2024), mAP50 se refiere a la precisión media al 50% de intersección sobre unión (IoU). Agrega que se trata de una métrica de evaluación habitual utilizada en tareas de detección de objetos para medir el rendimiento general del modelo.

Según Wang et al. (2021), mAP50 es el promedio de AP50 (precisión media) de los resultados de detección para todas las clases. El valor AP50 es el área bajo la curva de precisión y recuperación con un umbral IoU de 0,5.

2.2.16.3 mAP50-95

Bell et al. (2016) mencionan que la competencia asociada a Microsoft COCO usa una métrica de evaluación que promedia el mAP sobre diferentes umbrales de IoU, desde 0.5 hasta 0.95. Esta métrica pone un mayor énfasis en la localización a comparación de la métrica de PASCAL VOC, que solo requiere un IoU de 0.5.

La métrica estándar de COCO es mAP promediado sobre umbrales IoU en [0,5:0,05:0,95] (Huang et al. 2017; Ren et al., 2016).

Según Reis et al. (2024), para obtener mAP50-95, tomamos pasos de 0,05 partiendo de un umbral IoU de 0,5 y deteniéndonos en 0,95. La precisión media en este intervalo es el AP de una clase. Si hacemos esto para todas las clases y sacamos la media de todas ellas, generamos el mAP50-95.

2.3 Definiciones conceptuales

2.3.1 Modelo

Un modelo de inteligencia artificial es un programa que ha sido entrenado en un conjunto de datos para reconocer determinados patrones o tomar ciertas decisiones sin más intervención humana (IBM, s.f.b).

2.3.2 Época

Cada iteración sobre todos los datos de entrenamiento se denomina época. En cada época, la red calculará los gradientes de los pesos con respecto a la pérdida en el lote, y actualizará los pesos en consecuencia (Chollet, 2017, p. 53).

2.3.3 Tamaño de lote

El tamaño del lote “batch size” es el número de muestras de entrenamiento utilizadas en una repetición (Firdaus, 2021).

2.3.4 Conjunto de entrenamiento

Conjunto de datos sobre el que ejecutas tu algoritmo de aprendizaje (Ng, 2017, p. 15).

2.3.5 Conjunto de validación

Un conjunto de datos de validación es un subconjunto de los datos de entrenamiento que se ocultan a los algoritmos de aprendizaje automático hasta el final del proyecto (Brownlee, 2016, p. 24).

2.3.6 Aprendizaje por transferencia

El aprendizaje por transferencia es una técnica de aprendizaje automático en la cual se emplea un modelo desarrollado para una tarea inicial como base para abordar una segunda tarea de aprendizaje (Tan et al., 2018).

El aprendizaje por transferencia ha sido una de las técnicas más potentes a lo largo de la historia del aprendizaje profundo. El paradigma del pre-entrenamiento en conjuntos de datos de gran escala y el ajuste en conjuntos de datos o tareas de destino se aplica a una amplia gama de tareas tanto en visión por ordenador como en procesamiento del lenguaje natural (Bu et al., 2021).

2.3.7 Inferencia

La inferencia es el proceso de pasar datos reales por un modelo de inteligencia artificial entrenado para hacer una predicción o resolver una tarea (Martineau, 2023).

2.3.8 Detectron2

Fue introducido por Facebook AI Research en 2019 como una versión mejorada del marco original de Detectron. Este framework está principalmente dirigido a la detección de objetos y la segmentación de instancias (Butt et al., 2023).

2.3.9 CUDA

Proporciona un entorno de desarrollo para crear aplicaciones de alto rendimiento aceleradas en la unidad de procesamiento gráfico. Incluye librerías para la aceleración con la unidad de procesamiento gráfico, herramientas de depuración y optimización (NVIDIA, s.f.).

2.3.10 JupyterLab

JupyterLab es una plataforma ampliamente utilizada para la programación y la ciencia de datos mediante cuadernos computacionales “notebooks” (Olney y Fleming, 2021).

2.3.11 Label-Studio

Label-Studio es una herramienta de código abierto para el etiquetado de datos. Permite etiquetar datos como texto, audio, vídeos, imágenes y series temporales con una interfaz de usuario sencilla y directa y exportar a varios formatos de modelo. Puede utilizarse para preparar datos sin procesar o mejorar los datos de entrenamiento existentes para obtener modelos más precisos (Tkachenko et al., 2020).

2.3.12 Cuadro delimitador

En la detección de objetos, solemos utilizar un cuadro delimitador "bounding box" para describir la ubicación espacial de un objeto (Zhang, 2021).

2.3.13 Recuadros de anclaje

Los algoritmos de detección de objetos examinan múltiples regiones en una imagen para identificar objetos de interés y ajustar los límites de estas regiones para mejorar la precisión en la predicción de los cuadros delimitadores. Un enfoque común es generar cuadros delimitadores de diferentes tamaños y formas centrados en cada píxel, conocidos como recuadros de anclaje "anchor box" (Zhang, 2021).

CAPÍTULO III
MARCO METODOLÓGICO

3.1 Planteamiento metodológico

Según Hernández Sampieri et al. (2014), el enfoque cuantitativo se basa en el razonamiento deductivo o la lógica, que empieza con la teoría y de la cual el investigador deriva expresiones lógicas llamadas "hipótesis". Por ello, este trabajo adopta el enfoque cuantitativo.

Según Carrasco (2006), la investigación básica es la que no tiene propósitos aplicativos inmediatos, pues solo busca ampliar y profundizar el caudal de conocimientos científicos existentes acerca de la realidad.

La presente investigación es de tipo básica, puesto que solo se amplían los conocimientos acerca de comparación de rendimientos de diferentes modelos de detección de frutos de olivo.

3.1.1 Nivel

De acuerdo con Horna (2012), el propósito de la investigación descriptivo-comparativa radica en reconocer las variaciones o similitudes de un suceso entre dos o más conjuntos.

La presente investigación es descriptivo-comparativa, ya que se realizó la comparación del rendimiento en los modelos YOLOv8, RetinaNet, Faster R-CNN.

3.1.2 Diseño

Según Hernández Sampieri et al. (2014), la investigación no experimental se define como aquella en la que no se manipulan intencionadamente variables. En este tipo de estudios, no se busca alterar deliberadamente las variables independientes para observar su impacto en otras variables. En lugar de ello, se observan los fenómenos en su entorno natural con el propósito de analizarlos.

La presente investigación es no experimental ya que no se realiza la manipulación de la variable rendimiento de modelos de inteligencia artificial.

3.2 Población y muestra

3.2.1 Población

Según Hernández Sampieri et al. (2014) "Una población es el conjunto de todos los casos que concuerdan con una serie de especificaciones" (p.174). Por ende, nuestra población es el "conjunto de datos de recortes de imágenes de olivo" que comprende de 19,560 imágenes de recortes de árboles de olivos.

Estas imágenes fueron proporcionadas por el proyecto de investigación "Estimación de productividad de olivares usando técnicas de visión artificial y aprendizaje profundo en la región de Tacna" de la Universidad Nacional Jorge Basadre Grohmann, Tacna – Perú, este proyecto fue ejecutado en los años 2023 y 2024.

3.2.2 Muestra

Según Carrasco (2006), "Fragmento representativo de la población, que debe poseer las mismas propiedades y características de ella" (p.238).

3.2.2.1 Muestras no probabilísticas

Según Carrasco (2006), "En este tipo de muestras, no todos los elementos de la población tienen la probabilidad de ser elegidos para formar parte de la muestra" (p.243).

3.2.2.2 Muestras intencionadas

Según Carrasco (2006), "El investigador procede a seleccionar la muestra en forma intencional, eligiendo aquellos elementos que considera convenientes y cree que son los más representativos" (p.243).

La muestra para el presente trabajo de investigación fue de 10,728 imágenes de recortes de árboles de olivo. Se eligió esta muestra teniendo en cuenta el horario en que se tomaron las imágenes.

3.3 Equipos y materiales

Para la toma de imágenes se utilizó un flexómetro para medir la distancia entre el árbol de olivo y la cámara, un trípode y una cámara Canon EOS Rebel T6i con las

siguientes características: sensor óptico de 22.3 mm x 14.9 mm, resolución de imagen: 24.2 megapíxeles.

Para el procesamiento de las imágenes y el entrenamiento de modelos, se utilizó una workstation de alto rendimiento con las siguientes características: procesador Intel® Xeon® Silver 4214 2.20 GHz, memoria RAM: 64 GB DDR4 2933 366 MHz, almacenamiento de 1 TB SSD, unidad de procesamiento gráfico NVIDIA RTX A5000 de 24 GB.

Para el etiquetado se utilizó el programa Label Studio v14.3, para entrenar los modelos de Faster R-CNN y RetinaNET se utilizó el framework Detectron2 junto con el lenguaje de programación Python v3.9.0 y para el modelo YOLOv8 se utilizó el framework Ultralytics YOLOv8 junto con el lenguaje Python v3.10.12, se usó CUDA v12.2 y estos modelos fueron entrenados en el entorno de JupyterLab v4.0.6.

3.4 Procedimiento de las pruebas experimentales

Para este trabajo no es necesario emplear pruebas experimentales, ya que se trata de una investigación puramente descriptivo-comparativa.

3.5 Técnicas de recolección de datos

Según Hernández Sampieri et al. (2014), la observación es un método de recolección de datos que utiliza un conjunto de categorías y subcategorías para registrar de manera sistemática, válida y confiable los comportamientos y situaciones observables.

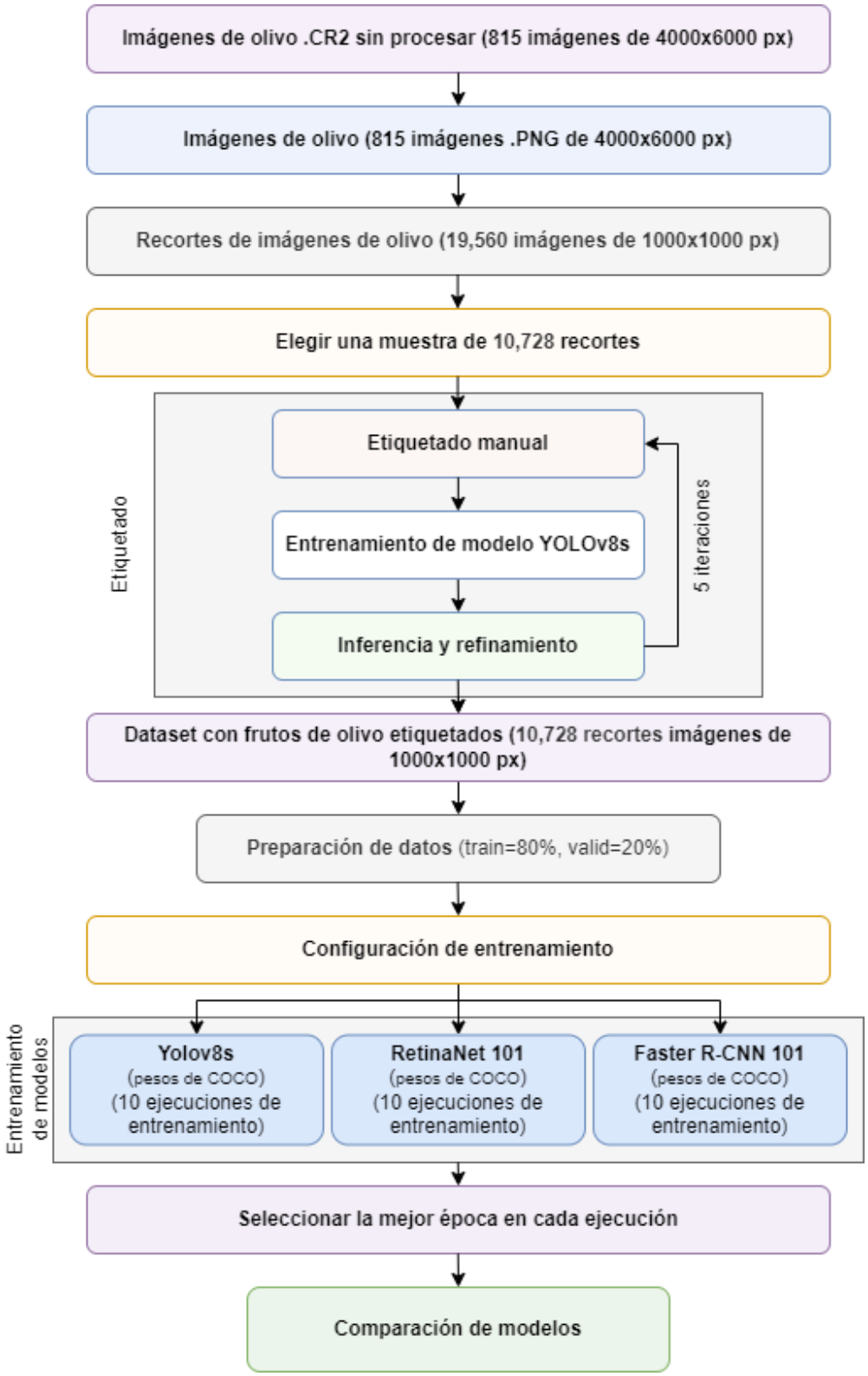
Esta investigación utiliza la observación como instrumento porque permite registrar sistemáticamente los resultados de los indicadores mAP50 y mAP50-95 de los modelos.

3.6 Técnicas para el procesamiento de datos

La Figura 8 presenta el flujo de trabajo seguido en esta investigación. En este diagrama, se ilustran las distintas etapas que comprenden desde la adquisición de datos hasta la comparación de modelos.

Figura 8

Flujo de trabajo



Se explicarán las tareas de procesamiento que comprenden todas las actividades hasta antes del entrenamiento del modelo.

Inicialmente, las imágenes de los árboles de olivo fueron tomadas en La Yarada Los Palos – Tacna, parcela 49 de la Asoc. Agroindustrial Frontera del Sur, ubicada con coordenadas 18°15'16.1"S 70°26'30.7"W. Una foto satelital del lugar puede verse en la Figura 9.

Se midió una distancia de cinco metros con cada árbol antes de tomar las fotos, se obtuvieron 815 imágenes sin procesar en formato .CR2.

Figura 9

Lugar de toma de fotografías

Nota: Adaptado de Google, s.f., https://maps.app.goo.gl/Y943nuFTw3cvodCc7 Todos los derechos reservados 2024 por Google.

Luego, en la primera etapa del flujo de trabajo, las imágenes obtenidas en formato .CR2 se transformaron al formato .PNG gracias a la librería Pillow en Python para facilitar su manejo y procesamiento. El código utilizado para realizar la tarea puede encontrarse en el anexo 07. Un ejemplo de estas imágenes puede verse en la Figura 10.

Figura 10

Ejemplo de fotografía original de olivo

Cada imagen originalmente tenía una resolución de 4000x6000 píxeles. Con el objetivo de poder usar estas imágenes en los modelos sin generar reducciones significativas debido al tamaño de entrada de las arquitecturas y evitar que los frutos en las imágenes se volvieran indetectables por tener un tamaño reducido, se realizaron recortes de cada imagen con un programa en Python. Puede verse el código de esta tarea en el anexo 07.

Se obtuvieron un total de 24 recortes de 1000x1000 píxeles por cada imagen. Este procedimiento dio como resultado un conjunto de 19,560 recortes. El tamaño total que alcanzó la carpeta con los recortes de olivo fue de 25 gigabytes y en la Figura 11 se muestra un ejemplo de estas imágenes.

Figura 11

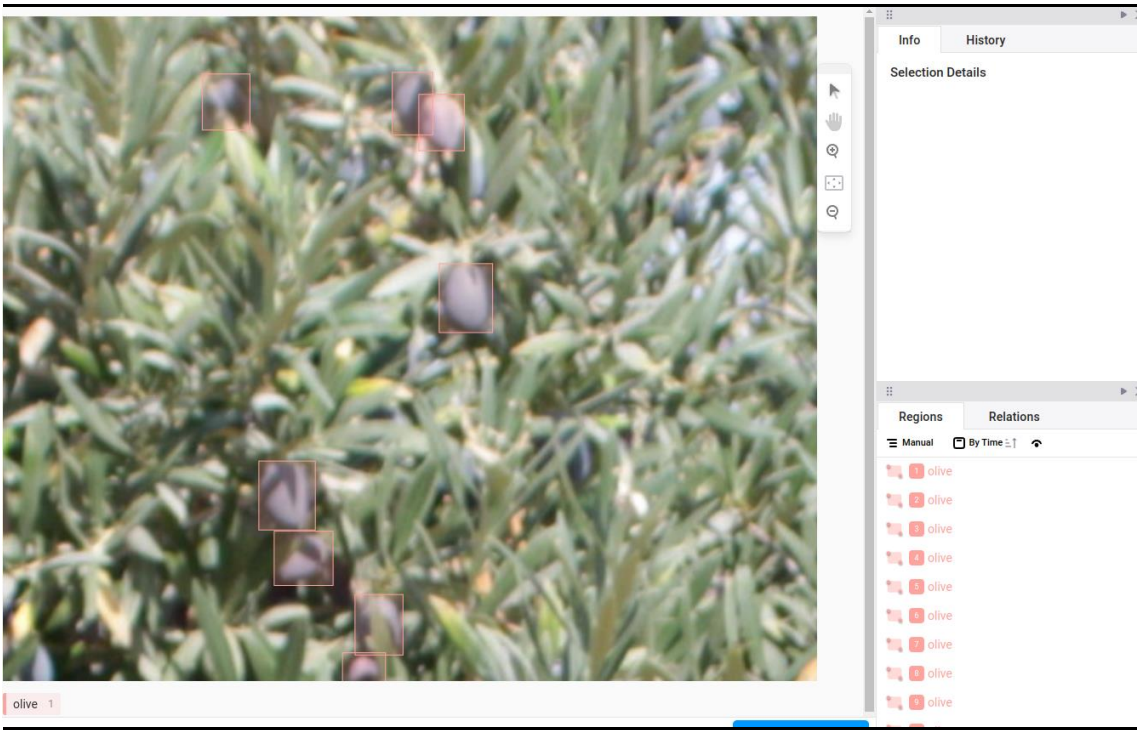
Ejemplo de recortes de imagen

Después de generar los recortes de las imágenes, se llevó a cabo la selección de una muestra representativa. Se optó por una muestra de 10,728 imágenes, considerando la variación en la iluminación según la hora de captura de las fotografías en una proporción equitativa. Este número también se seleccionó estratégicamente para optimizar el tiempo empleado en el proceso de etiquetado.

La etapa de etiquetado se inició con 480 imágenes, en las cuales se marcaron las aceitunas mediante cuadros delimitadores en Label-Studio. La Figura 12 muestra la tarea de etiquetado en Label-Studio.

Figura 12

Etiquetado en Label-Studio



Posteriormente, se entrenó un primer modelo de YOLOv8s con estas 480 imágenes. Se utilizó este modelo para realizar inferencias en un conjunto adicional de 576 imágenes. Luego se codificó un pequeño programa para poder generar un archivo .JSON , cuyo código se encuentra en el anexo 07. Este archivo se usó para importar las coordenadas de las cajas delimitadoras predichas por el modelo a Label-Studio, para posteriormente hacer una refinación manual para corregir posibles errores de etiquetado del modelo.

Este ciclo de entrenamiento, inferencia y refinamiento se repitió a lo largo de cinco iteraciones. Se incorporaron progresivamente más imágenes etiquetadas en cada iteración, resultando en modelos más precisos. La cantidad de imágenes usadas para el entrenamiento de modelos aumentó de 480 en la primera iteración hasta 5,064 en la quinta.

La duración total de este procedimiento fue de cuatro meses para poder llegar a etiquetar 10,728 imágenes de la muestra.

Luego del etiquetado de las 10,728 imágenes, se procedió a dividir el conjunto de datos en 80% para entrenamiento (8,549 imágenes) y 20% para validación (2,179 imágenes). Se generó una configuración del conjunto de datos compatible para el

entrenamiento de los modelos con sus respectivos frameworks. El tamaño del conjunto de datos final fue de 13.5 gigabytes.

Para la selección de las arquitecturas a usar en los modelos, se tomó en cuenta YOLOv8s por su eficiencia computacional y su capacidad para realizar detecciones precisas en tiempo real, además de ser un modelo reciente. La elección de RetinaNet 101 y Faster R-CNN 101 se justifica por su disponibilidad y facilidad de implementación en la plataforma de Detectron2, este framework proporciona implementaciones de modelos de vanguardia y herramientas para el desarrollo rápido de sistemas de visión por computadora. Además, tanto RetinaNet 101 como Faster R-CNN 101 son modelos bien establecidos y ampliamente utilizados en aplicaciones de detección de objetos.

En la Tabla 3 se muestran los resultados de mAP50-95 que tuvieron las arquitecturas escogidas al ser entrenadas y luego validadas en el conjunto de datos "COCO validation 2017" por sus respectivos autores, así como el número de parámetros con los que cuenta cada arquitectura.

Tabla 3

Arquitecturas de modelos escogidos

Modelo	**mAP50-95 COCOval2017**	**Número de parámetros**
YOLOv8s	44.9	11.2 millones
Faster R-CNN 101	43.0	105 millones
RetinaNet 101	40.4	57 millones

Nota: Datos obtenidos de Jocher et al. (2023) y Wu et al. (2019)

CAPÍTULO IV
RESULTADOS

4.1 Descripción de las pruebas experimentales

En el entorno de JupyterLab se procedió a configurar el entrenamiento de los modelos a los siguientes valores: 100 épocas, tamaño de lote de 4, ratio de aprendizaje de 0.01 y SGD como optimizador.

Se cargaron los modelos con los pesos pre-entrenados en el conjunto de datos COCO, esto para aplicar aprendizaje por transferencia.

Para finalmente, realizar las diez ejecuciones del entrenamiento con los tres modelos (YOLOv8s, Faster R-CNN 101, RetinaNet 101) a comparar y generar los reportes de los indicadores en cada época en un archivo .csv y guardar los modelos con mejor desempeño para su posterior análisis y evaluación.

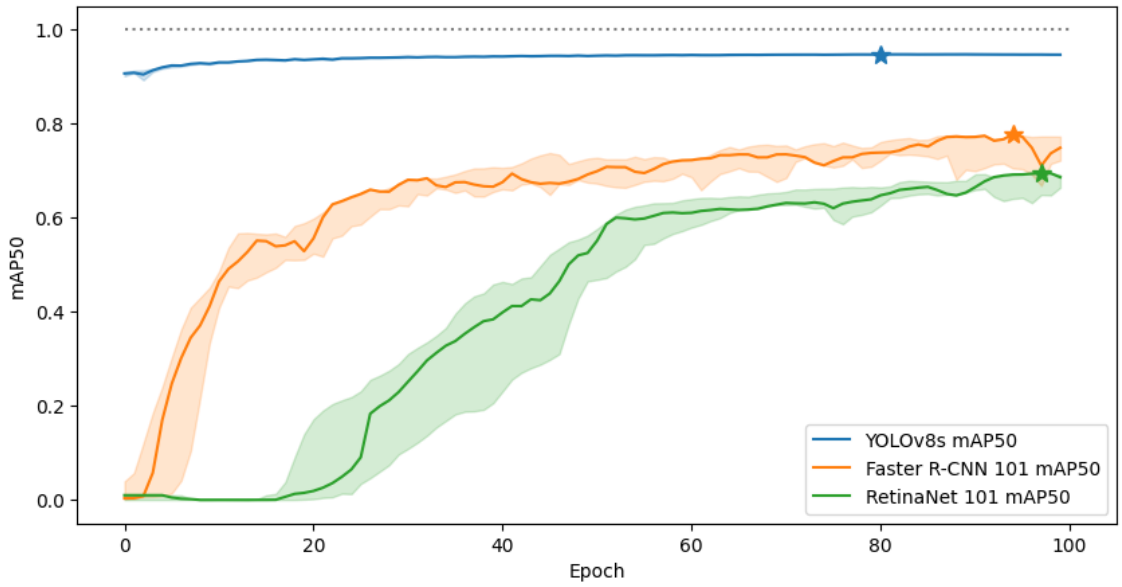
Toda esta codificación se muestra en el anexo 07.

4.2 Presentación y análisis de los resultados

4.2.1 Resultados para el indicador mAP50

Figura 13

Diagrama de líneas para el indicador mAP50

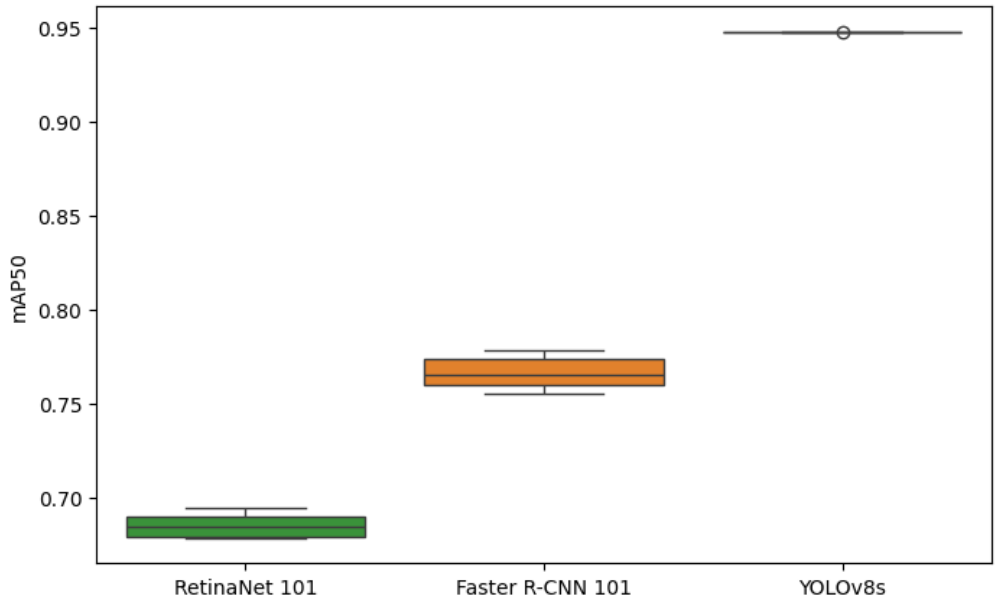


En la Figura 13, se presentan en un diagrama de líneas los resultados del indicador mAP50 de los diez experimentos realizados con cada arquitectura, destacando los experimentos con los mejores resultados.

Los puntos marcados con "estrellas" representan el valor máximo del indicador mAP50 obtenido por los modelos de cada arquitectura. YOLOv8s alcanzó un valor máximo de mAP50 de 0.94751 en la época 81, Faster R-CNN 101 logró un valor máximo de mAP50 de 0.778182 en la época 95, y RetinaNet 101 obtuvo un valor máximo de mAP50 de 0.694976 en la época 98.

Figura 14

Diagrama de cajas para el indicador mAP50

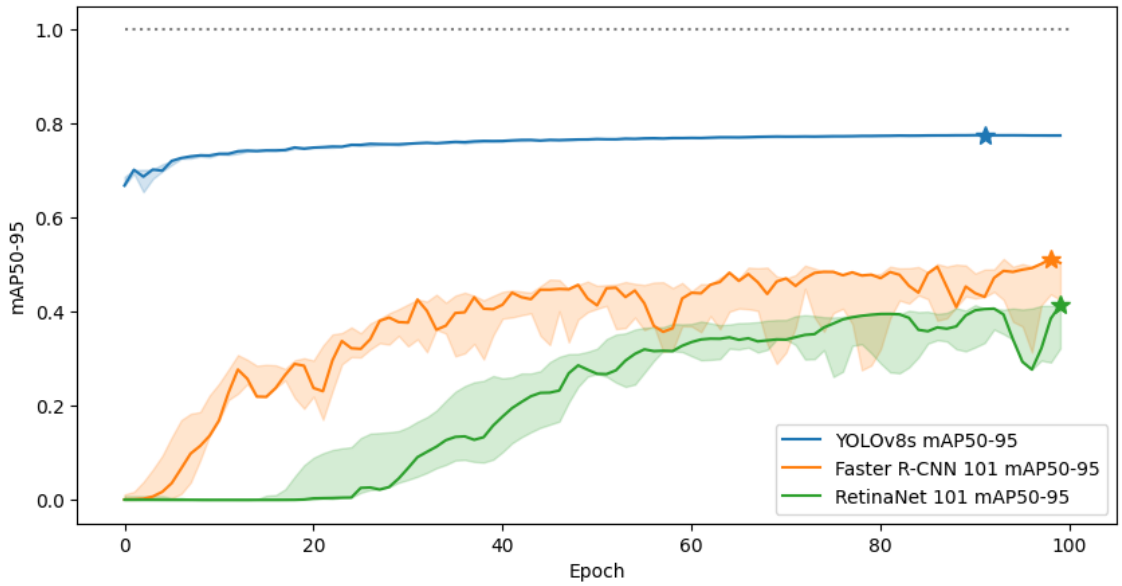


En la Figura 14 se presentan los valores de la métrica mAP50 de las diez ejecuciones de cada arquitectura en un gráfico de cajas. Podemos observar que el rango de valores de mAP50 de los modelos YOLOv8s es superior al de los modelos Faster R-CNN 101 y RetinaNet 101. Además, YOLOv8s muestra una dispersión mínima de datos en comparación con los otros dos modelos. Se destaca un valor atípico en YOLOv8s, correspondiente a un modelo con un valor de mAP50 de 0.947090.

4.2.2 Resultados para el indicador mAP50-95

Figura 15

Diagrama de líneas para el indicador mAP50-95

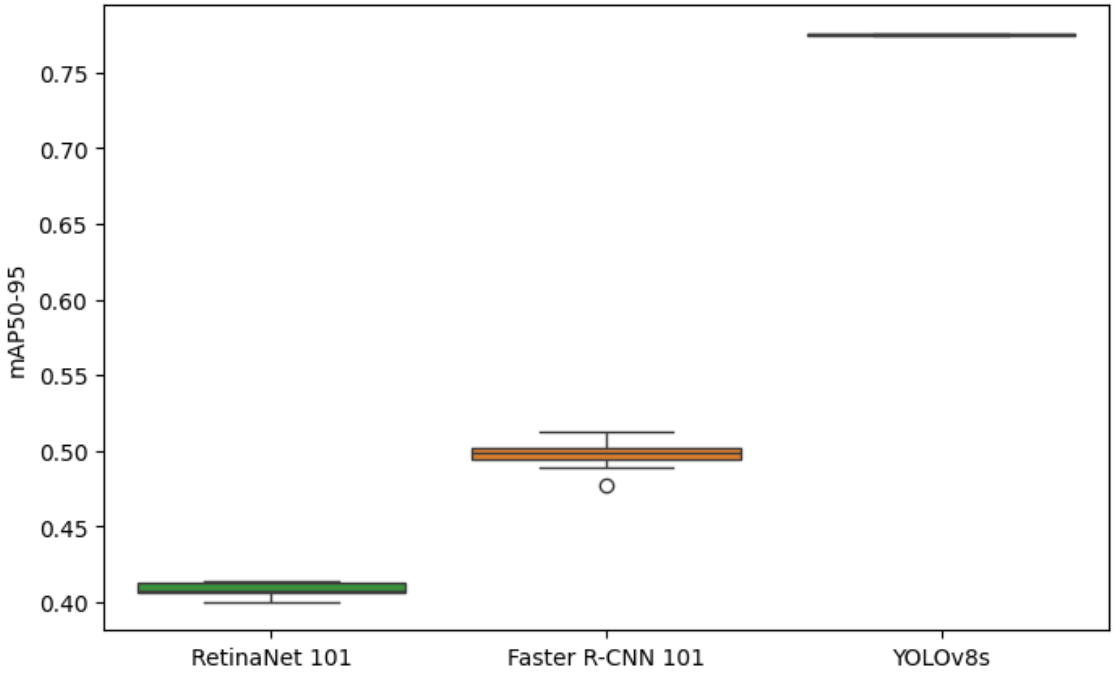


En la Figura 15, representada en un diagrama de líneas, se muestran los resultados del indicador mAP50-95 de los diez experimentos realizados con cada arquitectura, resaltando los experimentos con mejores resultados.

Los puntos marcados con "estrellas" representan el mayor valor del indicador mAP50-95 obtenido por los modelos de cada arquitectura. YOLOv8s obtuvo un valor máximo de mAP50-95 de 0.77562 en la época 92, Faster R-CNN 101 un valor máximo de mAP50-95 de 0.512923 en la época 99 y RetinaNet 101 un valor máximo de mAP50-95 de 0.414203 en la época 100.

Figura 16

Diagrama de cajas para el indicador mAP50-95



En la Figura 16, se representan los valores de la métrica mAP50-95 de las diez ejecuciones de cada arquitectura en un gráfico de cajas. Podemos observar que el rango de valores de mAP50-95 de los modelos YOLOv8s es superior a los de Faster R-CNN 101 y RetinaNet 101. Además, YOLOv8s presenta una dispersión mínima de datos en comparación con los otros dos modelos. Se marca un valor atípico en Faster R-CNN 101 correspondiente a un modelo con un valor de mAP50-95 de 0.477223.

Tabla 4

Datos obtenidos del entrenamiento de modelos

Datos	Indicador	Evaluación	Modelos		
			YOLOv8s	Faster R-CNN 101	RetinaNet 101
Conjunto de validación	mAP50	Min	**0.94709**	0.755218	0.678889
		Max	**0.94751**	0.778182	0.694976
		Media	**0.947349**	0.7664784	0.6853619
		Mediana	**0.947385**	0.7652955	0.68478
		iqr	**0.00011**	0.01434125	0.01075675
		Desviación estándar	**0.00011976**	0.00850974	0.00610431
	mAP50-95	Min	**0.77443**	0.477223	0.400307
		Max	**0.77562**	0.512923	0.414203
		Media	**0.775076**	0.4983881	0.4087192
		Mediana	**0.77508**	0.498716	0.4079515
		iqr	**0.0005275**	0.00763675	0.00652975
		Desviación estándar	**0.00037253**	0.01050299	0.00469501

Un resumen de los datos puede verse en la Tabla 4 en donde se muestra una evaluación del valor mínimo y el valor máximo por cada indicador obtenido por los modelos; también se muestra la media, mediana, el rango intercuartil (iqr) y la desviación estándar por las diez ejecuciones de los modelos en sus respectivas métricas. Los mejores resultados de cada evaluación se encuentran remarcados con negrita.

Finalmente, en la Tabla 5 se muestran los tiempos promedios de las diez ejecuciones de entrenamiento de cada modelo. Los modelos con la arquitectura YOLOv8s fueron los que más tardaron en entrenarse, luego fueron los de Faster R-CNN 101 y finalmente los de RetinaNet101 fueron los que más rápido llegaron a entrenar.

Tabla 5

Tiempo promedio de entrenamiento de modelos

Modelos	Tiempo promedio de entrenamiento
YOLOv8s	7 horas, 44 minutos y 30 segundos
Faster R-CNN 101	4 horas, 49 minutos y 34 segundos.
RetinaNet 101	3 horas, 5 minutos y 35 segundos.

4.3 Contrastación de hipótesis

4.3.1 Prueba de normalidad

Según González et al. (2013), para realizar la prueba de normalidad con datos menores a treinta (n<30) se recomienda el uso de la prueba Shapiro-Wilk.

Para esta investigación se decidió aplicar la prueba de normalidad Shapiro-Wilk ya que el número de datos por cada modelo no es mayor a treinta. Se aplicó la prueba en SPSS a los resultados de los dos indicadores y se usó un nivel de confianza de 95% y una significancia α de 5%.

4.3.1.1 Prueba de normalidad para el indicador mAP50 de modelos

H0: Los datos del indicador mAP50 de los modelos provienen de una distribución normal.

H1: Los datos del indicador mAP50 de los modelos no provienen de una distribución normal.

Tabla 6

Prueba de normalidad para el indicador mAP50 de modelos

Pruebas de normalidad				
Indicador	Modelo	Shapiro-Wilk		
		Estadístico	gl	Sig.
mAP50	Faster R-CNN 101	,922	10	,375
	RetinaNet 101	,891	10	,173
	YOLOv8s	,918	10	,343

- Si p-valor valor ≥ α, entonces se acepta H0.
- Si p-valor valor < α, entonces se acepta H1.

Observando la Tabla 6, vemos que el p-valor de los modelos para el indicador de mAP50 es mayor a la significancia de 5%. Entonces se acepta la hipótesis nula y se afirma que los datos provienen de una distribución normal.

4.3.1.2 Prueba de normalidad para el indicador mAP50-95 de modelos

H0: Los datos del indicador mAP50-95 de los modelos provienen de una distribución normal.

H1: Los datos del indicador mAP50-95 de los modelos no provienen de una distribución normal.

Tabla 7

Prueba de normalidad para el indicador mAP50-95 de modelos

Pruebas de normalidad				
Indicador	Modelo	Shapiro-Wilk		
		Estadístico	gl	Sig.
mAP50-95	Faster R-CNN 101	,943	10	,592
	RetinaNet 101	,923	10	,383
	YOLOv8s	,922	10	,370

- Si p-valor ≥ α, entonces se acepta H0.
- Si p-valor < α, entonces se acepta H1.

Observando la Tabla 7, vemos que el p-valor de los modelos para el indicador de mAP50-95 es mayor a la significancia de 5%. Entonces se acepta la hipótesis nula y se afirma que los datos provienen de una distribución normal.

Los resultados de estas pruebas se encuentran en el anexo 3 y nos muestran que los datos corresponden a una distribución normal, por lo que se aplica el estadístico ANOVA para contrastación de hipótesis.

4.3.2 Prueba de hipótesis (ANOVA)

Según González et al. (2013), el análisis de varianza (ANOVA) es la prueba paramétrica más comúnmente empleada para determinar si las medias de dos o más muestras proceden de poblaciones idénticas.

4.3.2.1 Prueba de hipótesis para el indicador mAP50 de modelos

H0: No existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

H1: Existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

Tabla 8

Prueba ANOVA para el indicador mAP50 de modelos

ANOVA						
		Suma de cuadrados	gl	Media cuadrática	F	Sig.
mAP50	Entre grupos	,360	2	,180	4919,720	,000
	Dentro de grupos	,001	27	,000		
	Total	,361	29			

Según los resultados mostrados en la Tabla 8, se obtuvo un p-valor unilateral de 0,000 < 0,05, entonces se rechaza la hipótesis nula y se afirma que existe una diferencia significativa en el rendimiento de modelos según el indicador mAP50.

4.3.2.2 Prueba de hipótesis para el indicador mAP50-95 de modelos

H0: No existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50-95 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

H1: Existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50-95 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

Tabla 9

Prueba ANOVA para el indicador mAP50-95 de modelos

ANOVA						
		Suma de cuadrados	gl	Media cuadrática	F	Sig.
mAP50-95	Entre grupos	,729	2	,365	8257,470	,000
	Dentro de grupos	,001	27	,000		
	Total	,731	29			

Según los resultados vistos en la Tabla 9, se obtuvo un p-valor unilateral de 0,000 < 0,05, entonces se rechaza la hipótesis nula y se afirma que existe una diferencia significativa en el rendimiento de modelos según el indicador mAP50-95.

4.3.3 Prueba de post hoc (Tukey HSD)

González et al. (2013) mencionan que la prueba de Tukey HSD es una técnica estadística utilizada como un procedimiento post hoc en el análisis de varianza (ANOVA). Se aplica cuando hay más de dos niveles en el factor de interés, permitiendo identificar las diferencias específicas entre cada par de niveles.

Según los resultados obtenidos en la Tabla 10, observamos un p-valor unilateral de 0,000 < 0,05 en cada comparación, entonces se puede afirmar que hay una diferencia significativa entre cada uno de los grupos comparados.

Tabla 10

Prueba post hoc Tukey HSD a modelos

Comparaciones múltiples

Variable dependiente	(I) Modelo	(J) Modelo	Diferencia de medias (I-J)	Desv. Error	Sig.	Intervalo de confianza al 95%	
						Límite inferior	Límite superior
mAP50	Faster R-CNN 101	RetinaNet 101	,081116500000000*	,002704225359473	,000	,074411595806840	,087821404193160
		YOLO v8s	-,180870600000000*	,002704225359473	,000	-,187575504193159	-,174165695806840
	RetinaNet 101	Faster R-CNN 101	-,081116500000000*	,002704225359473	,000	-,087821404193160	-,074411595806840
		YOLO v8s	-,261987100000000*	,002704225359473	,000	-,268692004193160	-,255282195806840
	YOLO v8s	Faster R-CNN 101	,180870600000000*	,002704225359473	,000	,174165695806840	,187575504193159
		RetinaNet 101	,261987100000000*	,002704225359473	,000	,255282195806840	,268692004193160
mAP50-95	Faster R-CNN 101	RetinaNet 101	,089668900000000*	,002972031718713	,000	,082299991982330	,097037808017670
		YOLO v8s	-,276687900000000*	,002972031718713	,000	-,284056808017670	-,269318991982330
	RetinaNet 101	Faster R-CNN 101	-,089668900000000*	,002972031718713	,000	-,097037808017670	-,082299991982330
		YOLO v8s	-,366356800000000*	,002972031718713	,000	-,373725708017670	-,358987891982330
	YOLO v8s	Faster R-CNN 101	,276687900000000*	,002972031718713	,000	,269318991982330	,284056808017670
		RetinaNet 101	,366356800000000*	,002972031718713	,000	,358987891982330	,373725708017670

CAPÍTULO V
DISCUSIÓN

5.1 Pruebas de validación

Para validar los indicadores propuestos en este trabajo, se tomarán como referencia los estudios de diversos autores que han abordado comparación de modelos de detección de objetos, utilizando métricas aceptadas y ampliamente utilizadas en la comunidad científica.

El indicador mAP50 para la comparación de modelos de detección de objetos es usado en los trabajos de Aljaafreh et al. (2023) y Zhu et al. (2024).

El indicador mAP50-95 y mAP50 para la comparación de modelos de detección de objetos son usados ambos en los trabajos de Butt et al. (2023) y Angelika et al. (2024).

5.2 Aplicación de tecnología encontrada

Esta investigación no es aplicada, entonces solo se describieron y compararon los resultados de los indicadores obtenidos por los modelos de detección de objetos aplicados a los frutos de olivo.

5.3 Contraste con trabajos de investigación similares

En el presente trabajo se desarrolló el entrenamiento de modelos de inteligencia artificial para la detección de frutos de olivo, se obtuvieron los valores de los indicadores mAP50 y mAP50-95 para las tres arquitecturas seleccionadas YOLOv8s, Faster R-CNN 101 y RetinaNet 101. YOLOv8s tuvo un valor máximo de 0.94751 en el indicador mAP50 y 0.77562 en el indicador mAP50-95, Faster R-CNN 101 tuvo un valor máximo de 0.778182 en el indicador mAP50 y 0.512923 en el indicador mAP50-95, RetinaNet 101 obtuvo un valor máximo de 0.694976 en el indicador mAP50 y 0.414203 en el indicador mAP50-95.

Al contrastar nuestros resultados con otros trabajos que también comparan modelos de inteligencia artificial para detección de objetos con diferentes arquitecturas, se puede observar una misma tendencia en los resultados. Primero mencionaremos los

trabajos que usaron las mismas arquitecturas, el trabajo de Angelika et al. (2024), donde los modelos de inteligencia artificial para detección de objetos tienen la tarea de localización de placas de autos, YOLOv8 con AP50 de 0.93 y AP50-95 de 0.708 obtuvo un mejor resultado a comparación de Faster R-CNN con AP50 de 0.71 y AP50-95 de 0.5 en el conjunto de validación. También se observó una similitud de resultados con el estudio de Butt et al. (2023) en el que la tarea estuvo enfocada a la detección de agujeros de bala en dianas, donde YOLOv8s obtuvo el mejor desempeño con un valor de mAP50 de 0.965 y mAP50-95 de 0.646, en comparación a Faster R-CNN 101 con mAP50 de 0.83554 y mAP50-95 de 0.5009, y RetinaNet 101 con mAP50 de 0.66416 y mAP50-95 de 0.41488.

Finalmente, veremos los resultados de otros trabajos en donde se entrenaron modelos de inteligencia artificial enfocados a la detección de aceitunas. En el artículo de Aljaafreh et al. (2023) se obtuvo un modelo YOLOv5s con valor de 0.7708 en mAP50-95 y en la investigación de Zhu et al. (2024), donde las clases a detectar son cuatro fases de maduración de la aceituna, se obtuvieron resultados de mAP de 0.9460, 0.9545, 0.9375, y 0.9605 para las variedades Frantoio, Ezhi 8, Leccino y Picholine con una arquitectura propuesta llamada Olive-EfficientDet. Se reportaron resultados similares en la detección de frutos de olivo, lo que sugiere que esta tarea es factible con diferentes enfoques y conjuntos de datos.

Si bien los resultados de este trabajo y de las otras investigaciones son alentadores, la efectividad de los modelos puede variar según el conjunto de datos utilizado y otras condiciones de captura de imágenes. Por lo tanto, se necesita más investigación para validar la generalización de nuestros modelos en diferentes contextos y condiciones.

CONCLUSIONES

Primero

De los resultados obtenidos de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna, el modelo con mejor desempeño en cuanto al indicador mAP50 fue YOLOv8s, con un valor máximo de 0.94751 (94.751%) y un valor promedio en sus diez ejecuciones de 0.947349 (94.7349%). Le sigue Faster-RCNN 101, con un valor máximo de 0.778182 (77.8182%) y un valor promedio en sus diez ejecuciones de 0.7664784 (76.64784%). Finalmente, RetinaNet 101 obtuvo un valor máximo de 0.694976 (69.4976%) y un valor promedio en sus diez ejecuciones de 0.6853619 (68.53619%).

Segundo

Se pudo comprobar que, de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna, el modelo con mejor desempeño en cuanto al indicador mAP50-95 fue YOLOv8s, con un valor máximo de 0.77562 (77.562%) y un valor promedio en sus diez ejecuciones de 0.775076 (77.5076%). Le sigue Faster-RCNN 101, con un valor máximo de 0.512923 (51.2923%) y un valor promedio en sus diez ejecuciones de 0.4983881 (49.83881%). Finalmente, RetinaNet 101 obtuvo un valor máximo de 0.414203 (41.4203%) y un valor promedio en sus diez ejecuciones de 0.4087192 (40.87192%).

Tercero

Se concluye que hay una diferencia significativa en los resultados para los indicadores de mAP50 y mAP50-95. El valor p en la prueba ANOVA fue 0.000, siendo menor que alfa (0.05), por lo que se acepta la hipótesis alternativa. Esto indica que existe una diferencia significativa en el rendimiento de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna.

RECOMENDACIONES

Primero

Para futuras investigaciones se sugiere explorar y comparar el rendimiento de diferentes arquitecturas de modelos de detección de objetos.

Segundo

Es recomendable realizar experimentos con la modificación de parámetros clave en los modelos y la configuración del entrenamiento para evaluar su impacto en el rendimiento de estos.

Tercero

Se debe tener en cuenta la importancia del tamaño y la diversidad del conjunto de datos en el entrenamiento de modelos de detección de objetos. Se recomienda explorar técnicas para aumentar la cantidad y la variabilidad de los datos, así como la recopilación de conjuntos de datos más amplios y representativos.

Cuarto

Dado que algunos modelos pueden requerir un poder computacional considerable durante el entrenamiento y la inferencia, es importante evaluar los recursos disponibles y seleccionar modelos que se ajusten a las limitaciones computacionales.

Quinto

La elección del tamaño de las imágenes de entrada puede afectar significativamente el rendimiento y la eficiencia de los modelos. Se sugiere evaluar el impacto del tamaño de las imágenes en el rendimiento del modelo y seleccionar el tamaño más adecuado para el contexto específico de la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

Agraria.pe. (29 de febrero de 2024). *Tacna: piden declarar en emergencia la producción de olivo.* Agraria.pe. https://agraria.pe/noticias/tacna-piden-declarar-en-emergencia-a-la-produccion-de-olivo--34936

Aljaafreh, A., Elzagzoug, E. Y., Abukhait, J., Soliman, A. H., Alja'Afreh, S. S., Sivanathan, A. y Hughes, J. (2023). A Real-Time Olive Fruit Detection for Harvesting Robot Based on YOLO Algorithms. *Acta Technologica Agriculturae*, *26*(3), 121-132. https://doi.org/10.2478/ata-2023-0017

Angelika Mulia, D., Safitri, S. y Putra Kusuma Negara, G. (2024). YOLOv8 and Faster R-CNN Performance Evaluation with Super-resolution in License Plate Recognition. *International Journal of Computing and Digital Systems*, *16*(1), 1-10. http://dx.doi.org/10.12785/ijcds/160129

Aquino, A., Ponce, J. M., Millán, B., Tejada Guzmán, D. y Andújar, J. M. (Septiembre, 2019). *Identificación y conteo de aceitunas en imágenes digitales tomadas en el olivar mediante morfología matemática y redes neuronales convolucionales* [Presentación de artículo]. XL Jornadas de Automática, Ferrol, España. https://doi.org/10.17979/spudc.9788497497169.818

Arias, F. G. (2012). *El Proyecto de Investigación. Introducción a la Metodología Científica.* (6ta. ed.). Editorial Episteme.

Bell, S., Zitnick, C. L., Bala, K. y Girshick, R. (Junio, 2016). *Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks* [Presentación de artículo]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA. https://doi.org/10.1109/CVPR.2016.314

Bellocchio, E., Ciarfuglia, T. A., Costante, G. y Valigi, P. (2019). Weakly supervised fruit counting for yield estimation using spatial consistency. *IEEE Robotics and Automation Letters*, *4*(3), 2348-2355. https://doi.org/10.1109/LRA.2019.2903260

Brownlee, J. (2016). *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. Machine Learning Mastery.

Butt, M., Glas, N., Monsuur, J., Stoop, R., y de Keijzer, A. (2023). Application of YOLOv8 and Detectron2 for Bullet Hole Detection and Score Calculation from Shooting Cards. *AI*, *5*(1), 72-90. https://doi.org/10.3390/ai5010005

Callier, P., y Sandel, O. (2021). De l'intelligence artificielle à son application en médecine. *Actualités Pharmaceutiques*, *60*(611), 18-20. https://doi.org/10.1016/j.actpha.2021.10.005

Cano, P., Satorres, S., Gómez, J. y Gámez, J. (2021). Automatic system for the detection of defects on olive fruits in an oil mill. *Applied Sciences*, *11*(17), 8167. https://doi.org/10.3390/app11178167

Carrasco S. (2006). *Metodología de la Investigación Científica.* Editorial San Marcos.

Chemelil, P. K. (2021). *Single Shot Multi Box Detector Approach to Autonomous Vision-Based Pick and Place Robotic Arm in the Presence of Uncertainties* [Tesis de maestría, Jomo Kenyatta University of Agriculture and Technology]. JKUAT Institutional Repository.

Chollet, F. (2017). *Deep learning with python*. Manning Publications.

COCOdataset. (s.f.). *Detection Evaluation.* COCOdataset. https://cocodataset.org/#detection-eval

Du, L., Zhang, R. y Wang, X. (2020). Overview of two-stage object detection algorithms. *Journal of Physics: Conference Series, 1544*(1), 012033. https://doi.org/10.1088/1742-6596/1544/1/012033

Ekman, M. (2021). *Learning deep learning: Theory and practice of neural networks, computer vision, natural language processing, and transformers using TensorFlow*. Addison-Wesley Professional.

Exportemos.pe. (s.f.). *PERFIL DEL MERCADO Y COMPETIVIDAD EXPORTADORA DE ACEITUNA.* Exportemos.pe. https://boletines.exportemos.pe/recursos/boletin/24959.pdf

Firdaus, F. F., Nugroho, H. A. y Soesanti, I. (Abril, 2021). *Deep neural network with hyperparameter tuning for detection of heart disease* [Presentación de artículo]. 2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), Bandung, Indonesia. https://doi.org/10.1109/APWiMob51111.2021.9435250

Ghosal, P., Nandanwar, L., Kanchan, S., Bhadra, A., Chakraborty, J. y Nandi, D. (Febrero, 2019). Brain tumor classification using ResNet-101 based squeeze and excitation deep neural network [Presentación de artículo]. 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India. https://doi.org/10.1109/ICACCP.2019.8882973

Ghoury, S., Sungur, C. y Durdu, A. (Abril, 2019). *Real-time diseases detection of grape and grape leaves using faster r-cnn and ssd mobilenet architectures* [Presentación de artículo]. International conference on advanced technologies, computer engineering and science *(ICATCES 2019),* Alanya, Turquía. https://www.researchgate.net/publication/334987612_Real-Time_Diseases_Detection_of_Grape_and_Grape_Leaves_using_Faster_R-CNN_and_SSD_MobileNet_Architectures

Gómez, M. y Vidal, J. (2006). *Variedades del Olivar*. Ministerio de Agricultura, Pesca y Alimentacion. https://www.mapa.gob.es/ministerio/pags/biblioteca/hojas/hd_2006_2117.pdf

González, C. G., Lise, A. V. y Felpeto, A. B. (2013). *Tratamiento de datos con R, Statistica y SPSS*. Ediciones Díaz de Santos.

Goodfellow, I., Bengio, Y. y Courville, A. (2016). *Deep learning*. MIT press. https://www.deeplearningbook.org/

Google. (s.f.). Recuperado el 29 de abril de 2024 de https://maps.app.goo.gl/Y943nuFTw3cvodCc7

Hernández Sampieri, R., Fernández Collado, C., y Pilar Baptista Lucio, M. (2014). *Metodología de la investigación* (6ta. ed.). McGraw-Hill.

Horna, A. A. V. (2012). *Desde la idea hasta la sustentación: 7 pasos para una tesis exitosa.* Instituto de Investigación de la Facultad de Ciencias Administrativas y Recursos Humanos. Universidad de San Martín de Porres.

Hoyo Dorado, E. D. (2021). *Diseño, implementación y evaluación de una red neuronal convolucional para la detección de puntos principales en naranjas* [Tesis de maestría, Universitat Politècnica de València] Repositorio institucional UPV.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer I., Wojna Z., Song Y., Guadarrama S. y Murphy, K. (2017). *Speed/accuracy trade-offs for modern convolutional object detectors.* arXiv. https://arxiv.org/abs/1611.10012

IBM. (s.f.a). *¿Qué son las redes neuronales convolucionales?.* IBM. https://www.ibm.com/es-es/topics/convolutional-neural-networks

IBM. (s.f.b). *What is an AI model?* IBM. https://www.ibm.com/topics/ai-model#:~:text=An%20AI%20model%20is%20a,they've%20been%20programmed%20for.

Janiesch, C., Zschech, P. y Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685–695. https://doi.org/10.1007/s12525-021-00475-2

Jayawardena, G. y Jayarathna, S. (Agosto, 2020). *Automated filtering of eye gaze metrics from dynamic areas of interest* [Presentación de artículo]. 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science *(IRI)*, Las Vegas, USA. https://doi.org/10.1109/IRI49571.2020.00018

Jocher, G., Chaurasia, A. y Qiu, J. (2023). *Ultralytics YOLO (Version 8.0.0)*. Github. Recuperado el 28 de abril de 2024 de https://github.com/ultralytics/ultralytics

Li, G., Huang, X., Ai, J., Yi, Z. y Xie, W. (2021). Lemon-YOLO: An efficient object detection method for lemons in the natural environment. *IET Image Processing*, *15*(9), 1998-2009. https://doi.org/10.1049/ipr2.12171

Lin, T. Y., Goyal, P., Girshick, R., He, K. y Dollár, P. (2017). *Focal Loss for Dense Object Detection.* arXiv. https://arxiv.org/abs/1708.02002

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P y Zitnick, C. L. (Septiembre, 2014). *Microsoft COCO: Common objects in context* [Presentación de artículo]. Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland. https://doi.org/10.1007/978-3-319-10602-1_48

Martineau, K. (5 de octubre de 2023). *What is AI inferencing?*. IBM. https://research.ibm.com/blog/AI-inference-explained

Ministerio de Comercio Exterior y Turismo. (16 de enero de 2024). *Reporte de Comercio - Reporte Comercio Regional - RCR - Tacna 2023 - I Semestre.* gob.pe https://www.gob.pe/institucion/mincetur/informes-publicaciones/5051942-reporte-de-comercio-reporte-comercio-regional-rcr-tacna-2023-i-semestre

Ng, A. (2017). *Machine Learning Yearning*. Deeplearning.ai

Olney, A. M. y Fleming, S. D. (2021). JupyterLab extensions for blocks programming, self-explanations, and HTML injection. *Joint Proceedings of the Workshops at the 14th International Conference on Educational Data Mining vol. 3051*. https://par.nsf.gov/servlets/purl/10311836

Padilla, R., Netto, S. L. y Da Silva, E. A. (Julio, 2020). *A survey on performance metrics for object-detection algorithms* [Presentación de artículo]. 2020 international conference on systems, signals and image processing (IWSSIP), Niteroi, Brazil. https://doi.org/10.1109/IWSSIP48289.2020.9145130

Peña Moliner, D. (2020). *Extending object classification convolutional neural networks to custom logo detection* [Tesis de maestría, Universitat Politècnica de Catalunya]. UPCommons.

Reis, D., Kupec, J., Hong, J. y Daoudi, A. (2024). *Real-time flying object detection with YOLOv8.* arXiv. https://arxiv.org/abs/2305.09972

Ren, S., He, K., Girshick, R. y Sun, J. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* arXiv. https://arxiv.org/abs/1506.01497

Sportelli, M., Apolo-Apolo, O. E., Fontanelli, M., Frasconi, C., Raffaelli, M., Peruzzi, A. y Perez-Ruiz, M. (2023). Evaluation of YOLO object detectors for weed detection in different turfgrass scenarios. *Applied Sciences*, *13*(14), 8502. https://doi.org/10.3390/app13148502

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. y Liu, C. (Octubre, 2018). *A survey on deep transfer learning* [Presentación de artículo]. Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece. https://doi.org/10.1007/978-3-030-01424-7_27

Tkachenko, M., Malyuk, M., Holmanyuk, A. y Liubimov, N. (2020). *Label Studio: Data labeling software*. Github. Recuperado el 28 de abril de 2024 de https://github.com/heartexlabs/label-studio

W., J. (19 de mayo de 2020). *RetinaNet-101 Feature Pyramid Net Trained on MS-COCO Data.* Wolfram. https://resources.wolframcloud.com/NeuralNetRepository/resources/RetinaNet-101-Feature-Pyramid-Net-Trained-on-MS-COCO-Data/

Wang, B. (2022). *A Parallel Implementation of Computing Mean Average Precision.* arXiv. https://arxiv.org/abs/2206.09504

Wang, Y., Hao, Z., Zuo, F. y Pan, S. (2021). A fabric defect detection system based improved yolov5 detector. *Journal of Physics: Conference Series*, 2010(1), 012191. https://doi.org/10.1088/1742-6596/2010/1/012191

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y. y Girshick, R. (2019). *Detectron2.* Github. Recuperado el 28 de abril de 2024 de https://github.com/facebookresearch/detectron2

Zhang, A., Lipton, Z. C., Li, M. y Smola, A. J. (2021). *Dive into deep learning*. arXiv. https://arxiv.org/abs/2106.11342

Zhang, T., Wang, D. y Lu, Y. (2024). A data-centric strategy to improve performance of automatic pavement defects detection. *Automation in Construction*, *160*, 105334. https://doi.org/10.1016/j.autcon.2024.105334

Zhang, Y., Zhang, W., Yu, J., He, L., Chen, J. y He, Y. (2022). Complete and accurate holly fruits counting using YOLOX object detection. *Computers and Electronics in Agriculture*, *198*, 107062. https://doi.org/10.1016/j.compag.2022.107062

Zhao, Z. Q., Zheng, P., Xu, S. T. y Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), 3212-3232. https://doi.org/10.1109/TNNLS.2018.2876865

Zhu, X., Chen, F., Zhang, X., Zheng, Y., Peng, X., & Chen, C. (2024). Detection the maturity of multi-cultivar olive fruit in orchard environments based on Olive-EfficientDet. *Scientia Horticulturae*, *324*, 112607. https://doi.org/10.1016/j.scienta.2023.112607

ANEXOS

ANEXO 01: MATRIZ DE CONSISTENCIA

Título: Análisis comparativo de modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna

<table>
<tr><th>PROBLEMA</th><th>OBJETIVOS</th><th>HIPÓTESIS</th><th>VARIABLE</th></tr>
<tr><td>Problema general</td><td>Objetivo general</td><td>Hipótesis general</td><td>Variables e indicadores</td></tr>
<tr><td>¿Cómo es el rendimiento de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna?</td><td>Comparar el rendimiento de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna</td><td>Existe una diferencia significativa en el rendimiento de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna</td><td rowspan="3">Variable:
Rendimiento de modelos de inteligencia artificial basados en aprendizaje profundo
Indicadores:
I1: mAP50
I2: mAP50-95</td></tr>
<tr><td>Problemas específicos</td><td>Objetivos específicos</td><td>Hipótesis específicas</td></tr>
<tr><td>¿Cómo es el rendimiento en cuanto al indicador mAP50 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna?
¿Cómo es el rendimiento en cuanto al indicador mAP50-95 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna?</td><td>Comparar el rendimiento en cuanto al indicador mAP50 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna
Comparar el rendimiento en cuanto al indicador mAP50-95 de diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos - Tacna</td><td>Existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna
Existe una diferencia significativa en el rendimiento en cuanto al indicador mAP50-95 de los diferentes modelos de inteligencia artificial basados en aprendizaje profundo aplicados a la detección de frutos de olivo en La Yarada Los Palos – Tacna</td></tr>
<tr><td colspan="2">Población: 19,560 imágenes de recortes de árboles de olivo
Muestra: 10,728 imágenes de recortes de árboles de olivo</td><td colspan="2">Tipo de investigación: básica Nivel de investigación: descriptivo-comparativo
Diseño de investigación: no experimental</td></tr>
</table>

ANEXO 02: DATOS OBTENIDOS DE ENTRENAMIENTO DE MODELOS

#	Modelo	mAP50	mAP50-95
1	Faster R-CNN 101	0.776239	0.496157
2	Faster R-CNN 101	0.759356	0.498229
3	Faster R-CNN 101	0.772699	0.512923
4	Faster R-CNN 101	0.778182	0.512628
5	Faster R-CNN 101	0.766796	0.499203
6	Faster R-CNN 101	0.756353	0.489454
7	Faster R-CNN 101	0.77472	0.502424
8	Faster R-CNN 101	0.763795	0.501588
9	Faster R-CNN 101	0.755218	0.477223
10	Faster R-CNN 101	0.761426	0.494052
11	RetinaNet 101	0.686559	0.40712
12	RetinaNet 101	0.686255	0.412641
13	RetinaNet 101	0.680984	0.408052
14	RetinaNet 101	0.679032	0.407851
15	RetinaNet 101	0.678889	0.406055
16	RetinaNet 101	0.691582	0.414203
17	RetinaNet 101	0.679098	0.400307
18	RetinaNet 101	0.683305	0.40385
19	RetinaNet 101	0.692939	0.414192
20	RetinaNet 101	0.694976	0.412921
21	YOLOv8s	0.94742	0.77529
22	YOLOv8s	0.94751	0.77483
23	YOLOv8s	0.9473	0.77533
24	YOLOv8s	0.94709	0.77479
25	YOLOv8s	0.94741	0.77484
26	YOLOv8s	0.94741	0.77539
27	YOLOv8s	0.94743	0.77487
28	YOLOv8s	0.94736	0.77562
29	YOLOv8s	0.94723	0.77443
30	YOLOv8s	0.94733	0.77537

ANEXO 03: PRUEBA DE NORMALIDAD DE DATOS PARA INDICADORES MAP50 Y MAP50-95 DE MODELOS EN SPSS 25

Pruebas de normalidad

		Shapiro-Wilk		
	Modelo	Estadístico	gl	Sig.
mAP50	Faster R-CNN 101	,922	10	,375
	RetinaNet 101	,891	10	,173
	YOLOv8s	,918	10	,343
mAP50-95	Faster R-CNN 101	,943	10	,592
	RetinaNet 101	,923	10	,383
	YOLOv8s	,922	10	,370

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

ANEXO 04: PRUEBA ESTADÍSTICA ANOVA Y TUKEY PARA INDICADORES MAP50 Y MAP50-95 DE MODELOS EN SPSS 25

ANOVA

		Suma de cuadrados	gl	Media cuadrática	F	Sig.
mAP50	Entre grupos	,360	2	,180	4919,720	,000
	Dentro de grupos	,001	27	,000		
	Total	,361	29			
mAP50-95	Entre grupos	,729	2	,365	8257,470	,000
	Dentro de grupos	,001	27	,000		
	Total	,731	29			

Pruebas post hoc

Comparaciones múltiples

HSD Tukey

Variable dependiente	(I) Modelo	(J) Modelo	Diferencia de medias (I-J)	Desv. Error	Sig.	Intervalo de confianza al 95% Límite inferior	Límite superior
mAP50	Faster R-CNN 101	RetinaNet 101	,081116500*	,002704225	,000	,07441160	,08782140
		YOLOv8s	-,180870600*	,002704225	,000	-,18757550	-,17416570
	RetinaNet 101	Faster R-CNN 101	-,081116500*	,002704225	,000	-,08782140	-,07441160
		YOLOv8s	-,261987100*	,002704225	,000	-,26869200	-,25528220
	YOLOv8s	Faster R-CNN 101	,180870600*	,002704225	,000	,17416570	,18757550
		RetinaNet 101	,261987100*	,002704225	,000	,25528220	,26869200
mAP50-95	Faster R-CNN 101	RetinaNet 101	,089668900*	,002972032	,000	,08229999	,09703781
		YOLOv8s	-,276687900*	,002972032	,000	-,28405681	-,26931899
	RetinaNet 101	Faster R-CNN 101	-,089668900*	,002972032	,000	-,09703781	-,08229999
		YOLOv8s	-,366356800*	,002972032	,000	-,37372571	-,35898789
	YOLOv8s	Faster R-CNN 101	,276687900*	,002972032	,000	,26931899	,28405681
		RetinaNet 101	,366356800*	,002972032	,000	,35898789	,37372571

*. La diferencia de medias es significativa en el nivel 0.05.

ANEXO 05: FOTOS DE RECOLECCIÓN DE IMÁGENES EN CAMPO

Figura 17

Toma de fotografía a un árbol de olivo

Figura 18

Vista de la cámara tomando fotografía

Figura 19

Fotografía en el campo durante la recolección de imágenes de olivo

ANEXO 06: IMÁGENES DE EJEMPLO DE INFERENCIA DE MODELOS

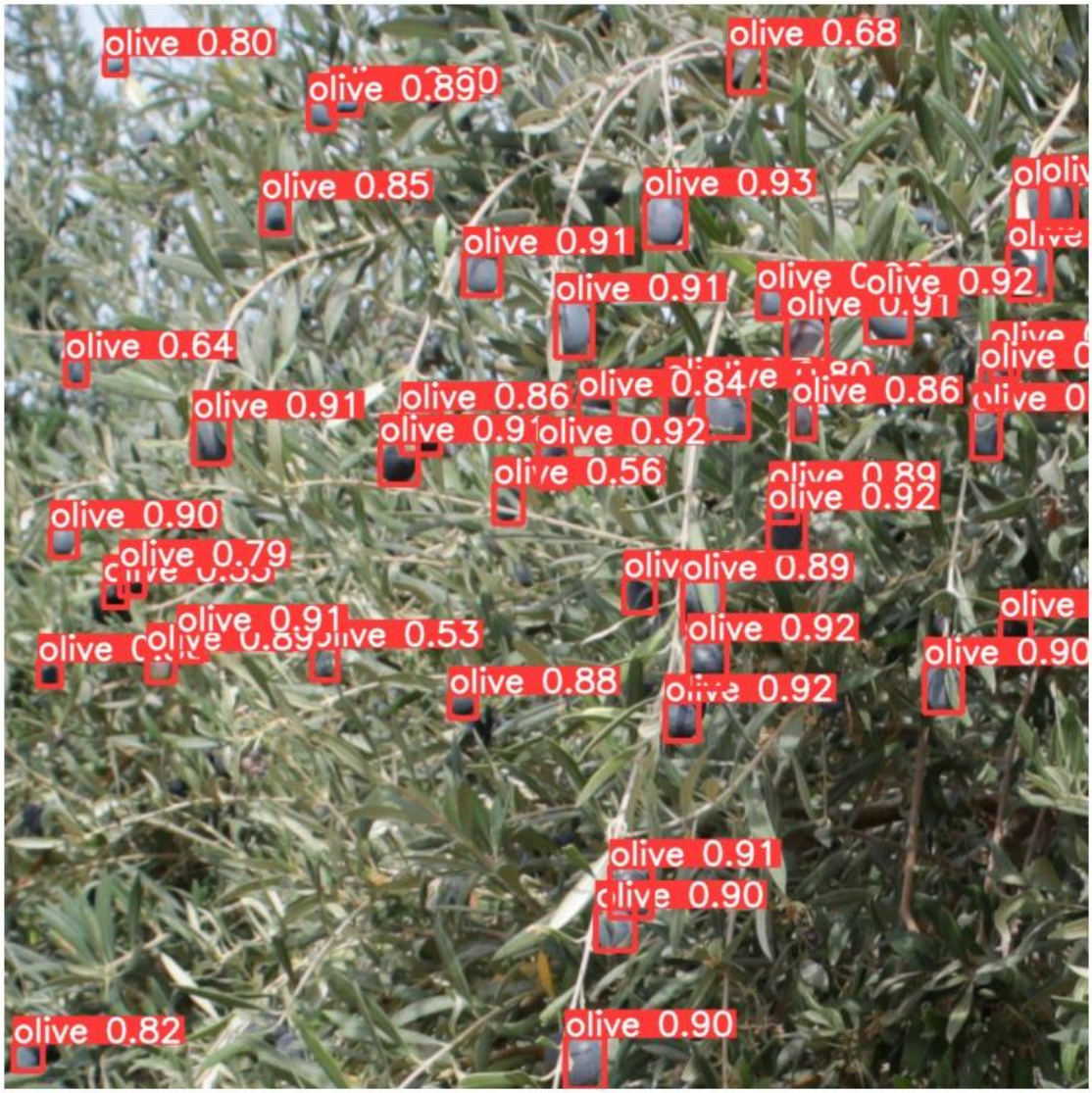
Figura 20

Imagen de ejemplo etiquetada manualmente

Nota: Esta imagen fue etiquetada manualmente con 51 cajas delimitadoras como referencia para ver las predicciones de los modelos.

Figura 21

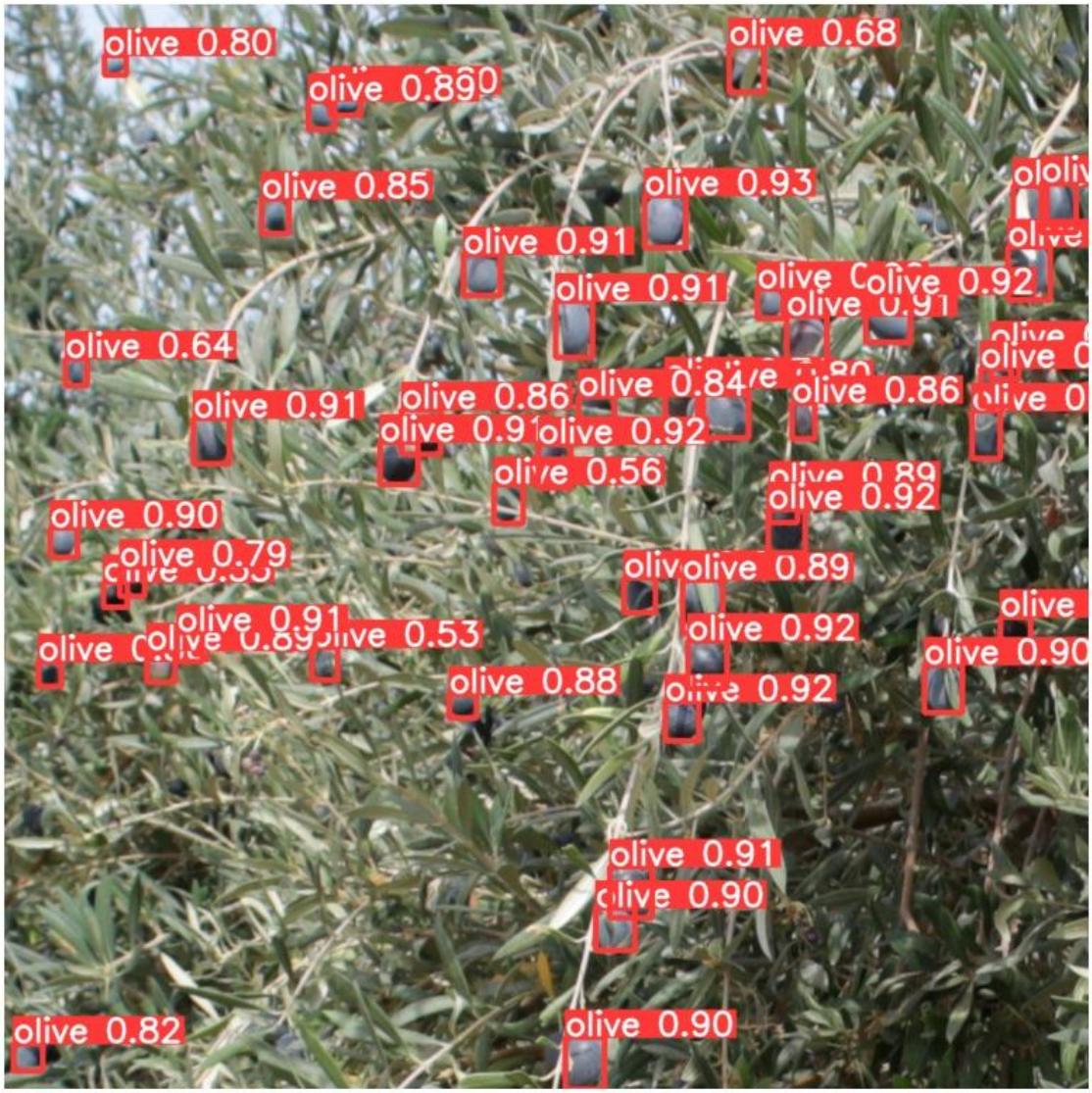
Inferencia del modelo YOLOv8s con mayor desempeño en el indicador mAP50



Número de detecciones : 47, Umbral de detección : 0.5

Figura 22

Inferencia del modelo YOLOv8s con mayor desempeño en el indicador mAP50-95



Número de detecciones : 46, Umbral de detección : 0.5

Figura 23

Inferencia del modelo Faster R-CNN 101 con mayor desempeño en el indicador mAP50

Figura 24

Inferencia del modelo Faster R-CNN 101 con mayor desempeño en el indicador mAP50-95

Figura 25

Inferencia del modelo RetinaNet 101 con mayor desempeño en el indicador mAP50

Figura 26

Inferencia del modelo RetinaNet 101 con mayor desempeño en el indicador mAP50-95

Número de detecciones : 26, Umbral de detección : 0.5

Figura 27

Inferencia de árbol completo con el modelo YOLOv8s de mayor desempeño en indicador mAP50

ANEXO 07: CODIFICACIÓN

Código para conversión de formato .CR2 a .PNG.

```
1. from PIL import Image
2.
3. # Obtén la lista de subcarpetas en la carpeta de origen
4. subcarpetas = os.listdir(carpeta_origen)
5.
6. # Itera sobre las subcarpetas
7. for subcarpeta in subcarpetas:
8.     ruta_subcarpeta = os.path.join(carpeta_origen,
   subcarpeta)
9.
10.        # Verifica si la ruta es una carpeta (ignorando
   archivos)
11.        if os.path.isdir(ruta_subcarpeta):
12.            # Obtén la lista de archivos (imágenes) dentro de
   la subcarpeta
13.            archivos = os.listdir(ruta_subcarpeta)
14.
15.            # Itera sobre los archivos (imágenes) dentro de
   la subcarpeta
16.            for archivo in archivos:
17.                ruta_archivo = os.path.join(ruta_subcarpeta,
   archivo)
18.                ruta_archivo_modificada =
   ruta_archivo.replace("fotografias olivo",
   "fotografias_olivo_png")
19.                ruta_archivo_modificada =
   ruta_archivo_modificada.replace(".CR2", ".png")
20.                #.CR2 to .png
21.                im = Image.open(ruta_archivo)
22.                rgb_im = im.convert('RGB')
23.                rgb_im.save(ruta_archivo_modificada)
24.                print(ruta_archivo_modificada," --Creado--")
```

Código para generar recortes.

```
1. import os
2. from PIL import Image
3.
4. # Ruta de la carpeta raíz que contiene las carpetas con
   imágenes
5. root_folder =
   '/home/proyectoiaolivo/datasets/olivo/tree/fotografias_olivo
   _png'
6. # Especifica el tamaño de los recortes
7. width, height = 1000, 1000
8. i=0
9. # Recorre todas las carpetas y archivos
10.   for root, dirs, files in os.walk(root_folder):
11.
12.       for file in files:
13.           # Comprueba si el archivo es una imagen
14.           if file.lower().endswith(('.png')):
15.               # Construye la ruta completa del archivo
16.               file_path = os.path.join(root, file)
17.               # Procesa la imagen
18.               image = cv2.imread(file_path)
19.               # Especifica el tamaño de los recortes
20.               width, height = 1000, 1000
21.
22.               # Calcula el número de recortes en horizontal
   y vertical
23.               num_cols = image.shape[1] // width
24.               num_rows = image.shape[0] // height
25.
26.               # Itera sobre las filas y columnas para
   generar los recortes
27.               for y in range(num_rows):
28.                   for x in range(num_cols):
29.                       # Calcula las coordenadas de inicio y
   fin para el recorte
30.                       x_start = x * width
31.                       x_end = (x + 1) * width
32.                       y_start = y * height
33.                       y_end = (y + 1) * height
34.
35.                       # Recorta la región de la imagen
36.                       cropped_image = image[y_start:y_end,
   x_start:x_end]
37.
38.                       # Guarda el recorte en un archivo
   (puedes ajustar la nomenclatura)
39.
   cv2.imwrite('/home/proyectoiaolivo/datasets/olivo/tree/datas
   et_recortes/'+f'{file[:-4]}'+f'_recorte_{y}_{x}.png',
   cropped_image)
40.
   print('/home/proyectoiaolivo/datasets/olivo/tree/dataset_rec
   ortes/'+f'{file[:-4]} CREADO, Num:{i}')
41.               i+=1
```

Código para generar .JSON compatible con Label-Studio.

```
import os
import ultralytics
ultralytics.checks()
from IPython import display
display.clear_output()
from IPython.display import display, Image, clear_output
from ultralytics import YOLO
import json
import cv2
    import torch
    from datetime import datetime
    HOME = os.getcwd()
    print(HOME)

    Images_folderpath="/home/proyectoiaolivo/datasets/olivo/tree/Yolo_Inference"

    ###Lectura de Img
    images = []
    images_name=[]
    count=0
    for img in sorted(os.listdir(Images_folderpath)):
        images.append(cv2.imread(os.path.join(Images_folderpath, img)))
        images_name.append(img)
        count += 1
    print("Se leyeron "+str(count)+" Archivos en "+Images_folderpath)

    ##Inferencia con modelo Yolov8
    model = YOLO('/home/proyectoiaolivo/notebooks/olivo/tree/OlivoModel_v03.pt')
    low_limit=0
    i=0
    annotations_json=[]
    while low_limit < len(images_name):
        results = model(images[low_limit:low_limit+300],conf=0.25,save=False)
        print("--------- Inferencia completa de imagenes "+str(low_limit)+"/"+str(low_limit+300)+" ---------")
        for result in results:
            results_json=[]
            for box in result.boxes.xyxyn:
                result_json = {
                        "original_width": 1000,
                        "original_height": 1000,
                        "image_rotation": 0,
                        "value": {
                             "x": box[0].item()*100,
                             "y": box[1].item()*100,
                             "width": (box[2]-box[0]).item()*100,
                             "height": (box[3]-box[1]).item()*100,
                             "rotation": 0,
                             "rectanglelabels": [
                             "olive"
                             ]
                             },
```

```
                        "from_name": "label",
                        "to_name": "image",
                        "type": "rectanglelabels",
                        "origin": "manual"
                    }
                    results_json.append(result_json)
                annotation_json={
                    "annotations": [
                        {
                            "completed_by": 1,
                            "result": results_json,
                        }
                    ],
                    "drafts": [],
                    "predictions": [],
                    "data": {
                        "image": "/data/local-
files/?d="+images_name[i]
                    },
                    "meta": {},
                    "updated_by": 1,
                    "comment_authors": []
                }
                annotations_json.append(annotation_json)
                i+=1
            low_limit+=300
        fecha_ = datetime.now()
        cadena_fecha_= fecha_.strftime("%Y-%m-%d_%H:%M:%S")
        with
open('/home/proyectoiaolivo/notebooks/olivo/tree/JSON/YoloBo
xes_'+cadena_fecha_+'.json', "w") as json_file:
            json.dump(annotations_json, json_file)

        print("JSON YoloBoxes_"+cadena_fecha_hora+" generado y
guardado")
```

Código para entrenamiento de modelos YOLOv8s.

```
from ultralytics import YOLO
from IPython.display import display, Image

!yolo task=detect mode=train model=yolov8s.pt
data=/home/proyectoiaolivo/notebooks/olivo/tree/datasets/DataSetOlivo/data.yaml
epochs=100
batch=4
plots=False
val=True
patience=500
   pretrained=True
   lr0=0.01
   lrf=0.01
```

Código para entrenamiento de modelos Faster R-CNN 101.

```
import detectron2
import cv2
from detectron2.utils.logger import setup_logger
setup_logger()
import random
import numpy as np
from detectron2 import model_zoo
from detectron2.config import get_cfg
from detectron2.engine import DefaultPredictor
   from detectron2.data.catalog import DatasetCatalog
   from detectron2.utils.visualizer import Visualizer
   from detectron2.data import MetadataCatalog
   from detectron2.data.datasets import
register_coco_instances
   register_coco_instances("olive_train", {},
"/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_TrainCOCO/result.json",

"/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_TrainCOCO/images")
   register_coco_instances("olive_val", {},
"/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_ValidCOCO/result.json",

"/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_ValidCOCO/images")

   cfg = get_cfg()
   cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"))
   cfg.DATASETS.TRAIN = ("olive_train",)
   cfg.DATASETS.TEST = ("olive_val",)
   cfg.MODEL.DEVICE='cuda'
   cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml")
   cfg.SOLVER.IMS_PER_BATCH = 4
   cfg.DATALOADER.NUM_WORKERS = 2
   cfg.SOLVER.MAX_ITER = 100
   cfg.SOLVER.BASE_LR = 0.01
   cfg.SOLVER.STEPS = []
   cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
```

```
31.   cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
32.   cfg.TEST.EVAL_PERIOD = 1
```

```
1. from detectron2.evaluation import COCOEvaluator
2. from detectron2.engine import DefaultTrainer
3.
4. class CocoTrainer(DefaultTrainer):
5.
6.   @classmethod
7.   def build_evaluator(cls, cfg, dataset_name,
   output_folder=None):
8.
9.     if output_folder is None:
10.           os.makedirs("coco_eval", exist_ok=True)
11.           output_folder = "coco_eval"
12.
13.       return COCOEvaluator(dataset_name, cfg, False,
   output_folder)
```

```
1. %%capture cap
2. #Modelo 1
3. cfg.OUTPUT_DIR = "./output/01"
4. os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
5. trainer = CocoTrainer(cfg)
6. import time as t
7.
8. s1 = t.time()
9. trainer.resume_or_load(resume=False)
10.   trainer.train()
11.   s2 = t.time()
12.   print(s2-s1)
```

Código para entrenamiento de modelos RetinaNet 101.

```
import detectron2
import numpy as np
import random
import cv2

from detectron2.utils.logger import setup_logger
setup_logger()

from detectron2 import model_zoo
     from detectron2.config import get_cfg
     from detectron2.engine import DefaultPredictor
     from detectron2.data import MetadataCatalog
     from detectron2.utils.visualizer import Visualizer
     from detectron2.data.catalog import DatasetCatalog

     from detectron2.data.datasets import
   register_coco_instances
     register_coco_instances("olive_train2", {},
   "/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_Tra
   inCOCO/result.json",
   "/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_Tra
   inCOCO/images")
     register_coco_instances("olive_val2", {},
   "/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_Val
   idCOCO/result.json",
   "/home/proyectoiaolivo/Downloads/Bloques/DataSetOlivo_Val
   idCOCO/images")

     cfg = get_cfg()
     cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
     cfg.merge_from_file(model_zoo.get_config_file("COCO
   -Detection/retinanet_R_101_FPN_3x.yaml"))
     cfg.DATASETS.TRAIN = ("olive_train2",)
     cfg.DATASETS.TEST = ("olive_val2",)
     cfg.INPUT.RANDOM_FLIP="none"
     cfg.MODEL.DEVICE='cuda'
     cfg.DATALOADER.NUM_WORKERS = 2
     cfg.MODEL.WEIGHTS =
   "/home/proyectoiaolivo/Downloads/Bloques/model_final_971a
   b9.pkl"
     cfg.SOLVER.MAX_ITER = 100
     cfg.SOLVER.IMS_PER_BATCH = 4
     cfg.SOLVER.BASE_LR = 0.01
     cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
     cfg.SOLVER.STEPS = []
     cfg.MODEL.RETINANET.NUM_CLASSES = 1
     cfg.TEST.EVAL_PERIOD = 1
```

```
from detectron2.evaluation import COCOEvaluator
from detectron2.engine import DefaultTrainer

class CocoTrainer(DefaultTrainer):

  @classmethod
  def build_evaluator(cls, cfg, dataset_name,
  output_folder=None):

```

```
    if output_folder is None:
            os.makedirs("coco_eval", exist_ok=True)
            output_folder = "coco_eval"

       return COCOEvaluator(dataset_name, cfg, False,
   output_folder)
```

```
%%capture cap
#Modelo 1
cfg.OUTPUT_DIR = "./retina2/01"
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
import time as t

s1 = t.time()
trainer.resume_or_load(resume=False)
   trainer.train()
   s2 = t.time()
   print(s2-s1)
```

Código para inferencia de modelos YOLOv8s.

```
from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

model =
YOLO('/home/proyectoiaolivo/notebooks/olivo/tree/runs/detect
/YOLO1/weights/best.pt')
results =
model('/home/proyectoiaolivo/datasets/olivo/tree/InferenciaM
odelosTe/IMG_2513_recorte_1_2.png', save=True, conf=0.5)
```

Código para inferencia de modelos Faster R-CNN 101.

```
import detectron2
from detectron2.utils.logger import setup_logger
from matplotlib import pyplot as plt
setup_logger()

import numpy as np
import cv2
import random

   from detectron2 import model_zoo
   from detectron2.config import get_cfg
   from detectron2.engine import DefaultPredictor
   from detectron2.data.catalog import DatasetCatalog
   from detectron2.data import MetadataCatalog
   from detectron2.utils.visualizer import Visualizer

   cfg = get_cfg()
   cfg.OUTPUT_DIR = "./output/04/"

   cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_R_101_FPN_3x.yaml"))
   cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR,
"model_final.pth")
   cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
   cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1

   predictor = DefaultPredictor(cfg)

   outputs = predictor(im)
```

Código para inferencia de modelos RetinaNet 101.

```
1. import detectron2
2. from detectron2.utils.logger import setup_logger
3. from matplotlib import pyplot as plt
4. setup_logger()
5.
6. import numpy as np
7. import cv2
8. import random
9.
10.     from detectron2 import model_zoo
11.     from detectron2.config import get_cfg
12.     from detectron2.engine import DefaultPredictor
13.     from detectron2.data.catalog import DatasetCatalog
14.     from detectron2.data import MetadataCatalog
15.     from detectron2.utils.visualizer import Visualizer
16.
17.     cfg = get_cfg()
18.     cfg.OUTPUT_DIR = "./retina2/10/"
19.
20.     cfg.merge_from_file(model_zoo.get_config_file("COCO
   -Detection/retinanet_R_101_FPN_3x.yaml"))
21.     cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR,
   "model_final.pth")
22.     cfg.TEST.DETECTIONS_PER_IMAGE = 1000
23.     cfg.MODEL.RETINANET.SCORE_THRESH_TEST = 0.5
24.     cfg.MODEL.RETINANET.NUM_CLASSES = 1
25.
26.     predictor = DefaultPredictor(cfg)
27.
28.     outputs = predictor(im)
```