UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Aplicación de técnicas de visión artificial en dispositivos de bajo costo para mejorar la eficiencia en la agricultura de precisión

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Jaramillo Hernandez, Juan Felipe

Tutor/a: Julian Inglada, Vicente Javier

Cotutor/a: Rincón Arango, Jaime Andrés

CURSO ACADÉMICO: 2022/2023

Universitat Politècnica
de València

Departamento de Sistemas Informáticos y Computación

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Trabajo Fin de Máster

Aplicación de técnicas de visión artificial en dispositivos de bajo costo para mejorar la eficiencia en la agricultura de precisión

Autor(a): Juan Felipe Jaramillo Hernández
Director(a): Vicente Javier Julian Inglada
Cotutor(a): Jaime Andrés Rincón Arango

Valencia, Septiembre - 2023

Este Trabajo Fin de Máster se ha depositado en el Departamento de Sistemas Informàticos y Computación de la Universitat Politècnica de València para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Título: Aplicación de técnicas de visión artificial en dispositivos de bajo costo para mejorar la eficiencia en la agricultura de precisión

Septiembre - 2023

Autor(a):	Juan Felipe Jaramillo Hernández
Director(a):	Vicente Javier Julian Inglada Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València
Co-director(a):	Jaime Andrés Rincón Arango Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València

Resum

En el context del creixement tecnològic en els últims anys, gràcies al treball distribuït i les fonts de codi obert, la visió per ordinador ha estat una tecnologia innovadora que ha revolucionat la forma en què les màquines poden interactuar i comprendre el món visual que ens envolta. És indispensable aprofitar aquesta tecnologia per donar suport a la solució d'un dels problemes principals actuals al món: la fam en una població creixent. En aquest treball es concibeix, dissenya, implementa i opera un mètode de visió per ordinador i intel·ligència artificial per a la detecció de fruites amb una estimació integrada de la seua profunditat en l'escena. Aquest mètode es pot aprofitar per integrar-se en sistemes autònoms de collita de fruites o tasques de fenotipatge. El model de detecció d'objectes amb estimació de profunditat (DOD) s'entrena i avalua per a les tasques de detecció d'objectes comuns en COCO [1] i la detecció de fruites a MinneApple [2, 3, 4], i es compara amb els models de l'estat de l'art actual. Els resultats obtinguts demostren l'eficiència del mètode proposat per a operar-se en sistemes embebuts amb un equilibri de precisió i velocitat suficient per a aplicacions en temps real en dispositius de vora en el context de l'Internet de les Coses (IoT).

Resumen

En el contexto del crecimiento tecnológico en los últimos años gracias al trabajo distribuido y las fuentes de código abierto, la visión por computadora ha sido una tecnología innovadora que ha revolucionado la forma en que las máquinas pueden interactuar y comprender el mundo visual que nos rodea. Es indispensable aprovechar esta tecnología para apoyar en la solución de uno de los problemas principales actuales en el mundo: el hambre en una población creciente. En este trabajo se concibe, diseña, implementa y opera un método de visión por computadora e inteligencia artificial para la detección de frutos con una estimación integrada de su profundidad en la escena. Este método se puede aprovechar para integrarse en sistemas autónomos de cosecha de frutos o tareas de fenotipado. El modelo de detección de objetos con estimación de profundidad (DOD) se entrena y evalúa para las tareas de detección de objetos comunes en COCO [1] y la detección de frutos en MinneApple [2, 3, 4], y se compara con los modelos del estado del arte actual. Los resultados obtenidos demuestran la eficiencia del método propuesto para operarse en sistemas embebidos con un balance de precisión y velocidad suficiente para aplicaciones en tiempo real en dispositivos de borde en el contexto del Internet de las Cosas (IoT).

Abstract

In the context of technological growth in recent years, driven by distributed work and open-source resources, computer vision has been an innovative technology that has revolutionized the way machines can interact with and comprehend the visual world around us. Leveraging this technology is crucial to addressing one of the current world's major challenges: hunger in a growing population. This work conceives, designs, implements, and operates a computer vision and artificial intelligence method for fruit detection with integrated depth estimation in the scene. This method can be utilized for integration into autonomous fruit harvesting systems or phenotyping tasks. The Depth Object Detector (DOD) Model is trained and evaluated for common object detection tasks in COCO [1] and fruit detection in MinneApple [2, 3, 4]. It is compared to current state-of-the-art models. The results obtained demonstrate the efficiency of the proposed method for operation on embedded systems, with a balance of accuracy and speed suitable for real-time applications on edge devices in the context of the Internet of Things (IoT).

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación

En los últimos años, el desarrollo y la implementación de tecnologías como la inteligencia artificial (IA) [5, 6, 7], el internet de las cosas (IoT) [8, 9], la electrónica [10, 11, 12] y la computación en borde [13], se han vuelto cruciales para mejorar la eficiencia energética, la autonomía y la sostenibilidad de los sistemas agrícolas en un contexto mundial que plantea desafíos relacionados con la seguridad alimenticia por una población humana que crece exponencialmente [14].

La cosecha es uno de los procesos base en la cadena de producción alimenticia, ya que es la actividad agrícola que se encarga de recolectar frutos maduros y listos para ser consumidos o utilizados con fines comerciales. La innovación tecnológica de este proceso es crucial para mejorar la productividad agrícola al lograr una mejor gestión del suelo, que al mismo tiempo genere un impacto significativo en la resiliencia al cambio climático y en la remediación de la contaminación ambiental [15].

Por otro lado, la cosecha ha sido tradicionalmente un proceso manual realizado por mujeres y hombres. Sin embargo, en la tendencia de la era moderna de anteponer el bien individual ante el bien común [16], los empleadores de los campos agrícolas han optado por la desregularización espacial al filializar las responsabilidades laborales a cada empleado, planteando complejos problemas socioculturales debido a condiciones laborales precarias con contratos injustos [17, 18].

La integración de sistemas complejos de adquisición, procesamiento y control de señales pueden dar lugar a plataformas automatizadas que aborden desde un punto de vista operacional las problemáticas enunciadas anteriormente, con el yuxtapuesto de incrementar la productividad agrícola.

En este trabajo, se presenta el modelo DOD (*Depth Object Detector*) de visión artificial para la detección especializada de frutos con estimación de profundidad, basado en las técnicas del estado de arte actual y adaptado para sistemas de bajo costo. En el marco del proyecto Nacional COSASS (*COordinated intelligent Services for Adaptive Smart areaS*) [19], existe la posibilidad implementar esta técnica de adquisición de datos en robots autónomos de cosecha, como ya se ha evidenciado su potencial como tecnología transformadora [20, 21, 22, 23, 24].

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar un modelo de visión artificial para la detección de frutos con estimación de profundidad.

1.2.2. Objetivos Específicos

1. Construir, mediante el uso de datos de libre acceso, un conjunto de datos que integre la información tanto de la detección de objetos, como la profundidad estimada para cada objeto.
2. Diseñar e implementar un modelo de visión artificial para detectar objetos y estimar su profundidad, basado en el estado del arte actual.
3. Adaptar la arquitectura del modelo de visión artificial para su implementación en dispositivos de bajo costo como microcontroladores o sistemas embebidos con bajos recursos computacionales.
4. Evaluar, a través de un protocolo de pruebas, el desempeño del modelo implementado sobre el conjunto de datos, y sobre un sistema embebido.

1.3. Metodología

Este trabajo se divide en los siguientes capítulos:

1. **Estado del arte**: en este capítulo, se realiza una revisión exhaustiva de la estimación de profundidad, la detección de objetos, los conjuntos de datos y los trabajos relacionados en el contexto teórico y práctico del método propuesto.
2. **Modelo de Detección con Estimación de Profundidad**: se describe la arquitectura del modelo DOD, abordando su inspiración, sus componentes, el proceso de inferencia, la función de pérdida, las métricas de evaluación y la asignación de etiquetas adoptada.
3. **Entrenamiento**: se detalla el proceso de entrenamiento del modelo DOD, incluyendo la adaptación de conjuntos de datos para la estimación de profundidad, las estrategias de aumento de datos y los protocolos de prueba para las fases de entrenamiento, la estrategia de entrenamiento implementada, y la adaptación a sistemas embebidos mediante la cuantización.
4. **Resultados**: en este capítulo se presentan los resultados obtenidos en diferentes escenarios. Se evalúa el rendimiento del modelo DOD en diferentes conjuntos de datos para la detección de objetos comunes y la detección de frutos. También se evalúa el rendimiento del modelo cuantizado en el sistema embebido Raspberry Pi 4.
5. **Conclusiones**: este capítulo contiene las conclusiones extraídas a partir de los resultados obtenidos en los experimentos. Se discuten los logros y limitaciones del método propuesto y se proponen posibles direcciones para futuros trabajos de investigación.

Capítulo 2

Estado del Arte

Gracias al desarrollo científico y la atención en los últimos años, técnicas novedosas de la visión por computador permiten la síntesis y representación de escenas tridimensionales (3D). Algunas tecnologías a destacar pueden ser los campos de luz [25], los campos de radiancia neuronal [26] y las nubes de puntos 3D [27]. Estos métodos buscan representar digitalmente, con alta resolución, las propiedades de los rayos de luz en una escena real.

Otros métodos, computacionalmente más económicos, pueden estimar la profundidad de una escena estática (en el mayor de los casos) a partir de imágenes bidimensionales (2D). Incluso mejor, se puede optimizar la estimación de la profundidad monóscopica concentrándose únicamente en los objetos clave mediante técnicas de detección de objetos [28, 29, 30, 31]. Esta estrategia maximiza la eficiencia computacional y optimiza el tiempo de procesamiento al priorizar el primer plano de la escena, convirtiéndose en una opción atractiva para aplicaciones en tiempo real, como la conducción autónoma [32, 33], la robótica [34], la vigilancia [35] o la cirugía asistida [36].

2.1. Estimación de Profundidad

La estimación de profundidad de una escena es un desafío fundamental en la visión por computador que consiste en predecir la profundidad de los objetos en el primer o segundo plano de una escena a partir de una o más imágenes 2D al estimar la profundidad de cada píxel.

Se aborda tradicionalmente como un problema matemático que busca modelar el comportamiento biológico de los mamíferos con visión estereoscópica [37]. Este enfoque se basa en la triangulación de la distancia relativa de cada punto en la escena utilizando la disparidad entre dos vistas de la misma escena con perspectivas ligeramente desplazadas [38].

Al trazar la intersección de un punto espacial en ambas imágenes, se genera un plano epipolar que permite medir geométricamente la distancia del punto espacial con respecto al punto de vista central de la cámara virtual entre las dos imágenes [39]. Si bien las técnicas basadas en este enfoque pueden lograr una estimación precisa del mapa de profundidad, su implementación demanda altos requisitos de memoria y hardware, lo cual se traduce en tiempos de cálculo prolongados.

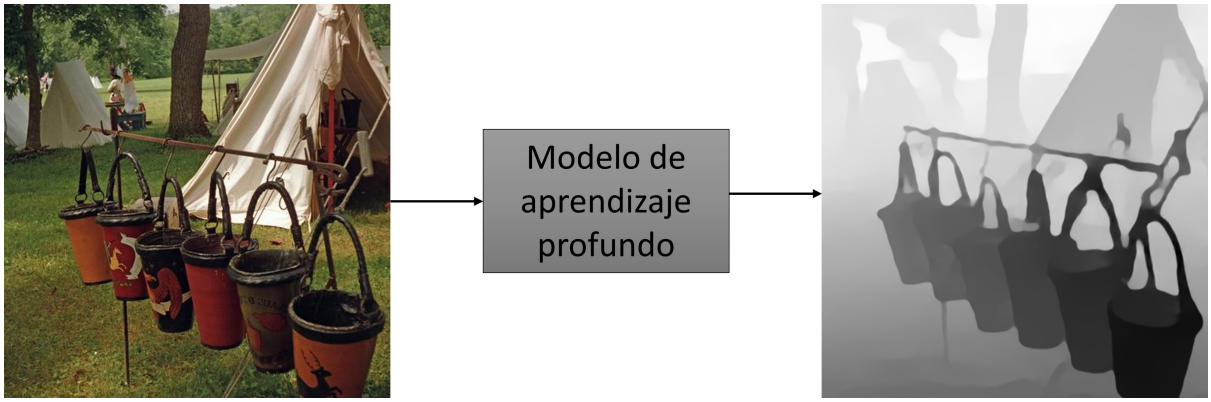


Figura 2.1: Imágenes tomadas del conjunto de datos ReDWeb creado por Xian *et al.* [44].

Abordar esta tarea con imágenes monoscópicas es un problema mal condicionado ya que no existen referencias para la triangulación al perder las correspondencias de la visión estéreo [40]. Sin embargo, en los últimos años, con la ploriferación de conjuntos de datos, el avance del aprendizaje profundo y gracias a la capacidad de las redes neuronales de representar matemáticamente el contexto espacial de las escenas en el conjunto de datos, es posible intentar predecir directamente el mapa de profundidad de una escena a partir de una sola imagen 2D como entrada [41, 42, 43], como se observa en la Figura 2.1.

En la estimación de profundidad monocular, el estado del arte actual está definido por Ranft *et al.* [45, 46], que en el año 2021 introducen una arquitectura que utiliza los transformes de visión (ViT) [47] en lugar de las redes convoluciones como columna vertebral para tareas de predicción densa. El transformer tiene un campo receptivo global a una resolución constante y relativamente alta, permitiendo predicciones más detalladas y globalmente coherentes en comparación con las redes totalmente convolucionales, especialmente cuando hay una gran cantidad de datos de entrenamiento disponibles.

En el año 2023, Zhao *et al.* [48] presentan su arquitectura VPD (Percepción Visual con un modelo de Difusión pre-entrenado), que aprovecha la información semántica de un modelo de difusión de texto a imagen pre-entrenado en tareas de percepción visual al utilizar el autoencoder de eliminación de ruido pre-entrenado como columna vertebral al adaptarlo mediante la inserción de entradas textuales adecuadas para una mejor alineación con la etapa pre-entrenada, permitiendo una interacción más efectiva entre los contenidos visuales, en este caso, entre imágenes de tres canales y sus mapas de profundidad [48].

Ambos métodos mencionados anteriormente requieren alta capacidad de cómputo, lo cual los descarta como las opciones más adecuadas para aplicaciones en tiempo real. Por otro lado, Peluso *et al.* [49] proponen un método de estimación de profundidad monocular eficiente para microcontroladores, basada en una red neuronal convolucional ligera con una arquitectura piramidal poco profunda. Mediante estrategias de optimización para realizar cálculos con datos de 8-bits, y mapeando la descripción de alto nivel de la red a capas de bajo nivel optimizadas para la arquitectura del microcontrolador objetivo, los resultados experimentales muestran que es posible obtener estimaciones de profundidad lo suficientemente precisas en microcontroladores [49].

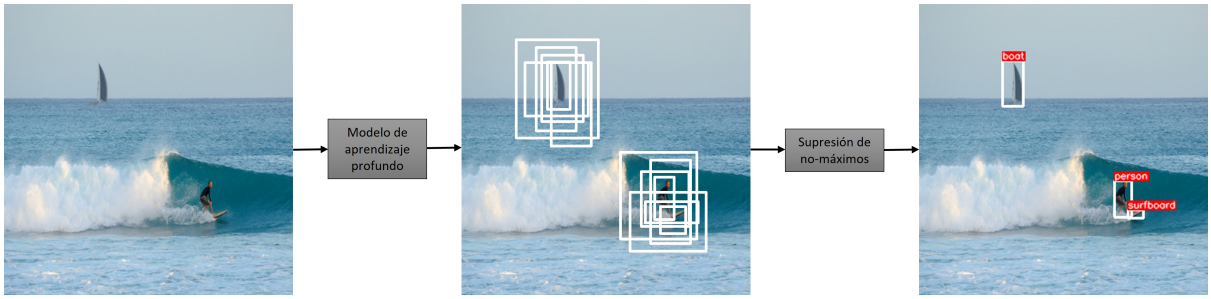


Figura 2.2: Imágenes tomadas del conjunto de datos COCO [1].

2.2. Detección de Objetos

La detección de objetos es un técnica fundamental de la visión por computador que permite a los sistemas informáticos identificar y localizar objetos dentro de imágenes o vídeos mediante algoritmos avanzados para analizar patrones visuales y distinguir los objetos de fondo en una escena. La detección de objetos ha revolucionado la forma en que las máquinas pueden interactuar y comprender el mundo visual que nos rodea.

Entre 1990 y 2010, la detección de objetos se basaba principalmente en el uso de filtros dinámicos y algoritmos iterativos para representar matemáticamente los patrones discriminativos en tareas con formas altamente diferenciables [39]. Sin embargo, en la última década, el aprendizaje automático ha mejorado la capacidad de discriminación entre categorías de objetos, lo que ha permitido aplicar esta técnica incluso en tareas dentro de escenas de alta complejidad.

Los algoritmos de detección de objetos basados en aprendizaje profundo son sistemas complejos debido a la necesidad de representar todos los posibles cuadros delimitadores (*bounding boxes*) de objetos en una imagen de entrada seleccionando aquellos con máxima confianza a partir de ciertos criterios, como se observa en la Figura 2.2.

El desafío radica en la implementación y conexión entre las múltiples etapas del algoritmo, como lo es el modelo de predicción y la etapa de inferencia. En el entrenamiento, es necesario usar estrategias dinámicas de asignación de etiquetas para asegurar que la pérdida agregada al gradiente tenga la información más relevante al asignar las referencias adecuadas a las predicciones adecuadas [50, 51].

Entre las arquitecturas que marcan el estado del arte actual, el algoritmo YOLO (*You Only Look Once*), introducido en el año 2015 por Redmon *et al.* [52], se ha destacado en la detección de objetos por su equilibrio entre velocidad y precisión gracias a su evolución mediante iteraciones sucesivas que mejoran las versiones anteriores para superar limitaciones y mejorar el rendimiento [53].

La idea general en el algoritmo YOLO consiste en dividir la imagen de entrada en $n = m \times m$ celdas contiguas (véase la Figura 2.3), donde cada una de las celdas predice B cuadros delimitadores, cada uno con cuatro predicciones l, t, r, b con las dimensiones del cuadro relativas al centro de la celda, y una confianza C de pertenecer a una clase determinada. Las $B \times n$ predicciones se pasan por un algoritmo de Supresión No Máxima (NMS por sus siglas en inglés) para seleccionar los cuadros delimitadores con mayor confianza por cada objeto detectado en la imagen.

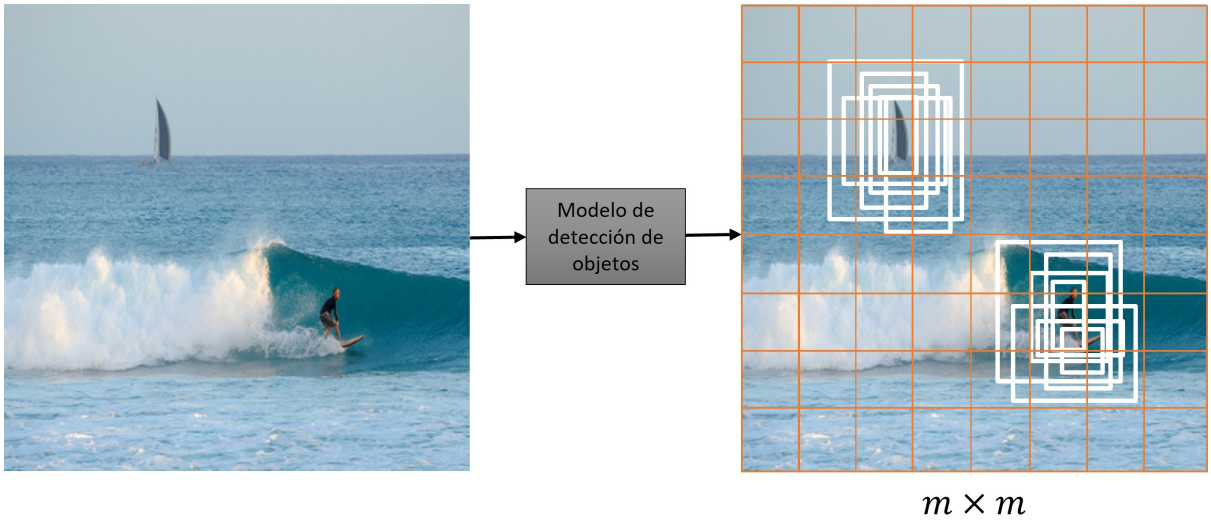


Figura 2.3: La salida de un modelo de detección basado en YOLO [53] tiene dimensiones $m \times m \times B \times (4 + nc)$, siendo nc el número de clases detectables, 4 distancias del cuadro delimitador, B el número de predicciones por cada celda y $m \times m$ el número de celdas de predicción en la imagen, señaladas con color naranja en la figura. Imágenes tomadas del conjunto de datos COCO [1].

El modelo YOLOv8, propuesto por Jocher *et al.* [54] en febrero de 2023, marca el estado del arte actual en la detección de objetos. Es una arquitectura completamente convolucional y se destaca por contar con tres capas de predicción para identificar objetos en distintas resoluciones. Cada capa de predicción cuenta con dos cabezales desacoplados, uno de clasificación y otro de regresión. Estas tres capas se realimentan mediante un cuello que forma un lazo cerrado de control, regulando el camino más corto y más largo del gradiente, mejorando el aprendizaje y la convergencia de la red.

Por otro lado, la revolución de los modelos de atención cruzada, como ChatGPT [55], ha marcado un avance en la IA generativa para tareas de texto-a-texto, texto-a-imagen e imagen-a-imagen. El modelo SAM (*Segment Anything*), propuesto recientemente por el equipo de investigación de Meta [56], consta de un codificador de imágenes basado en ViT y un decodificador de máscara guiado por indicaciones. Chaoning *et al.* [57] proponen MobileSAM (*Mobile Segment Anything*), una versión más óptima y rápida que SAM, con las mismas prestaciones pero menor número de parámetros, ideal para aplicaciones móviles para la detección de objetos y la segmentación semántica.

En la actualidad, los modelos generativos de detección y segmentación de imágenes tienen un gran potencial para generar nuevos conjuntos de datos, especialmente en tareas donde la disponibilidad de datos es limitada, abriendo nuevas posibilidades para la investigación y aplicaciones prácticas.

2.3. Conjuntos de Datos

En esta sección se enuncian algunos de los conjuntos de datos más populares para la estimación de profundidad y la detección de objetos.

Los conjuntos de datos para la estimación de profundidad se componen de imágenes RGB y sus correspondientes mapas de profundidad como imágenes de un solo canal. Algunos de los conjuntos de datos más destacables y de libre acceso son:

- **ReDWeb** [44] que contiene 3,600 imágenes recolectadas de imágenes estéreo en la Web y post-procesadas para obtener su mapa de disparidad.
- **Virtual KITTI** [58, 59] es un conjunto de datos sintético foto-realístico diseñado para múltiples tareas en el campo de la detección autónoma, como la detección de objetos, el seguimiento de multi-objetos, la segmentación semántica de la escena, el flujo óptico y la estimación de la profundidad. Consta de 5 mundos virtuales bajo diferentes circunstancias de iluminación y climatológicas, con un total de 21,260 imágenes completamente anotadas.
- **Hypersim** [60] es un conjunto de datos sintético foto-realístico para comprensión holística de la escenas de interior. Cuenta con 77,400 imágenes de 461 escenas, cada imagen contiene las anotaciones completas a nivel de píxel de su información geométrica en la escena, como su profundidad, así como además etiquetas de tareas de detección y segmentación semántica de instancias.

Los conjuntos de datos para la detección de objetos se componen de imágenes RGB y sus correspondientes etiquetas con las coordenadas de los cuadros delimitadores y las clases pertenecientes. Algunos de los conjuntos de datos más populares y de libre acceso son los siguientes:

- **MS COCO** [1] (*Microsoft Common Objects in Context*) es un conjunto de datos de gran escala para la detección de objetos, la segmentación y la detección de puntos-clave. Cuenta con 328,000 imágenes naturales en diferentes escenas de interior y exterior.
- **ImageNet** [61] es el conjunto de datos de mayor escala con alrededor de 14 millones de imágenes naturales anotadas para tareas de clasificación de imágenes y alrededor de 1 millón de imágenes para la tarea de detección de objetos.
- **PASCAL VOC** [62] (*Visual Object Classes*) es un conjunto de datos con alrededor de 3,000 imágenes etiquetadas para la detección de objetos y es ampliamente utilizado como conjunto de datos para la evaluación de modelos de clasificación, detección y segmentación.

Para las tareas de detección de frutos se destacan los siguientes conjuntos de datos:

- **MinneApple** [2, 3, 4] es un conjunto de datos de evaluación para tareas de detección y segmentación de manzanas. Consta de alrededor de 41,000 objetos anotados en 1,000 imágenes de campo en árboles de manzanas en cultivos controlados. Los frutos están etiquetadas mediante máscaras poligonales para cada instancia de objeto con el fin de mejorar la precisión en la detección, localización y segmentación.
- **Apples** [63], creado por Arfiani Nur Sayidah, es un conjunto de datos con alrededor de 8,000 instancias etiquetadas como manzanas en 700 imágenes.

2.4. Trabajos Relacionados

Wang *et al.* [28] (2021) presentan un enfoque de detección de objetos y estimación de profundidad en tiempo real basado en redes neuronales convolucionales. Para la estimación de profundidad, introducen la visión binocular en la red de estimación de disparidad basada en visión monocular, y utilizamos la restricción epipolar para mejorar la precisión de la predicción. Finalmente, integran la ubicación 2D del objeto detectado con la información de profundidad para lograr una detección y estimación de profundidad en tiempo real. Los resultados demuestran que el enfoque propuesto obtiene mejores resultados en comparación con los métodos convencionales. Sin embargo, la computación es compleja y costosa en términos de procesamiento.

Lee *et al.* [29] (2022) presentan una aproximación simplificada de la profundidad en los objetos de imágenes estereoscópicas al cuantificar los valores de profundidad en un pequeño número de valores representativos. Esto permite evitar la complejidad en los cálculos al estimar únicamente un valor representativo de profundidad para cada instancia de objeto, y no tener que estimar los valores de todos los píxeles que contienen el objeto. Sus resultados en el conjunto de datos KITTI [64] demuestran que, a pesar de la baja complejidad, su método de aproximación conduce a una mejora significativa en el rendimiento de detección de objetos.

Fan *et al.* [30] (2022), buscan mejorar el rendimiento en tiempo real de la reconstrucción 3D al proponer un enfoque novedoso para reducir el consumo de recursos computacionales mediante la extracción de regiones significativas de los mapas de profundidad mediante la fusión de detección de objetos en 2D y estimación de profundidad monocular auto-supervisada.

Usman *et al.* [31] (2022) introducen un sistema de fusión punto-píxel para la detección y clasificación de objetos con información de profundidad para un sistema de conducción autónoma. Específicamente, combinan los puntos de un sensor LIDAR con una imagen 2D la cual se procesa mediante un modelo de detección de objetos que extrae las regiones de interés para determinar la profundidad en los objetos destacados, así descartando el resto de los puntos de lidar y conservando únicamente las regiones de interés.

Por otro lado, Bedettrin *et al.* [65] (2022) plantean una metodología de evaluación para medir la mejora de los modelos de detección de objetos que utilizan la información de profundidad en su entrada junto con la imagen en color. Sus resultados concluyen en que la información de profundidad no aporta gran ganancia en el desempeño de los modelos de detección.

Respecto a la detección de frutos, Häni *et al.* [2, 3, 4] (2019) presentan un nuevo conjunto de datos para avanzar en el estado del arte en detección, segmentación y conteo de frutos en entornos de huertos. Adicionalmente, presentan un análisis de rendimiento de referencia para la tarea usando distintas arquitecturas de modelos de detección de objetos, junto con su arquitectura propuesta Tiled Faster R-CNN.

Xiang *et al.* [66] (2021) proponen un sistema para la detección de frutos sueltos de palma de aceite utilizando la arquitectura Faster R-CNN [67] en un hardware NVIDIA Jetson TX2. En su estudio, se recopilaron 500 imágenes de frutos sueltos de una finca de palma de aceite en Bukit Bangkong, Selangor, durante el proceso de cosecha. El modelo alcanzó una precisión de aproximadamente 94 % para un umbral

de intersección de unión igual a 0.5. demostrando que el sistema desarrollado es capaz de detectar frutos sueltos de palma de aceite con precisión y tiene el potencial de contribuir al desarrollo de un sistema de recolección automática de frutos.

Nagaraju *et al.* [68] (2022) proponen una técnica de reconocimiento de frutos basada en la arquitectura YOLOv5 [69] que detecte chirimoyas, granadas y bayas de cera. Para esto, recolectaron imágenes de frutos en un entorno real y se preprocesaron para crear un conjunto de datos privado. Con una precisión media promedio de 89.4 % en el umbral 0.5 de intersección de unión, demuestran que su sistema tiene implicaciones significativas para los sistemas autónomos de recolección de frutos en huertos.

Dentro del marco de este trabajo, el método propuesto por Lee *et al.* [29] (2022) para la estimación de profundidad es el que tiene mayor relación con este trabajo, ya que aborda la estimación de la profundidad como un único valor representativo para cada instancia de objeto detectado. La novedad del modelo propuesto en este trabajo está en que la estimación de profundidad se acopla como un cabezal adicional en las capas de predicción del modelo de detección. En el contexto de la detección de frutos, Häni *et al.* [2, 3, 4] serán el punto de comparación del modelo DOD propuesto en este trabajo.

Capítulo 3

Modelo de Detección con Estimación de Profundidad

3.1. Inspiración del Modelo

La arquitectura propuesta del modelo de detección de objetos con estimación de profundidad DOD (*Depth Object Detector*) se basa en el modelo de detección de objetos YOLOv8 [54], cuyo desempeño en velocidad y precisión establece el estado del arte a fecha de publicación de este trabajo.

YOLOv8 [54] es un modelo completamente convolucional de código abierto [70] y su arquitectura se describe en Anexos: Figura 1. Se escogió por los siguientes aspectos a destacar, entre otros:

- Se basa en una arquitectura ELAN (*Efficient Layer Aggregation Networks*) [71], que a partir de un lazo de control para la ruta del gradiente más corta y más larga, busca mejorar la convergencia y la eficiencia de la red desde su diseño.
- Las predicciones finales se generan en tres niveles diferentes con campos receptivos de distinta resolución respecto a las dimensiones de la entrada. Además, cada nivel de detección se descompone en cabezales de predicción desacoplados para predecir de forma separada el cuadro delimitador y la clase de un objeto.
- A diferencia de versiones anteriores, cuya regresión del cuadro delimitador se basa en puntos de anclaje fijos inicializados previamente al entrenamiento [53], el cabezal de regresión es libre de anclaje [72] y predice directamente una distancia *left*, *right*, *top*, *bottom* relativa al centro de la celda de predicción.
- El entrenamiento adopta la estrategia de asignación de etiquetas TOOD (*Task-aligned One-stage Object Detection*) [50], agilizando el entrenamiento y la convergencia al seleccionar un número *top-*k de predicciones positivas para cada referencia basándose en su puntuación ponderada de clasificación y regresión.
- Se introduce la función objetiva de Pérdida de Distribución Focal (DFL por sus siglas en inglés) [73] en la regresión del cuadro delimitador, que penaliza una distribución de probabilidad en vez de una única estimación por cada valor, mejorando la precisión y sensibilidad en objetos complejos o ambiguos.

Modelo	Número de parámetros (M)	mAP 50-95 (%)
YOLOv8n	3.2	37.3
YOLOv8s	11.2	44.9
YOLOv8m	25.9	50.2
YOLOv8l	43.7	52.9
YOLOv8x	68.2	53.9

Cuadro 3.1: Comparación de las diferentes versiones de la arquitectura YOLOv8 [54] y su precisión promedio media (mAP) sobre el conjunto de datos COCO [1]. Información extraída de [70].

El modelo YOLOv8 [54] ha sido implementado en diferentes tamaños y entrenado con el conjunto de datos COCO [1]. La Tabla 3.1 resume el número de parámetros para cada versión cuya única diferencia reside en la profundidad de los mapas de características.

3.2. Arquitectura Propuesta

La Figura 3.1 presenta la arquitectura propuesta para la red neuronal convolucional del modelo de detección de objetos con estimación de profundidad DOD.

Considerando la Tabla 3.1, la arquitectura de YOLOv8n se ha modificado hasta reducirse aproximadamente a 1 millón de parámetros, en línea con los objetivos específicos 2 y 3 de este trabajo, para sistemas de bajo costo.

La arquitectura de la red se divide en tres partes: una extracción de características en forma piramidal seguida de un cuello de control para el gradiente, que realimenta tres niveles de predicción con una reducción de 8, 16 y 32 veces la dimensión de entrada. Finalmente, las salidas de la red por cada nivel se dan en los cabezales de predicción desacoplados, que predicen el cuadro delimitador, la clase y la profundidad relativa de los objeto en escena.

Se espera que las imágenes de entrada tengan una dimensión estándar de 320x320 píxeles en tres canales RGB. La resolución mínima posible es de 32x32, por otro lado, las dimensiones siempre deben ser divisibles entre 8.

3.2.1. Componentes

A continuación, se describen cada uno de los bloques que componen la red de la Figura 3.1:

- **Conv**: Módulo de convolución compuesto por una convolución espacial en 2D definida por un tamaño de kernel k, un tamaño de paso (*stride*) s, un tamaño de relleno (*padding*) p y un tamaño de filtros de entrada c_{in} y de salida c_{out}. La salida de la convolución pasa por una normalización del batch 2D [74] seguida de la función de activación SiLU (Unidad Lineal Sigmoide [75], Ecuación 3.1).

$$\text{silu}(x) = x\left(\frac{1}{1+e^{-x}}\right) \tag{3.1}$$

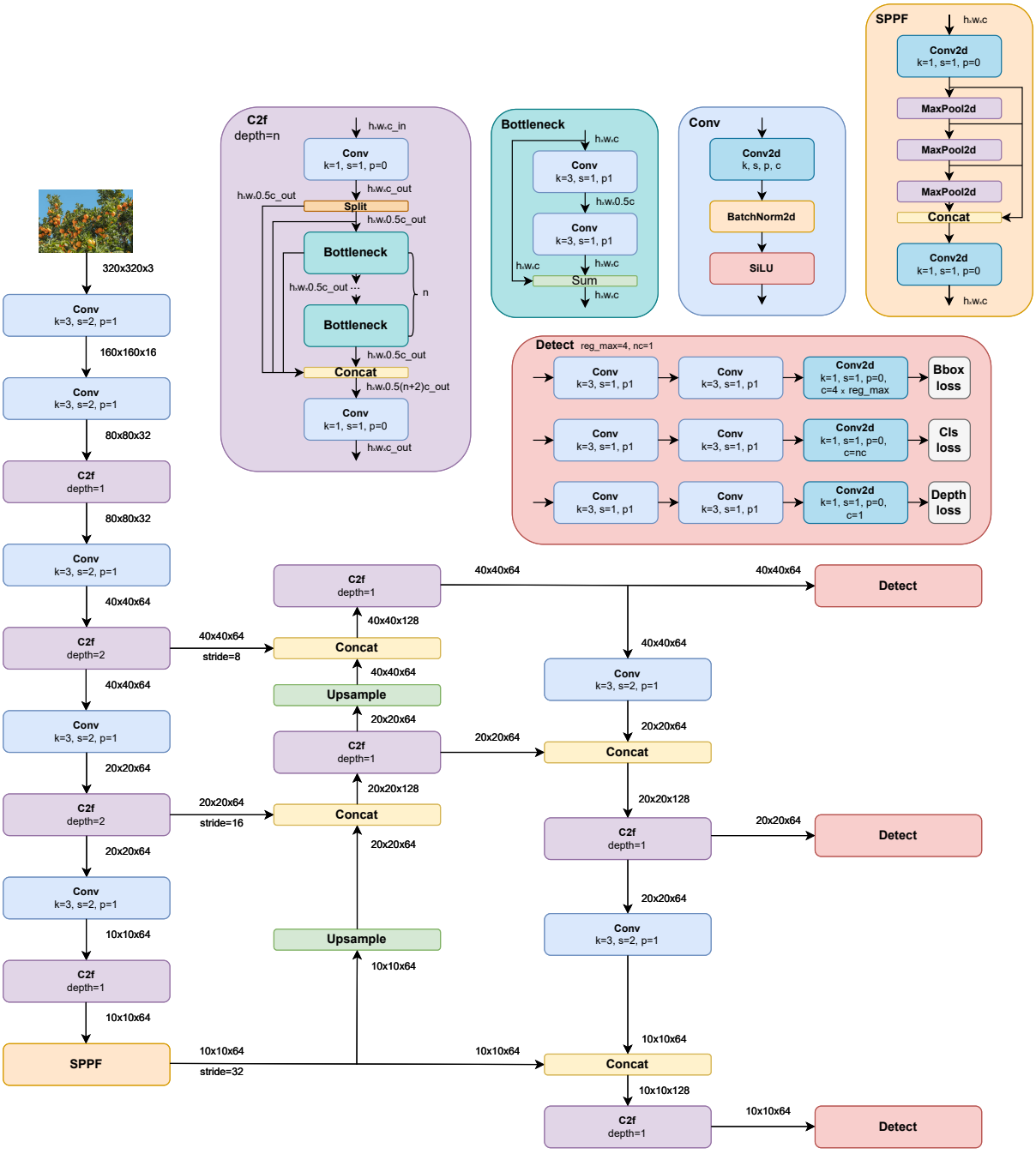


Figura 3.1: Arquitectura de la red neuronal convolucional de detección de objetos con estimación de profundidad DOD, inspirada en la arquitectura de YOLOv8 [54] (ver Anexos: Figura 1).

- **Bottleneck**: Bloque de cuello de botella basado en ResNet (2015) [76] y compuesto por dos módulos convolucionales de extracción de características con conexión residual para mitigar el problema del desvanecimiento del gradiente [77].

- **C2f** (*Conv-to-Features*): Bloque de cuello de botella parcial con dos módulos convolucionales entre etapas cruzadas de profundidad n. Inspirado en CSPNet (*Cross-Stage Partial Network*) [78], combina las características de alto nivel con la información contextual para mejorar la precisión de la red.

  En las redes neuronales convolucionales tradicionales, el flujo de información es secuencial a través de las capas profundas, descartando a los mapas de características en cada etapa. Sin embargo, las conexiones entre etapas cruzadas (*cross-stage*) se introducen para permitir que las características sean parcialmente preservadas, comunicadas y combinadas entre diferentes etapas de la red. Esto fomenta una mejor reutilización de características y permite que la red capture patrones y relaciones más complejas.

- **SPPF** (*Spatial Pyramid Pooling Fast*): Módulo rápido de agrupamiento piramidal espacial (SPP) [79] que acelera el cálculo al agrupar las características en un mapa de tamaño fijo. Mediante operaciones secuenciales de *pooling* por máximos, se busca separar las características más relevantes y aumentar significativamente el campo receptivo en el contexto sin disminuir la velocidad de la red.

- **Detect**: Bloque de detección compuesto por tres cabezales desacoplados para predecir respectivamente las dimensiones del cuadro delimitador, la clase perteneciente y la profundidad de un objeto detectado. Cada cabezal esta compuesto secuencialmente por dos módulos convolucionales y una convolución 2D.

  1. Cuadro delimitador: el número de mapas de salida c_{out} es igual a $4 * reg\_max$; 4 distancias l,r,t,b (izquierda, derecha, arriba y abajo) relativas al centro de la celda de predicción, multiplicadas por un número entero reg_max que limita el rango máximo de regresión de estas entre [0, $reg\_max - 1$].

     Es decir, el ancho y la altura de los cuadros están en el rango [0, $2 * (reg\_max - 1) * \ max\_stride$] debido a que $x_2y_2 - x_1y_1 = rb + lt$. El parámetro reg_max previene que los cuadros sean muy grandes o muy pequeños, asegurando cierta sensibilidad en las predicciones. La función objetivo DFL [73] introduce el parámetro reg_max. En el modelo DOD propuesto: $reg\_max = 4$ y $max\_stride = 32$.

  2. Clasificación: el número de mapas de salida c_{out} es igual al número de clases de clasificación nc. Para el problema de detección de frutos, se asume que el número de clases $nc = 1$, siendo la clase fruta como la única clase posible de detección. En el caso del conjunto de datos COCO [1], el número de clases es $nc = 80$.

  3. Profundidad: el mapa de salida c_{out} es igual al valor único (1) representativo de la profundidad relativa de un objeto respecto al contexto espacial en el que este se encuentra dentro de una escena.

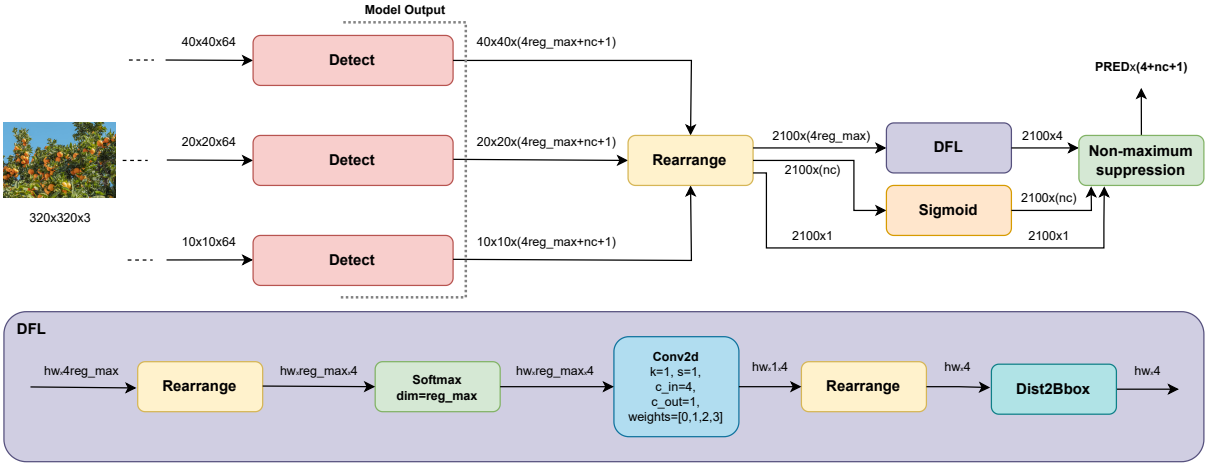


Figura 3.2: Proceso de inferencia del modelo de detección de objetos con estimación de profundidad DOD.

3.2.2. Proceso de Inferencia

Como bien describe la Figura 3.1, los cabezales de detección (bloque *Detect*) de la arquitectura propuesta generan, sin importar los objetos detectados, un número fijo de predicciones en cada inferencia. Tomando de ejemplo una imagen de entrada con dimensiones 320x320x3, las salidas concatenadas de los cabezales tendrían unas dimensiones de $40 \times 40 \times 20 \times 20 \times 10 \times 10 \times (4 \times reg\_max + nc + 1) = 2100 \times (4 \times reg\_max + nc + 1)$.

Este tensor, que contiene la distribución del cuadro delimitador, la clase perteneciente y la profundidad relativa de todas las predicciones posibles, se debe procesar para obtener un resultado filtrado que contenga únicamente las cuatro coordenadas del cuadro delimitador, la clase ganadora y la profundidad relativa de las detecciones válidas $pred$ en la imagen de entrada: $pred \times (4 + nc + 1)$.

El flujo del proceso de inferencia se describe en la Figura 3.2 y consiste básicamente en reorganizar los tensores de salida en los tres niveles de predicción para transformar las distribuciones del cuadro delimitador en cuatro coordenadas espaciales, obtener la puntuación de cada clase y pasar todas las predicciones por un algoritmo de Supresión No Máxima (NMS por sus siglas en inglés) para obtener las predicciones finales del modelo DOD.

A continuación, se describen los bloques utilizados en el proceso de inferencia:

- **DFL** (*Distributional Focal Loss*): Módulo integral de la Pérdida de Distribución Focal [73]. Como bien se ha explicado anteriormente, al integrar esta función objetiva en el diseño de la arquitectura, los cabezales de predicción del cuadro delimitador deben predecir una distribución de $4 \times reg\_max$ valores para cada distancia l,r,t,b.

  Para esto, se aplica la función Softmax a la distribución de valores de cada distancia para obtener un vector de probabilidades por cada distribución que se transformarán linealmente a las cuatro distancias mediante una convolución 2D sin gradiente con un número de filtros $c_{out} = 4$ y un tamaño de kernel $k = 1$, cuyos pesos han sido inicializados previamente como $w = [0, 1, 2, 3]$.

Una vez obtenidas las distancias l,r,t,b por cada predicción, se suman a las coordenadas del punto central de cada celda de predicción (ver Anexos: Figura 1) y se escalan de acuerdo al *stride* del nivel donde se predijeron, ya sea 8, 16 o 32, obteniendo finalmente un cuadro delimitador con coordenadas x, y respecto a las dimensiones de la imagen de entrada para cada predicción.

- **NMS** (*Non-Maximum Suppression*): La Supresión No Máxima (NMS) es una técnica de post-procesamiento empleada en algoritmos de detección de objetos para reducir el número de cuadros delimitadores superpuestos de acuerdo al Índice de Jaccard (también conocido como IoU por sus siglas en inglés de *intersection-over-union*) descrito en la Ecuación 3.2, que mide el grado de similitud entre dos cuadros.

$$IoU(A, B) = J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.2}$$

Al generar múltiples cuadros delimitadores con distintas puntuaciones de confianza, NMS identifica y conserva únicamente los cuadros más precisos, filtrando aquellos redundantes e irrelevantes. El Algoritmo 3.1 describe el procedimiento.

```
Requiere: Conjunto de cajas delimitadoras predichas B, puntajes de confianza S,
    umbral de IoU τ, umbral de confianza T
Asegura: Conjunto de cajas delimitadoras filtradas F
F ← ∅
Filtrar las cajas: B ← {b ∈ B | S(b) ≥ T}
Ordenar las cajas B por sus puntajes de confianza en orden descendente
mientras B ≠ ∅ hacer
    Seleccionar la caja b con el puntaje de confianza mas alto
    Agregar b al conjunto de cajas finales F: F ← F ∪ {b}
    Eliminar b del conjunto de cajas B: B ← B − {b}
    para todas las cajas restantes r en B hacer
        Calcular el IoU entre b y r: iou ← IoU(b, r)
        si iou ≥ τ entonces
            Eliminar r del conjunto de cajas B: B ← B − {r}
        fin si
    fin para
fin mientras
```

Algoritmo 3.1: Supresión No Máxima o NMS. Adaptado de [53]

3.3. Función de Pérdida

Los pesos de la red neuronal convolucional de la Figura 3.1 se ajustan minimizando la fórmula matemática descrita en la Ecuación 3.3, que es la función de pérdida generalizada que incorpora los pesos de pérdida individuales y un término de regularización con un decaimiento de peso ϕ, mediante la Ecuación 3.4 como regla de actualización de pesos con un factor de aprendizaje η y con un término de velocidad de actualización con momento β, descrito en la Ecuación 3.5.

$$\mathcal{L}(\theta) = \frac{\lambda_{box}}{N_{pos}}\mathcal{L}_{box}(\theta) + \frac{\lambda_{cls}}{N_{pos}}\mathcal{L}_{cls}(\theta) + \frac{\lambda_{dfl}}{N_{pos}}\mathcal{L}_{dfl}(\theta) + \frac{\lambda_{depth}}{N_{pos}}\mathcal{L}_{depth}(\theta) + \phi \left\|\theta\right\|_2^2 \tag{3.3}$$

$$\theta^t = \theta^{t-1} - \eta V^t \tag{3.4}$$

$$V^t = \beta V^{t-1} + \nabla_\theta \mathcal{L}(\theta^{t-1}) \tag{3.5}$$

Descrita matemáticamente por Reis *et al.* [80] para YOLOv8 [54], la función de pérdida extendida y adaptada para la estimación de profundidad se describe en la Ecuación 3.6.

El primer término es la pérdida de la Intersección Sobre la Unión Completa (CIoU por sus siglas en inglés) propuesta por Zheng *et al.* [81], e incorpora una mejora sobre el la Pérdida del Índice de Jaccard tradicional al tomar en cuenta tres factores geométricos importantes: el área superpuesta, la distancia entre los puntos centrales y la tasa de aspecto entre los cuadros predichos y de referencia, penalizando severamente las predicciones inexactas.

El segundo término es la Entropía Cruzada Binaria (BCE por sus siglas en inglés) como la pérdida de clasificación, que en el caso multi-etiqueta, permite que cada celda pueda predecir una o más clases, forzando al modelo a aprender la distribución de cada clase sin la influencia de las demás.

El tercer término es la Pérdida de Distribución Focal (DFL por sus siglas en inglés) propuesta por Li *et al.* [73], que obliga a la red a centrarse rápidamente en los valores cercanos al cuadro de referencia al aumentar explícitamente las probabilidades en la distribución $4 \times reg\_max$ predicha respecto a los valores más cercanos (a la izquierda y derecha) del cuadro de referencia.

El cuarto término es la Error Cuadrático Medio (MSE por sus siglas en inglés) o Norma L_2 como la pérdida para la estimación de profundidad entre los valores de profundidad representativos predichos y los valores de profundidad de referencia.

$$\begin{aligned}
\mathcal{L} = &\frac{\lambda_{box}}{N_{pos}} \sum_{x,y} \mathbb{1}_{c^*_{x,y}} [1 - q_{x,y} + \frac{\left\| b_{x,y} - \hat{b}_{x,y} \right\|_2^2}{\rho^2} + \alpha_{x,y} v_{x,y}] \\
&+ \frac{\lambda_{cls}}{N_{pos}} \sum_{x,y} \sum_{c \in clases} y_c log(\hat{y}_c) + (1 - y_c) log(1 - \hat{y}_c)) \\
&+ \frac{\lambda_{dfl}}{N_{pos}} \sum_{x,y} \mathbb{1}_{c^*_{x,y}} \left[-(d_{(x,y)+1} - d_{x,y}) log(\hat{d}_{x,y}) + (d_{x,y} - d_{(x,y)-1}) log(\hat{d}_{(x,y)+1)}) \right] \\
&+ \frac{\lambda_{depth}}{N_{pos}} \sum_{x,y} \mathbb{1}_{c^*_{x,y}} (z_{x,y} - \hat{z}_{x,y})^2
\end{aligned} \tag{3.6}$$

$$q_{x,y} = IoU(x,y) = \frac{\left| \hat{\beta}_{x,y} \cap \beta_{x,y} \right|}{\left| \hat{\beta}_{x,y} \cup \beta_{x,y} \right|} \tag{3.7}$$

$$v_{x,y} = \frac{4}{\pi^2} \left(\arctan\left(\frac{w_{x,y}}{h_{x,y}} \right) - \arctan\left(\frac{\hat{w}_{x,y}}{\hat{h}_{x,y}} \right) \right)^2 \tag{3.8}$$

$$\alpha_{x,y} = \frac{v}{1 - q_{x,y}} \tag{3.9}$$

$$\hat{y}_c = \sigma(\cdot) \tag{3.10}$$

$$\hat{d}_{x,y} = softmax(\cdot) \tag{3.11}$$

Donde:

- N_{pos} es el número total de celdas que contienen un objeto (predicciones positivas).
- $\mathbb{1}_{c^*_{x,y}}$ es la función indicadora para las celdas con objetos detectados.
- $q_{x,y}$ es la IoU entre un cuadro delimitador predicho y el cuadro de referencia de un objeto de referencia (Ecuación 3.7).
- $\beta_{x,y}$ es una tupla ($x_{coord}, y_{coord}, width, height$) que representa el cuadro delimitador de referencia.
- $\hat{\beta}_{x,y}$ es el cuadro predicho por una celda respectiva.
- $b_{x,y}$ es una tupla (x_{coord}, y_{coord}) que representa el punto central de un cuadro delimitador de referencia.
- $\hat{b}_{x,y}$ es el punto central del cuadro predicho por una celda respectiva.
- ρ es la distancia diagonal del mínimo cuadro delimitador que encierra al mismo tiempo el cuadro predicho y el cuadro de referencia.
- $v_{x,y}$ mide la consistencia en la tasa de aspecto entre cuadros a partir de su ancho y alto respectivamente ($w_{x,y}, h_{x,y}$),($\hat{w}_{x,y}, \hat{h}_{x,y}$). (Ecuación 3.8).
- $\alpha_{x,y}$ es una compensación positiva donde el factor de área de superposición tiene mayor prioridad para la regresión, especialmente para los casos que no se superponen (Ecuación 3.9).
- y_c es la etiqueta de referencia para la clase c para cada celda individual x, y, sin importar si un objeto está presente.
- $\hat{y}_c$ es la probabilidad predicha para la clase c individual x, y, sin importar si un objeto está presente (Ecuación 3.10).
- $d_{(x,y)+/-1}$ son tuplas (l, r, t, b) con los valores más cercanos a la izquierda y derecha de un cuadro de referencia cuya tupla ($x_{coord}, y_{coord}, width, height$) ha sido transformada a una distancia respecto al centro de la celda de predicción positiva.
- $\hat{d}_{x,y}$ son las probabilidades de la distribución $4 \times reg\_max$ predichas por una celda que contiene objeto.
- $z_{x,y}$ es el valor representativo de la profundidad relativa a la escena de fondo del objeto en el cuadro de referencia.
- $\hat{z}_{x,y}$ es el valor representativo de la profundidad relativa a la escena de fondo del objeto detectado en la celda de predicción.

3.4. Métricas de Evaluación

3.4.1. Detección: Precisión Promedio Media

La Precisión Promedio Media (mAP de *mean Average Precision*, Ecuación 3.12) mide el desempeño absoluto de un modelo de detección de objetos al medir la Precisión Promedio (AP) de todas las clases de clasificación en un umbral de Intersección sobre la Unión (IoU, Ecuación 3.2) para seleccionar las predicciones positivas. Se describe en la Ecuación 3.12.

$$mAP = \frac{1}{clases} \sum_{c \in clases} AP_c|_{IoU=thr} \tag{3.12}$$

Las métricas mAP 50 y mAp 50-95 son las métricas aceptadas para comparar diferentes modelos de detección de objetos. Miden la Precisión Promedio en la detección y localización de objetos en diferentes niveles de dificultad.

- **mAP 50**: Es la métrica oficial de PASCAL VOC [62], Ecuación 3.13. Calcula la Precisión Promedio Media para un umbral de Intersección sobre la Unión $IoU = 0.5$. Aquellos objetos detectados cuyo cuadro delimitador tenga una $IoU > 50\,\%$ con un cuadro de referencia, serán considerados como predicciones positivas.

$$mAP50 = \frac{1}{clases} \sum_{c \in clases} AP_c|_{IoU=0.5} \tag{3.13}$$

- **mAP 50-95**: Es la métrica oficial de COCO [1], Ecuación 3.14. Es el promedio de las Precisiones Promedio Medias calculadas utilizando un rango de umbrales de Intersección sobre la Unión $0.5 \leq IoU \leq 0.95$ en incrementos de 0.05. Esta métrica proporciona una visión más completa del rendimiento del modelo en diferentes niveles de superposición entre las cajas delimitadoras.

$$mAP50 - 95 = \frac{1}{clases \cdot thr} \sum_{n \in thr} \sum_{c \in clases} AP_c|_{IoU=n} \tag{3.14}$$

$$thr = [0.5, 0.55, 0.60, ..., 0.95] \tag{3.15}$$

3.4.1.1. Precisión Promedio

La Precisión Promedio (AP) (Ecuación 3.16) mide el desempeño de un sistema de clasificación al calcular el área bajo la curva (AuC) de la curva Precisión-Recuperación para una clase dada. Para esto, se cuenta el número de falsos positivos (FP), verdaderos positivos (TP) y falsos negativos (FN) predichos por el modelo para un umbral de confidencia para las probabilidades $\hat{y}_c$.

$$AP_c = AuC(PR_{curve}|c) \tag{3.16}$$

La precisión (Ecuación 3.17) mide la exactitud de las predicciones positivas. La recuperación (*Recall*, Ecuación 3.18) mide la proporción de casos positivos identificados correctamente. Ambas se miden en el mismo rango de umbrales de confidencia con el fin de obtener la curva de Precisión-Recuperación, cuya área bajo la curva (AuC,

Ecuación 3.19) mide el equilibrio entre el rango de recuperación y el número de falsos positivos.

$$Precision_{(c|p=thr)} = \frac{TP_{(c|p=thr)}}{TP_{(c|p=thr)} + FP_{(c|p=thr)}} \tag{3.17}$$

$$Recall_{(c|p=thr)} = \frac{TP_{(c|p=thr)}}{TP_{(c|p=thr)} + FN_{(c|p=thr)}} \tag{3.18}$$

$$AuC(PR_{curve}) = \int_0^1 Precision(Recall)\ dRecall \tag{3.19}$$

3.4.2. Profundidad: Error Cuadrático Medio

El Error Cuadrático Medio (MSE, Ecuación 3.20 mide el desempeño de un modelo de estimación de profundidad al medir una distancia Euclidiana entre los valores representativos de profundidad estimados y los valores de referencia. Se mide únicamente con las predicciones consideradas como positivas.

$$MSE = \frac{\lambda_{depth}}{N_{pos}} \sum_{x,y} \mathbb{1}_{c^*_{x,y}} (z_{x,y} - \hat{z}_{x,y})^2 \tag{3.20}$$

3.5. Asignación de Etiquetas

La arquitectura propuesta adopta la asignación de etiquetas TOOD (*Task-aligned One-stage Object Detection*) propuesta por Feng *et al.* [50]. En la detección, a menudo existen múltiples tareas relacionadas como la detección de objetos pequeños, la detección de objetos en poses inusuales o la detección de objetos parcialmente ocultos. La asignación de etiquetas es una estrategia dinámica para abordar la alineación de tareas en los modelos de detección de objetos.

El enfoque TOOD [50] selecciona las muestras positivas (predicciones que tienen un impacto en el aprendizaje) a partir de puntuaciones ponderadas de clasificación y regresión, como se describe en la Ecuación 3.21.

$$t = s^\alpha \times u^\beta \tag{3.21}$$

Donde $s = \hat{y}_c$ es la puntuación predicha a una clase de referencia, y $u = CIoU$ es la CIoU [81] (primer término en la Ecuación 3.6) entre el cuadro delimitador predicho y el cuadro de referencia:

1. Para cada objeto de referencia (*ground truth*), la asignación dinámica calcula la métrica de alineación t.
2. Para cada objeto de referencia, las *top*-k muestras con la alineación más grande se seleccionan como muestras positivas.

Capítulo 4

Entrenamiento

4.1. Conjuntos de Datos

Los conjuntos de datos (ver Estado del Arte: Conjuntos de Datos 2.3) utilizados para el entrenamiento del modelo son los siguientes:

- **COCO** (*Microsoft Common Objects in Context*): Se utiliza el conjunto de datos COCO [1] para entrenar una versión temprana del modelo DOD comparable con los modelos de detección del estado del arte actual para evaluar la arquitectura adaptada.
- **MinneApple**: Se utiliza el conjunto de datos MinneApple [2, 3, 4] para entrenar una versión posterior del modelo DOD comparable con los modelos implementados para la detección de frutos en este conjunto de datos.
- **Apples**: Se utiliza los conjuntos de datos Apples [63] y MinneApple [2, 3, 4] para entrenar una versión final del modelo DOD con mayor campo receptivo en cuanto a los tamaños relativos de los frutos en las imágenes.

4.1.1. Adaptación para Estimación de Profundidad

Los conjuntos de datos tienen únicamente las etiquetas para el problema de detección de objetos. La profundidad representativa de cada objeto de referencia se predice haciendo uso del modelo *visual transformer* MiDaS propuesto por Ranft *et al.* [45, 46]. Este proceso se describe en el Algoritmo 4.1.

```
1 Requiere: Conjunto de datos de deteccion de objetos D, modelo entrenado para
      estimacion de profundidad monoscopica h
2 Asegura: Conjunto de datos D' con la profundidad representativa para cada objeto de
      referencia
3 para cada conjunto de datos D hacer
4     para cada imagen hacer
5         Predecir el mapa de profundidad z ← h(imagen)
6         para cada objeto hacer
7             Extraer el cuadro del objeto en el mapa de profundidad z_obj ← z ∩ objeto
8             Calcular el valor representativo del cuadro de profundidad
                  z'_obj ← 0.5(mean(z_obj) + max(z_obj))
9             Almacenar la etiqueta obj_depth ← z'_obj
```

Algoritmo 4.1: Extracción de profundidad.

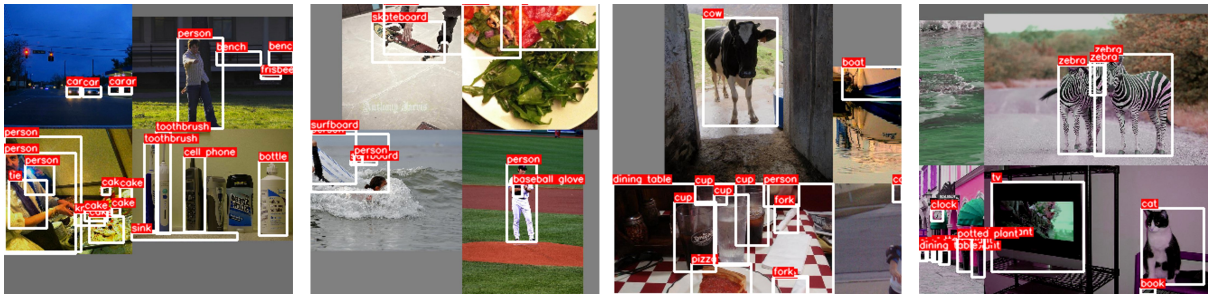


Figura 4.1: *Mosaic Augmentation*. Imágenes tomadas del conjunto de datos COCO [1].

Figura 4.2: MixUp *Augmentation*. Imágenes tomadas del conjunto de datos COCO [1].

4.1.2. Aumento de Datos

Las estrategias de *Data Augmentation* utilizadas en el entrenamiento se describen a continuación:

- **Mosaic**: La técnica de Mosaico se introduce en el entrenamiento del modelo YOLOv8 [54] y consiste en agregar cuatro imágenes en una sola imagen (Figura 4.1). Esto alivia la carga en cada *batch* de entrenamiento y hace que el modelo sea más robusto ante el contexto espacial de los objetos.

- **MixUp**: La técnica MixUp se introduce por Zhang *et al.* [82] y es una estrategia para combinar dos muestras de entrenamiento junto con sus respectivas etiquetas para crear un nuevo ejemplo sintético de acuerdo a un parámetro de mezcla α (Figura 4.2). Esta técnica puede regularizar el sobreajuste en el entrenamiento, además de mejorar la generalización ante variaciones en los datos.

$$img = \alpha \cdot img_t + (1 - \alpha) \cdot img_{(t+1)} \tag{4.1}$$

- **ColorJitter**: Con una probabilidad del 50 %, se aplican transformaciones leves en los colores de las imágenes para mejorar la capacidad del modelo para generalizar a diferentes condiciones de iluminación y color. Se utiliza un valor aleatorio en el rango [-0.3, 0.3] y [-0.2,0.2] al brillo, contraste y saturación de la imagen. Implementada en la librería de código abierto Albumentations [83].

- **HorizontalFlip**: Con probabilidad del 50 %, se invierte horizontalmente la imagen junto sus etiquetas. Implementada en la librería de código abierto Albumentations [83].

4.2. Protocolos de Prueba

A continuación, mediante protocolos de prueba, se describen los pasos realizados para ejecutar el entrenamiento y la validación del modelo DOD sobre los distintos conjuntos de datos utilizados.

Los protocolos de prueba han sido implementados en el lenguaje interpretado y secuencial orientado a objetos Python [84] y utilizando principalmente la librería de código abierto para el cálculo de tensores con aceleración de GPU PyTorch [85].

4.2.1. Fases del Entrenamiento

La Figura 4.3 describe el flujo secuencial entre las fases de entrenamiento para los distintos conjuntos de datos utilizados en el marco de evaluación del modelo DOD.

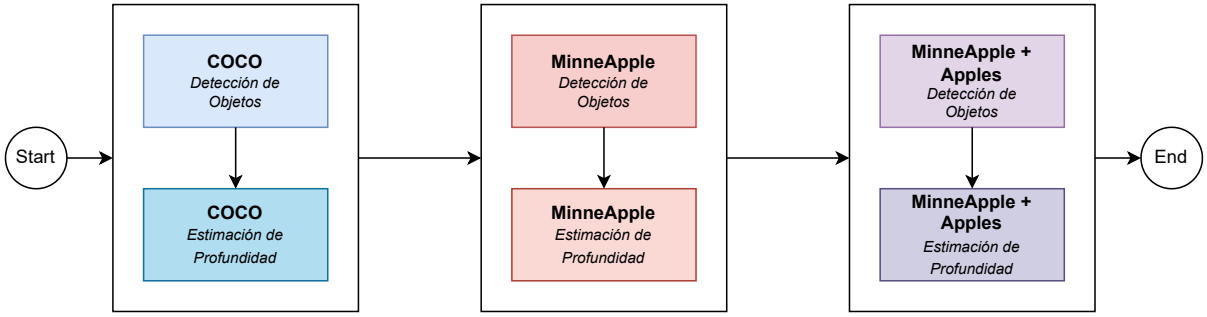


Figura 4.3: Diagrama de flujo con las fases de entrenamiento del modelo de detección con estimación de profundidad DOD.

4.2.1.1. Objetivo

Establecer un proceso lógico de entrenamiento y validación al entrenar de forma separada la capacidad de detectar objetos y la capacidad de estimar la profundidad en los diferentes conjuntos de datos utilizados.

4.2.1.2. Metodología

Para cada conjunto de datos: COCO [1], MinneApple [2, 3, 4] y Apples [63]:

1. Inicializar un modelo DOD con pesos aleatorios.
    - Nota: en el caso de la detección de frutos, se ha experimentado utilizando los pesos de la mejor versión obtenida en COCO [1] como punto de partida, pero esto genera peores resultados.
2. Entrenar el modelo únicamente para la tarea de detección de objetos. Se excluye el gradiente en los cabezales de estimación de profundidad y se excluye el cuarto término en la Función de Pérdida 3.6.
3. Entrenar la mejor versión obtenida en el paso anterior para la tarea de estimación de profundidad. Se usan todos los términos en la Función de Pérdida 3.6 y se aplica el gradiente en toda la red.

4. Copiar los pesos de los cabezales de estimación de profundidad en la mejor versión obtenida en el paso anterior en los cabezales de la mejor versión obtenida en el paso 2.

4.2.2. Estrategia de Entrenamiento

La Figura 4.4 describe el flujo en la estrategia de entrenamiento utilizada para ajustar los pesos del modelo DOD a una tarea específica, ya sea la detección de objetos o a la estimación de profundidad. Este protocolo se aplica individualmente para cada conjunto de datos.

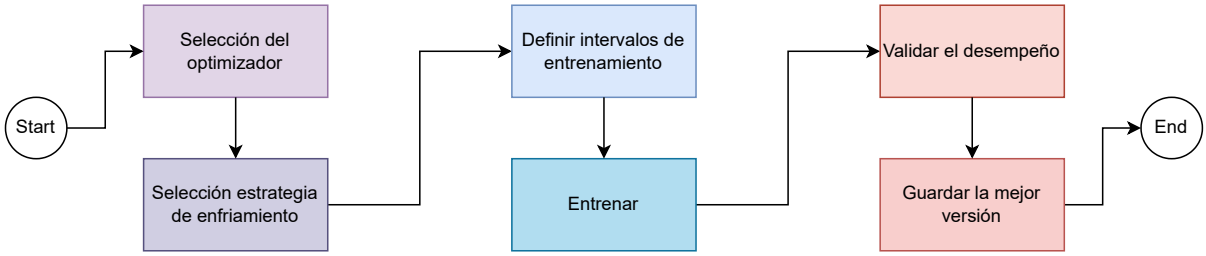


Figura 4.4: Diagrama de flujo de la estrategia de entrenamiento.

4.2.2.1. Objetivo

Establecer un conjunto de pasos sistemáticos para entrenar y evaluar el modelo DOD.

4.2.2.2. Metodología

1. Establecer un tamaño de batch de 32.
2. Seleccionar un optimizador para aplicar el descenso del gradiente, argumentando su elección en función de la convergencia.
   - El modelo YOLOv8 [54] fue entrenado con el optimizador Estocástico del Descenso del Gradiente (SGD por sus siglas en inglés) ya que a pesar de su tiempo de convergencia más lento respecto al optimizador Adam [86], generaliza mucho mejor en el conjunto de datos COCO [1] (2017) de 80 etiquetas [87].

     Keskar *et al.* [88] sugieren que empezar el entrenamiento con el optimizador Adam [86] permite aprovechar la velocidad inicial de convergencia, y en un número cercano de épocas, cambiar al optimizador SGD para aprovechar la mejor generalización [88].

     Sin embargo, Isa *et al.* [89] sugieren que el optimizador Adam [86] puede ser superior o igual que el optimizador SGD con una buen ajuste de hiperparámetros y aprovechando la variante AMSGrad propuesta por Sashank *et al.* [90] para ajustar mejor la velocidad del gradiente [89].
3. Elegir una función dinámica de enfriamiento para el factor de aprendizaje.
4. Definir los intervalos de entrenamiento asignado un número de épocas para:
   - Calentar previamente el modelo DOD con un factor de aprendizaje pequeño y ascendente.

- Aplicar la técnica de aumento de datos MixUp.
- En el caso de entrenar con el conjunto de datos COCO [54], aplicar la técnica de aumento de datos Mosaico.
- Finalizar el entrenamiento sin aplicar aumento de datos MixUp ni Mosaico. (La técnica de aumento de datos ColorJitter y HorizontalFlip se aplican en todas las épocas de entrenamiento)

5. Utilizar un promedio móvil exponencial (EMA por sus siglas en inglés) de los parámetros entrenados para mejorar la generalización.
   - Cuando se entrena un modelo de aprendizaje, es beneficioso mantener un promedio móvil de los parámetros entrenados, ya que estos pueden filtrar los picos ruidosos en las distribuciones de los datos estimadas por los pesos *online*.
   - Lo que se busca, es actualizar la versión EMA en cada paso en el bucle de entrenamiento usando los pesos de la versión *online* junto con la acumulación exponencial media de sus versiones anteriores. Los parámetros promediados son entonces una versión con mejor generalización respecto a los valores atípicos en la distribución matemática de las muestras [91].
6. Validar el desempeño del modelo DOD en cada época, tanto en la versión *online* como en el modelo EMA. Los resultados se comparan con los resultados de otros modelos (ya existentes o implementados) en el mismo conjunto de datos.
7. Guardar la versión del modelo DOD con los mejores resultados obtenidos durante el entrenamiento y la validación.

4.2.3. Adaptación a Sistemas Embebidos: Cuantización

La Figura 4.5 describe el flujo del proceso de cuantización estática de algunas operaciones internas del modelo DOD, así como sus pesos, de precisión de punto flotante de 32-bits a precisión entera con signo de 8-bits.

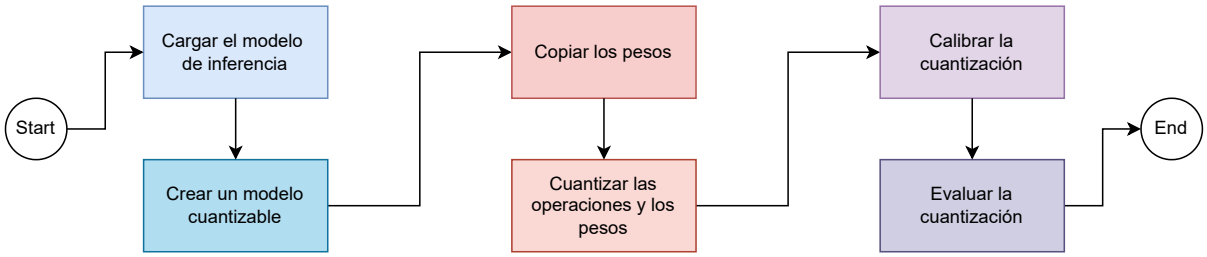


Figura 4.5: Diagrama de flujo del proceso de cuantización del modelo entrenado.

4.2.3.1. Objetivo

Optimizar el espacio de almacenamiento que requieren las operaciones y pesos del modelo de detección de frutos con estimación de profundidad DOD para poder operarlo de forma óptima en sistemas microcontrolados o embebidos, de bajo costo y uso específico como la tarjeta de desarrollo Raspberry Pi [92, 93].

4.2.3.2. Metodología

1. Inicializar el modelo DOD con los pesos de la mejor versión entrenada.
2. Crear una versión cuantizable del modelo DOD al indicar las operaciones a cuantizar mediante los métodos de colocación Quant/DeQuant de Pytorch [85].
   - La librería soporta únicamente la cuantización de las siguientes operaciones: convolución 2D, normalización del batch, capa lineal y activación Lineal Rectificadora ReLU.
   - La arquitectura propuesta en este trabajo (Figura 3.1) utiliza la función de activación SiLU por su mejor desempeño en el estado del arte respecto a ReLU. Entonces, las únicas operaciones cuantizables en el modelo propuesto son las convoluciones 2D y sus normalizaciones de batch.
   - Nota: se experimenta reentrenando el modelo usando la función ReLU para comparar si existen mejoras en la velocidad en la inferencia de los modelos cuantizados, pero las diferencias de ejecución son minúsculas a pesar de la caída en 2 % de mAP 50-95.
3. Copiar los pesos de todas las operaciones del modelo DOD en precisión punto flotante de 32-bits al modelo cuantizable.
4. Aplicar el método de cuantización QNNPACK (*Quantized Neural Networks PACKage*) [94] integrado en Pytorch [85] para convertir de precisión de punto flotante de 32-bits a precisión entera con signo de 8-bits.
5. Calibrar el modelo cuantizado utilizando un número pequeño de pasos de inferencia sobre los conjuntos de datos MinneApple [2, 3, 4] y Apples [63]. Esto para identificar los rangos de operación de las operaciones cuantizadas y poder asignar el tipo de variable más óptima para almacenar cada peso y operación.
6. Comparar el modelo DOD cuantizado contra el modelo DOD en precisión de punto flotante en:
   - La precisión promedia media (mAP) de detección y el error cuadrático medio de la estimación de profundidad en el conjunto de datos MinneApple [2, 3, 4].
   - La velocidad media de inferencia para un número de veces repetidas la partición de validación del conjunto de datos MinneApple [2, 3, 4].

Capítulo 5

Resultados

5.1. COCO

5.1.1. Experimentos Previos

Para seleccionar los mejores hiper-parámetros de entrenamiento, se realizaron previamente una gran variedad de experimentos. A destacar los siguientes:

1. Para la tarea de detección de objetos: entrenamiento con el optimizador SGD, 300 épocas en total, sin calentamiento, de las cuales 290 utilizarán la técnica de aumento Mosaico y MixUp. El factor de enfriamiento del factor de aprendizaje es lineal entre 0.001 y 0.0001. Este experimento busca replicar la estrategia de entrenamiento de YOLOv8 [54]. En Anexos: Figura 2 se describen las curvas de aprendizaje obtenidas.
2. Para la tarea de detección de objetos: entrenamiento con el optimizador Adam, 400 épocas en total sin calentamiento, las primeras 50 utilizan la técnica de aumento MixUp junto Mosaico, las siguientes 330 utilizan Mosaico y el factor de enfriamiento del factor de aprendizaje tiene forma lineal entre 0.001 y 0.0001. En Anexos: Figura 3 se describen las curvas de aprendizaje obtenidas.

5.1.2. Entrenamiento

La Figura 5.1 ilustra la curva de aprendizaje de cada término de la Función de Pérdida (Ecuación 3.6) en el entrenamiento del modelo DOD para la detección de objetos y la estimación de profundidad en el conjunto de datos COCO [1]:

- La Pérdida de la Intersección Sobre la Unión Completa [81].
- La Entropía Cruzada Binaria.
- La Pérdida de Distribución Focal [73].
- El Error Cuadrático Medio de la estimación de profundidad.

Los hiper-parámetros utilizados en la estrategia de entrenamiento son los siguientes:

- Optimizador Adam [86] con su variante AMSGrad [90].
- 3 épocas de calentamiento con un factor de aprendizaje lineal entre 0.0001 y 0.001.

- Factor de enfriamiento coseno entre 0.001 y 0.0001.
- Para la detección de objetos: 140 épocas de entrenamiento. MixUp: las primeras 50 épocas. Mosaico: las primeras 120 épocas.
- Para la estimación de profundidad: 50 épocas de entrenamiento. MixUp: ninguna época. Mosaico: las primeras 25 épocas.

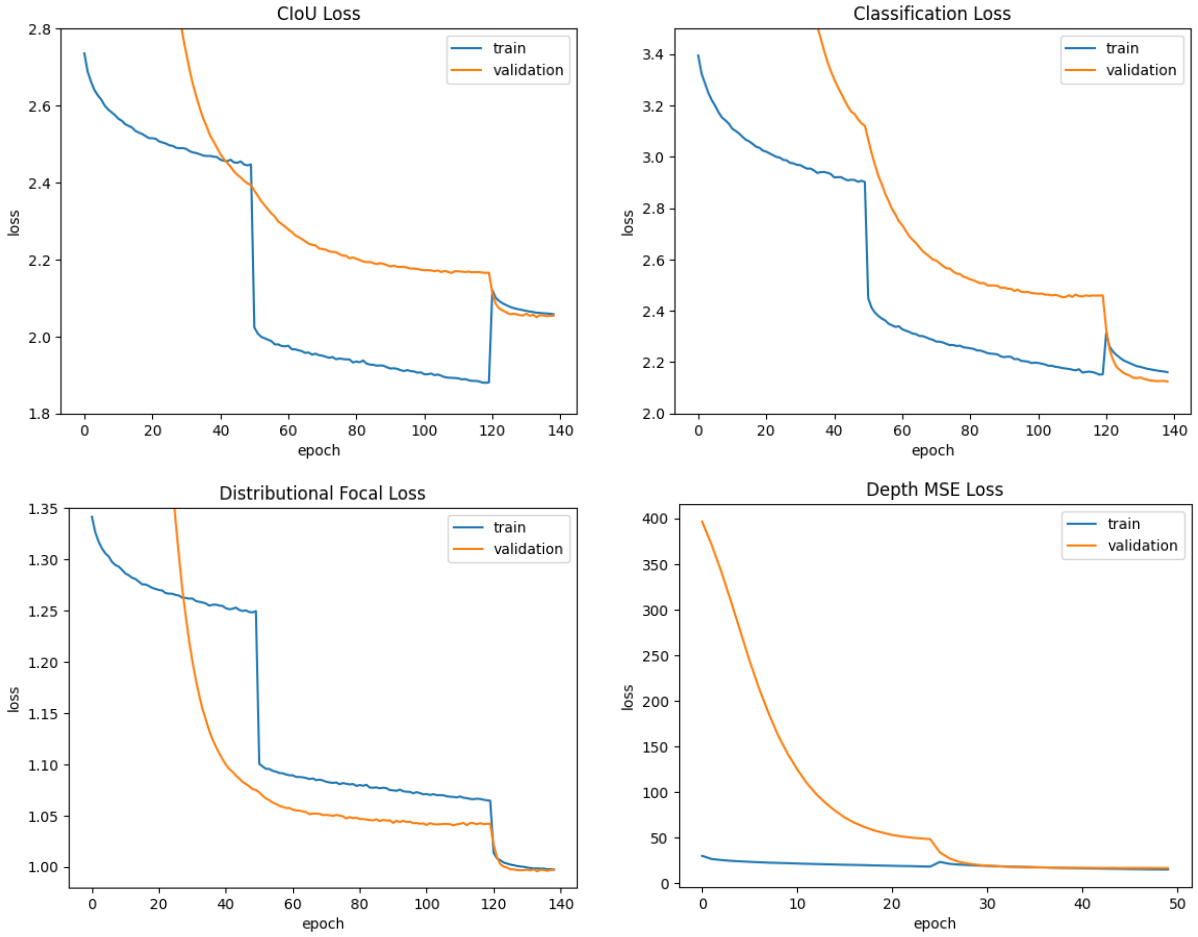


Figura 5.1: Curva de aprendizaje para cada factor de la Función de Pérdida (Ecuación 3.6) en el entrenamiento del modelo DOD sobre el conjunto de datos COCO [1]. Duración del entrenamiento: 55,440 segundos.

Modelo	P (%)	R (%)	mAP 50 (%)	mAP 50-95 (%)	MSE depth	Velocidad CPU (fps)	Velocidad GPU (fps)
YOLOv8n	59.3	39.7	42.8	29.3	-	47.3	83.8
DOD	41.3	25.9	24.3	12.3	59.3	57.1	84.7

Cuadro 5.1: Métricas de evaluación obtenidas por los modelos entrenados y evaluados en COCO [1].

Modelo	Número de parámetros (M)	Espacio de almacenamiento (MB)
YOLOv8n	3.15	6.23
DOD	1.06	4.24

Cuadro 5.2: Tamaños de las arquitecturas evaluadas.

5.1.3. Validación

Las Tablas 5.1 y 5.2 enuncian las métricas de comparación obtenidas a partir de la evaluación bajo las mismas condiciones del modelo DOD entrenado y el modelo preentrenado YOLOv8n [70, 54] en el conjunto de datos COCO [1]. Específicamente, se enuncian las siguientes métricas:

- Precisión para el mejor umbral de confidencia p.
- Recuperación para el mejor umbral de confidencia p.
- Precisión promedio media para $IoU = 0.5$ (mAP 50) e $IoU \in [0.5, 0.95; 0.05]$ (mAP 50 95).
- Error cuadrático medio de la estimación de profundidad.
- Tiempo medio de inferencia para cuatro veces las imágenes de la partición de validación con batch unitario.
  - CPU: AMD Ryzen 7 5800H 3.20 GHz.
  - GPU: NVIDIA GeForce RTX 3070 Laptop GPU.
- Número de parámetros.
- Tamaño en memoria de almacenamiento.

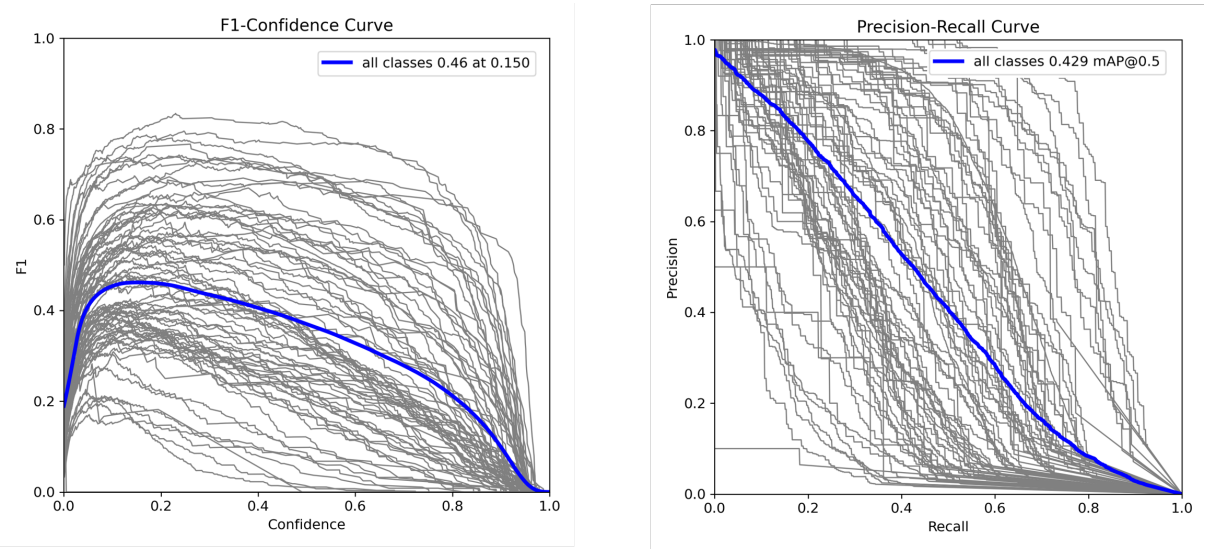
La Figura 5.2 ilustra las curvas F1-Confidencia y Precisión-Recuperación para cada detector evaluado en COCO [1].

- F1-Confidencia: muestra el valor armónico medio de la precisión y la recuperación para diferentes umbrales de confidencia.
- Precisión-Recuperación: muestra la compensación entre precisión y recuperación para diferentes umbrales.

En Anexos: Figuras 4-8 se ilustran las curvas de Precisión-Confidencia y Recuperación-Confidencia, junto con las matrices de confusión normalizadas y sin normalizar de cada modelo.

Algunos resultados de inferencia sobre las imágenes de validación se pueden observar en las Figuras 5.3-5.4.

(a) Modelo YOLOv8n



(b) Modelo DOD

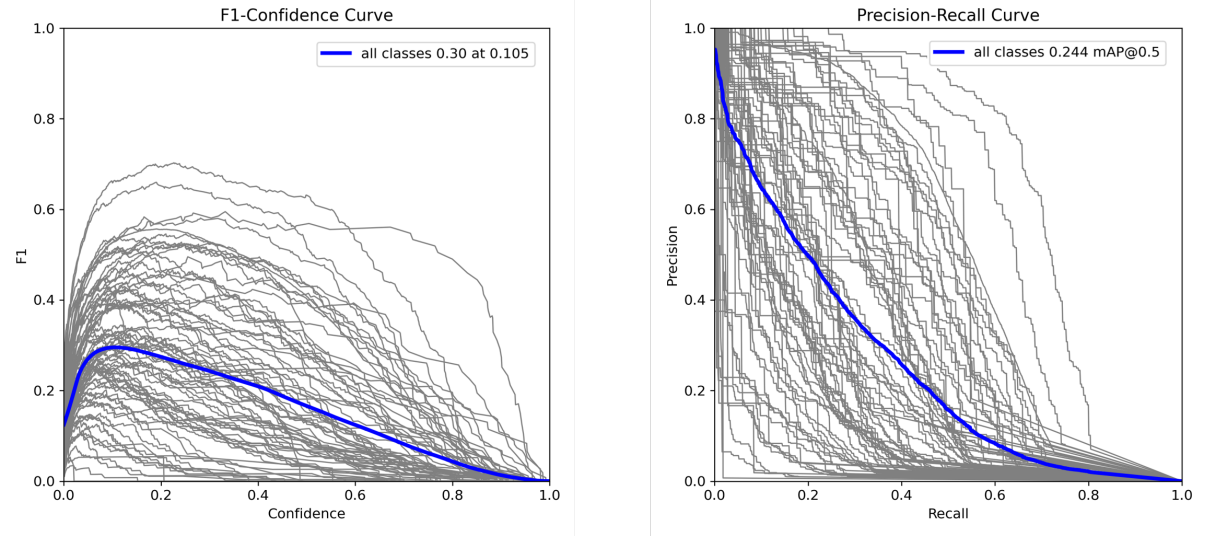


Figura 5.2: Curvas F1-Confidencia y Precisión-Recuperación de los modelos de detección de objetos YOLOv8n y DOD evaluados en COCO [1].

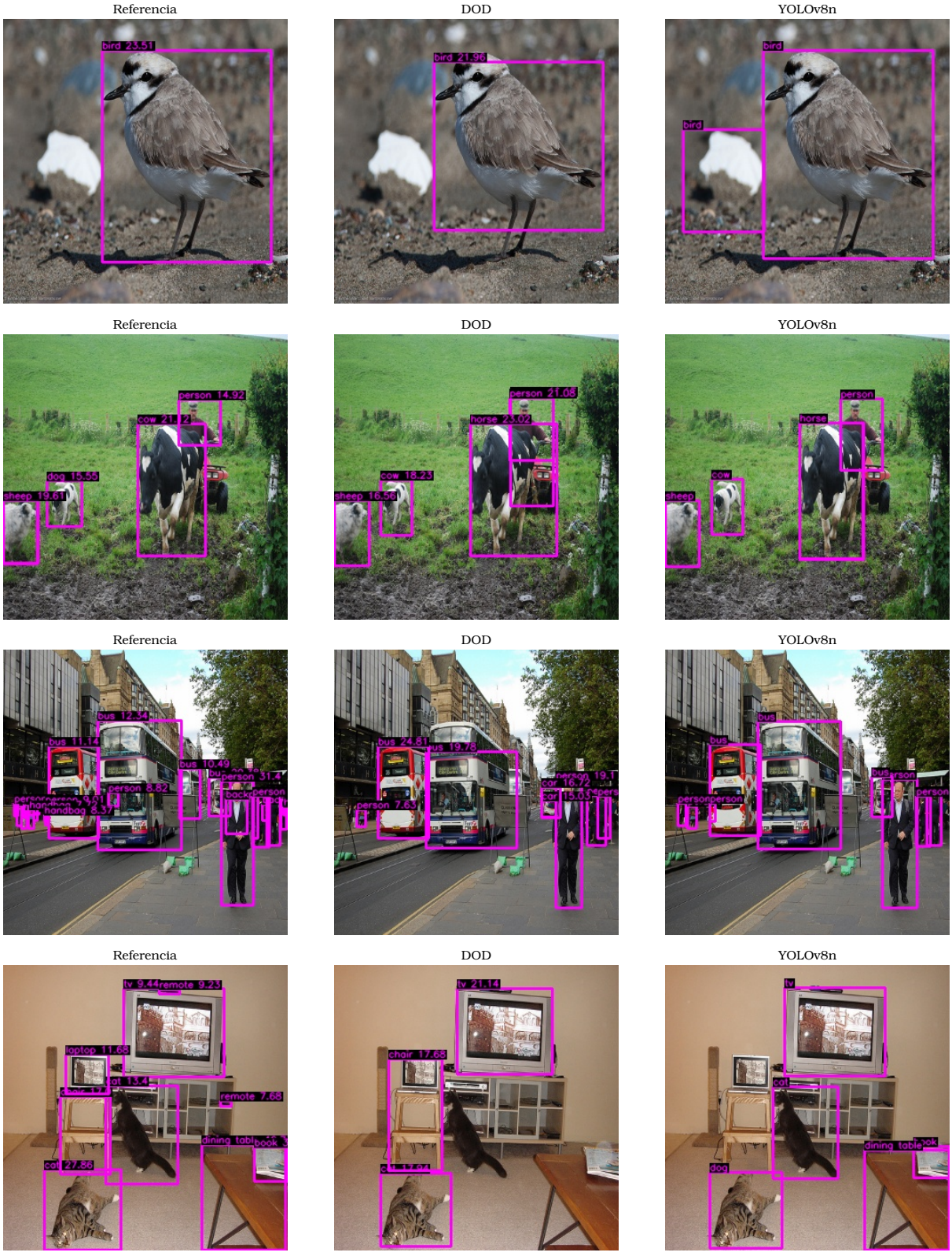


Figura 5.3: Resultados de inferencia de los modelos DOD y YOLOv8n después de aplicar la Supresión No Máxima con un umbral de confidencia del 10 % (Mejor F1-score, ver Figura 5.2) y un umbral de IoU del 60 %. Imágenes tomadas del conjunto de validación COCO [1].

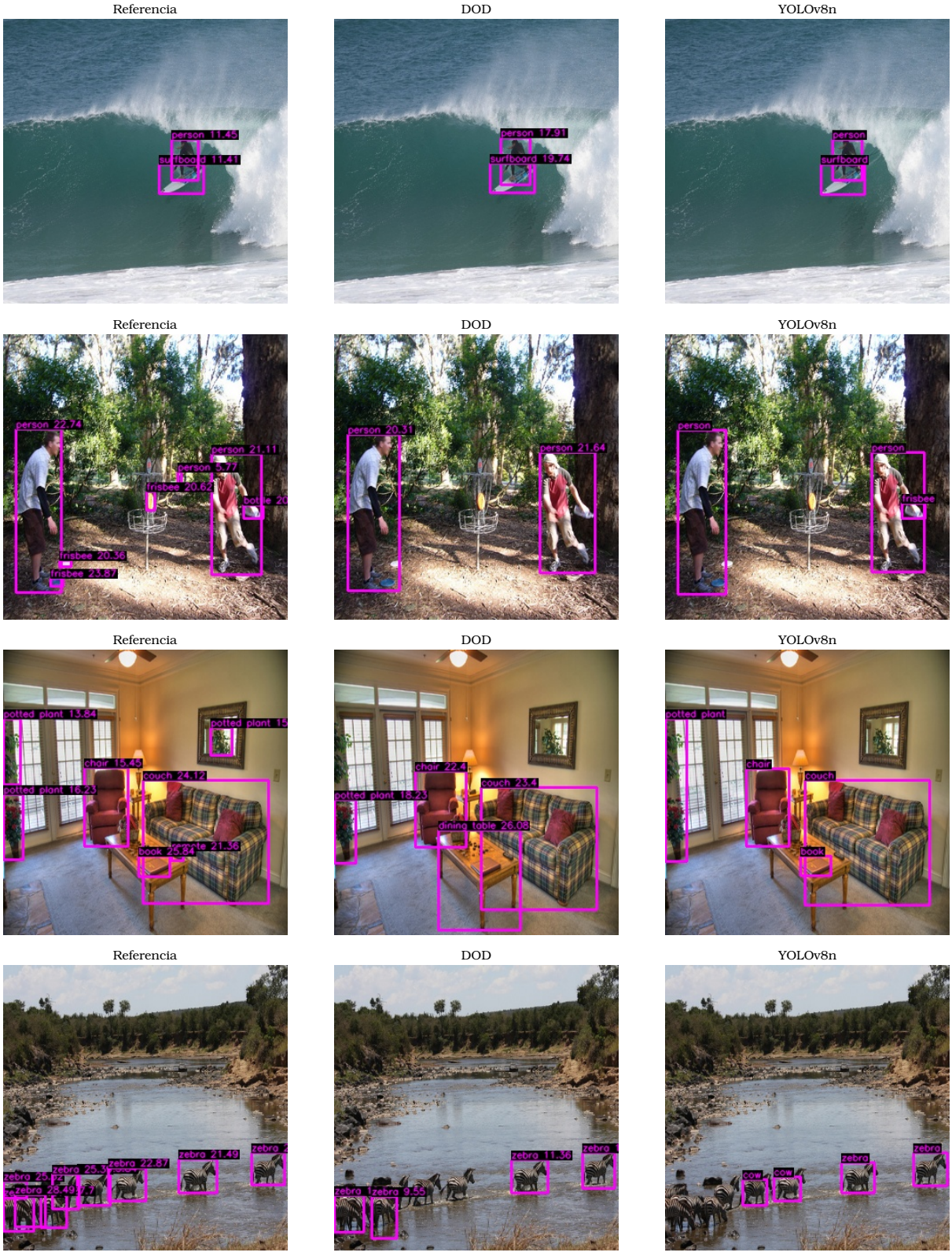


Figura 5.4: Resultados de inferencia de los modelos DOD y YOLOv8n después de aplicar la Supresión No Máxima con un umbral de confidencia del 10 % (Mejor F1-score, ver Figura 5.2) y un umbral de IoU del 60 %. Imágenes tomadas del conjunto de validación COCO [1].

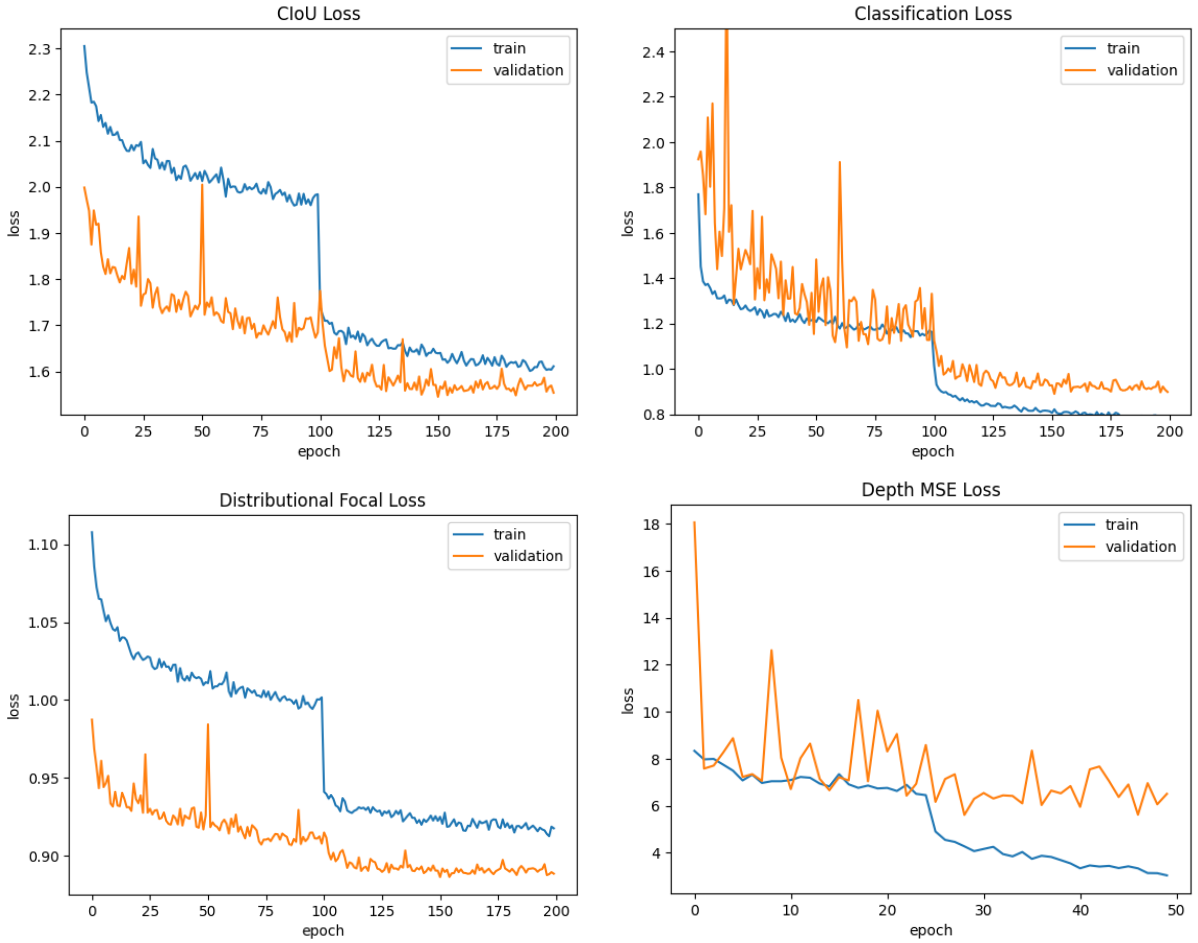


Figura 5.5: Curva de aprendizaje para cada factor de la Función de Pérdida (Ecuación 3.6) en el entrenamiento del modelo DOD sobre el conjunto de datos MinneApple [2, 3, 4]. Duración del entrenamiento: 2,426 segundos

5.2. MinneApple

5.2.1. Entrenamiento

La Figura 5.5 ilustra la curva de aprendizaje de cada término de la Función de Pérdida (Ecuación 3.6) en el entrenamiento del modelo DOD sobre el conjunto de datos MinneApple [2, 3, 4]. Los hiper-parámetros utilizados son:

- Optimizador Adam [86] con su variante AMSGrad [90].
- 3 épocas de calentamiento con un factor de aprendizaje lineal entre 0.0001 y 0.001.
- Factor de enfriamiento coseno entre 0.001 y 0.0001.
- Para la detección de objetos: 200 épocas de entrenamiento. MixUp: las primeras 100 épocas. Mosaico: no se aplica.
- Para la estimación de profundidad: 50 épocas de entrenamiento. MixUp: ninguna época. Mosaico: las primeras 25 épocas.

Modelo	P (%)	R (%)	mAP 50 (%)	mAP 50-95 (%)	MSE depth	Número de parámetros (M)
DOD	73.7	60.8	68.5	35.7	9.4	1.1
Tiled Faster RCNN	-	-	63.9	34.1	-	41
Faster RCNN	-	-	77.5	43.8	-	41
Mask RCNN	-	-	76.3	43.4	-	63

Cuadro 5.3: Métricas de evaluación obtenidas por los modelos entrenados y evaluados en MinneApple [2, 3, 4].

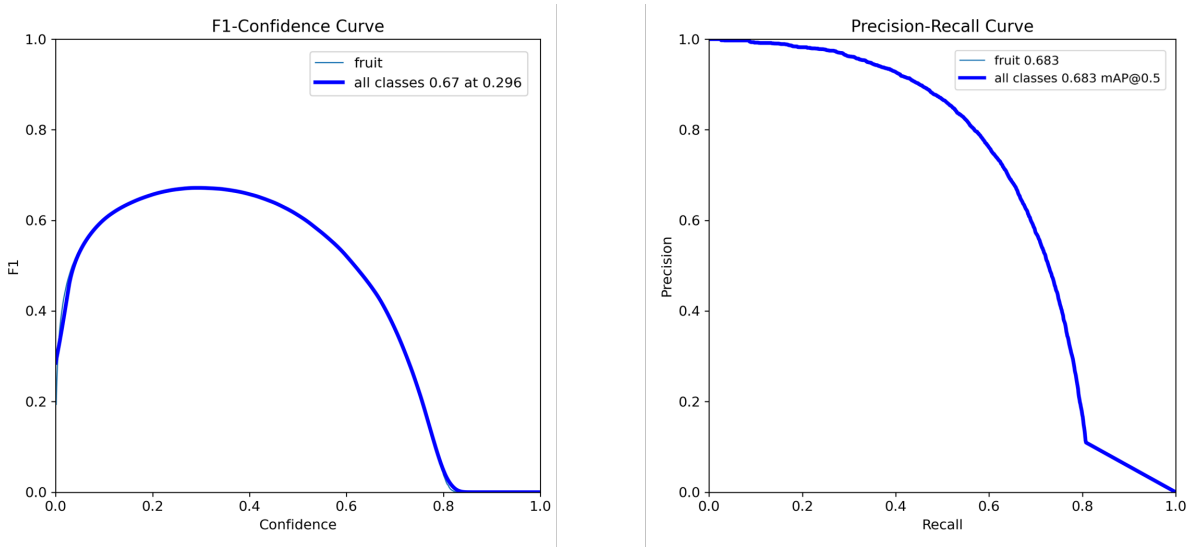


Figura 5.6: Curvas F1-Confidencia y Precisión-Recuperación del modelo de detección de frutos DOD entrenado y evaluado en MinneApple [2, 3, 4].

5.2.2. Validación

La Tabla 5.3 enuncia los resultados (descritos en la sección anterior) obtenidos por el modelo DOD junto con los resultados publicados por Häni *et al.* [2, 3, 4] por los modelos de detección basados en R-CNN (*Region with CNN features* [95]) con ResNet50 como etapa de extracción de características, pre-entrenados en COCO [1], y reentrenados y evaluados en MinneApple [2, 3, 4]:

- Tiled Faster R-CNN [2].
- Mask R-CNN [96].
- Faster R-CNN [67].

La Figura 5.6 muestra las curvas F1-Confidencia y Precisión-Recuperación obtenidas en la evaluación del modelo DOD. En Anexos: Figuras 9-10 se ilustran las curvas de Precisión-Confidencia y Recuperación-Confidencia, junto con la matriz de confusión normalizada y sin normalizar del modelo DOD.

Algunos resultados de inferencia sobre las imágenes de validación se pueden observar en las Figuras 5.7-5.10.

Figura 5.7: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] después de aplicar la Supresión No Máxima con un umbral de confidencia del 27 % (Mejor F1-score, ver Figura 5.6) y un umbral de IoU del 80 %. Imágenes tomadas de la partición de validación de MinneApple [2, 3, 4].

Figura 5.8: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] después de aplicar la Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes tomadas de la partición de validación de MinneApple [2, 3, 4].

Figura 5.9: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] después de aplicar la Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes tomadas de la partición de validación de MinneApple [2, 3, 4].

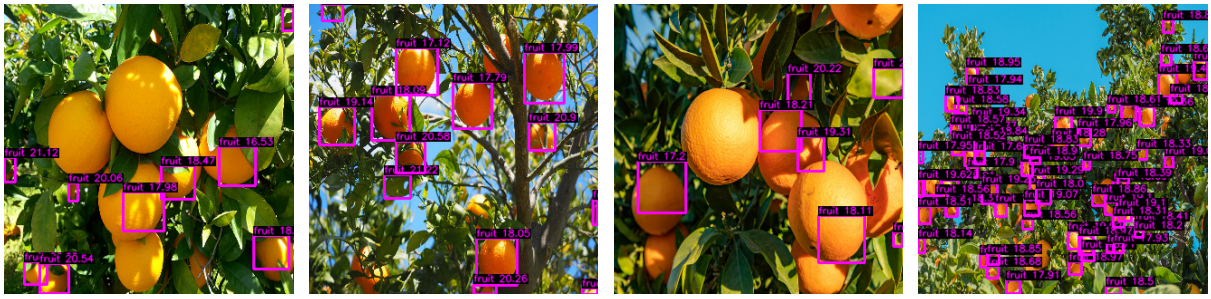


Figura 5.10: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4]. Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes de libre acceso tomadas de la Web.

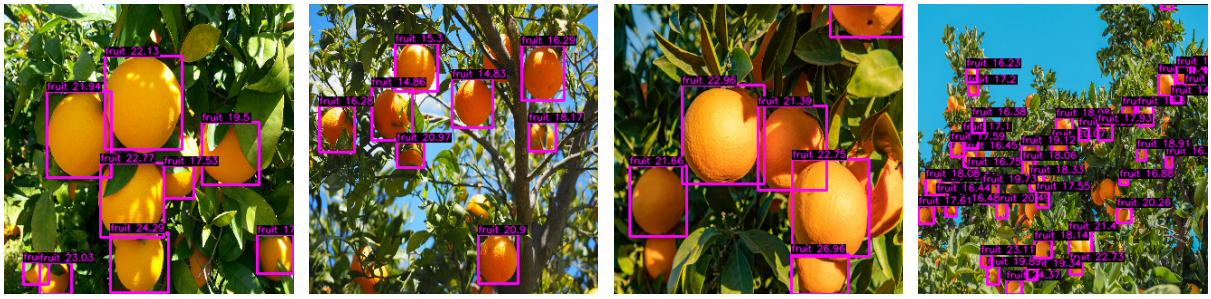


Figura 5.11: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] y Apples [63]. Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes de libre acceso tomadas de la Web.

Modelo	**P (%)**	**R (%)**	**mAP 50 (%)**	**mAP 50-95 (%)**	**MSE depth**
DOD (MinneApple + Apples)	68.6	58.1	62.4	31.9	8.5
DOD (MinneApple)	73.7	60.8	68.5	35.7	9.4

Cuadro 5.4: Métricas de evaluación obtenidas por los modelos DOD entrenados en MinneApple [2, 3, 4] y Apples [63] respectivamente, evaluados en MinneApple [2, 3, 4].

5.3. MinneApple y Apples

El modelo DOD entrenado con MinneApple [2, 3, 4] tiene un déficit de generalización en frutos con tamaños relativamente grandes o medianos, como se observa en la Figura 5.10. Por esta razón, se entrena un nuevo modelo DOD con la misma estrategia enunciada en la sección anterior, pero esta vez haciendo uso paralelo de los conjuntos de datos MinneApple [2, 3, 4] y Apples [63].

La Tabla 5.4 enuncia los resultados obtenidos en el conjunto de datos MinneApple [2, 3, 4], en comparación con los obtenidos en la sección anterior. Las curvas F1-Confidencia y Precisión-Recuperación se observan en Anexos: Figura 11. La Figura 5.11 muestra los resultados de inferencia para las mismas imágenes de la Figura 5.10.

Modelo	P (%)	R (%)	mAP 50 (%)	mAP 50-95 (%)	MSE depth	Velocidad CPU AMD (fps)	Velocidad CPU ARM (fps)
DOD (fp32)	68.6	58.1	62.4	31.9	8.5	27.2	2.14
DOD (int8)	67.2	57.8	61.4	30.4	9.3	33.6	2.34
DOD (ReLU int8)	63.7	54.2	57.7	28.5	31.7	32.3	2.36

Cuadro 5.5: Métricas de evaluación obtenidas por los modelos DOD de distinta precisión entrenados en MinneApple [2, 3, 4] y Apples [63], evaluados en MinneApple [2, 3, 4].

Modelo	Número de parámetros (M)	Espacio de almacenamiento (MB)
DOD (fp32)	1.04	4.18
DOD (SiLU/ReLU int8)	1.04	1.32

Cuadro 5.6: Tamaños de las arquitecturas evaluadas.

5.4. Cuantización

Las Tablas 5.5 y 5.6 describe los resultados obtenidos en la evaluación en MinneApple [2, 3, 4] por el modelo DOD entrenado en MinneApple [2, 3, 4] y Apples [63] en precisión de punto flotante de 32-bits, precisión entera con signo de 8-bits, y su versión modificada con ReLU como función de activación.

Las velocidades descritas corresponden al tiempo medio de inferencia para cuatro veces las imágenes de la partición de validación con batch unitario en los procesadores:

- AMD Ryzen 7 5800H 3.20 GHz.
- ARM Cortex-A72 1.5 GHz 64-bits: Broadcom SoC BCM2711 en el sistema embebido Raspberry Pi 4.

Las Figuras 5.12-5.15 ilustran la comparación de algunos resultados de inferencia obtenidos por:

- El modelo DOD de precisión punto flotante de 32-bits entrenado en MinneApple [2, 3, 4].
- El modelo DOD de precisión punto flotante de 32-bits entrenado en MinneApple [2, 3, 4] y Apples [63].
- El modelo DOD de precisión entera con signo de 8-bits entrenado en MinneApple [2, 3, 4] y Apples [63].

Figura 5.12: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] y Apples [63] respectivamente, para diferentes precisiones. Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes tomadas de la partición de validación de MinneApple [2, 3, 4].

Figura 5.13: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] y Apples [63] respectivamente, para diferentes precisiones. Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes tomadas de la partición de validación de MinneApple [2, 3, 4].

Figura 5.14: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] y Apples [63] respectivamente, para diferentes precisiones. Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes tomadas de la partición de validación de MinneApple [2, 3, 4].

Figura 5.15: Resultados de inferencia del modelo DOD entrenado en MinneApple [2, 3, 4] y Apples [63] respectivamente, para diferentes precisiones. Supresión No Máxima con un umbral de confidencia del 20 % y un umbral de IoU del 80 %. Imágenes de libre acceso tomadas de la Web.

Capítulo 6

Conclusiones

En este trabajo, se ha presentado el modelo DOD, un método novedoso de visión artificial para la detección de objetos con estimación de profundidad para aplicaciones de tiempo real en sistemas embebidos o microcontrolados de bajo costo. La concepción, el diseño, la implementación y la operación del modelo propuesto se han inspirado en el estado del arte actual de las tecnologías clave en la detección de objetos.

La capacidad de detección del modelo propuesto se ha validado mediante la evaluación en el conjunto de datos COCO [1], y la comparación con el modelo YOLOv8n [54] que establece el estado del arte actual. A pesar que las métricas obtenidas son inferiores, el modelo DOD logra resultados visuales satisfactorios en esta compleja tarea con 80 clases, todo ello con una arquitectura de aproximadamente 1 millón de parámetros.

Por otra parte, el desempeño en la detección de frutos se evalúa en el conjunto de datos MinneApple [2, 3, 4]. Los resultados obtenidos superan las expectativas al obtener métricas más altas en comparación con el método propuesto por Häni *et al.* [2, 3, 4], de al menos 41 millones de parámetros. Los resultados visuales junto las métricas validan la eficacia y precisión del modelo DOD para esta tarea.

Respecto a la estimación de profundidad, la evaluación del modelo DOD se limita al error cuadrático medio debido a la carencia, en el momento de publicación de este trabajo, de un método análogo evaluado en estas tareas, así como de conjuntos de datos de detección con etiquetas de profundidad obtenidas a través de mediciones físicas fiables.

En cuanto al rendimiento del modelo cuantizado en el sistema embebido Raspberry Pi 4, es relevante resaltar su pequeña huella de memoria y su velocidad de inferencia. Estos dos aspectos son esenciales para las aplicaciones en tiempo real y de bajo costo, lo que hace que el modelo DOD propuesto en este trabajo sea una solución especializada para estas aplicaciones.

Como trabajo futuro, es necesario ajustar los rangos de los valores de profundidad obtenidos en las detecciones mediante la calibración con mediciones de instrumentos físicos. Este proceso permitirá recopilar y analizar una gran cantidad de experimentos que, a su vez, servirán para el re-entrenamiento y ajuste de la precisión en la estimación de la profundidad en las detecciones del modelo DOD. El código fuente se encuentra en el repositorio GitHub Depth Object Detector DOD.

Bibliografía

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[2] N. Häni, P. Roy, and V. Isler, "Minneapple: A benchmark dataset for apple detection and segmentation," 2019.

[3] ——, "A comparative study of fruit detection and counting methods for yield mapping in apple orchards," *Journal of Field Robotics*, vol. 37, no. 2, pp. 263–282, aug 2019. [Online]. Available: https://doi.org/10.1002%2Frob.21902

[4] ——, "Apple counting using convolutional neural networks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2559–2565.

[5] N. Jahan, T. Akilan, and A. R. Phalke, "Machine learning for global food security: A concise overview," in *2022 IEEE International Humanitarian Technology Conference (IHTC)*, 2022, pp. 63–68.

[6] P. Kolhe, K. Kalbande, and A. Deshmukh, "Internet of thing and machine learning approach for agricultural application: A review," in *2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22)*, 2022, pp. 1–6.

[7] C. Kiruthiga and K. Dharmarajan, "Machine learning in soil borne diseases, soil data analysis crop yielding: A review," in *2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 2023, pp. 702–706.

[8] A. Basharat and M. M. B. Mohamad, "Security challenges and solutions for internet of things based smart agriculture: A review," in *2022 4th International Conference on Smart Sensors and Application (ICSSA)*, 2022, pp. 102–107.

[9] V. Ranganathan, P. Kumar, U. Kaur, S. H. Li, T. Chakraborty, and R. Chandra, "Re-inventing the food supply chain with iot: A data-driven solution to reduce food loss," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 41–47, 2022.

[10] D. Bini, D. Pamela, and S. Prince, "Machine vision and machine learning for intelligent agrobots: A review," in *2020 5th International Conference on Devices, Circuits and Systems (ICDCS)*, 2020, pp. 12–16.

[11] M. Sharma and N. Hema, "Comparison of agricultural drones and challenges in implementation: A review," in *2021 7th International Conference on Signal Processing and Communication (ICSC)*, 2021, pp. 26–30.

[12] M. Shahrooz, A. Talaeizadeh, and A. Alasty, "Agricultural spraying drones: Advantages and disadvantages," in *2020 Virtual Symposium in Plant Omics Sciences (OMICAS)*, 2020, pp. 1–5.

[13] X. Zhang, Z. Cao, and W. Dong, "Overview of edge computing in the agricultural internet of things: Key technologies, applications, challenges," *IEEE Access*, vol. 8, pp. 141 748–141 761, 2020.

[14] United Nations Department of Economic and Social Affairs, Population Division, "World population prospects 2022: Summary of results," United Nations, Office of the Director, Population Division, Department of Economic and Social Affairs, United Nations, New York, 10017, USA, Technical Report, 2022, this report is available in electronic format on the Division's website. [Online]. Available: http://www.unpopulation.org

[15] (2023) "digital action" @ WSIS forum 2023: FAO takes stock of agrifood systems transformation for SDGs. Food and Agriculture Organization of the United Nations. [Online]. Available: https://www.fao.org/e-agriculture/news/digital-action%E2%80%9D-wsis-forum-2023-fao-takes-stock-agrifood-systems-transformation-sdgs

[16] Z. Bauman, *Retrotopía*. PAIDOS IBERICA, 2017. [Online]. Available: https://www.casadellibro.com/libro-retrotopia/9788449333224/5185713

[17] E. Briones-Vozmediano and A. González-González, "Explotación y precariedad sociolaboral, la realidad de las personas migrantes trabajadoras en agricultura en España," *Archivos de Prevención de Riesgos Laborales*, vol. 25, pp. 18 – 24, 03 2022. [Online]. Available: https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1578-25492022000100018

[18] Euronews. (2020, July) La explotación laboral en los campos europeos, en el punto de mira. Euronews. Accessed: 22-07-2023. [Online]. Available: https://es.euronews.com/my-europe/2020/07/17/la-explotacion-laboral-en-los-campos-europeos-en-el-punto-de-mira

[19] G. de España. (2023) Coordinated intelligent services for adaptive smart areas. Accessed: 20-07-2023. [Online]. Available: https://cosass.usal.es/

[20] A. Nanda, K. K. Swain, K. S. Reddy, and R. Agarwal, "stransporter: An autonomous robotics system for collecting fresh fruit crates for the betterment of the post harvest handling process," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 577–582.

[21] H. Kang, H. Zhou, and C. Chen, "Visual perception and modeling for autonomous apple harvesting," *IEEE Access*, vol. 8, pp. 62 151–62 163, 2020.

[22] R. Arikapudi and S. G. Vougioukas, "Robotic tree-fruit harvesting with telescoping arms: A study of linear fruit reachability under geometric constraints," *IEEE Access*, vol. 9, pp. 17 114–17 126, 2021.

[23] J. F. Elfferich, D. Dodou, and C. D. Santina, "Soft robotic grippers for crop handling or harvesting: A review," *IEEE Access*, vol. 10, pp. 75 428–75 443, 2022.

[24] A. Qiu, C. Young, A. L. Gunderman, M. Azizkhani, Y. Chen, and A.-P. Hu, "Tendon-driven soft robotic gripper with integrated ripeness sensing for blackberry harvesting," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 831–11 837.

[25] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu, "Light Field Image Processing: An Overview," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 7, pp. 926–954, oct 2017.

[26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020.

[27] B. Fei, W. Yang, W.-M. Chen, Z. Li, Y. Li, T. Ma, X. Hu, and L. Ma, "Comprehensive review of deep learning-based 3d point cloud completion processing and analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22 862–22 883, 2022.

[28] H.-M. Wang, H.-Y. Lin, and C.-C. Chang, "Object detection and depth estimation approach based on deep convolutional neural networks," *Sensors*, vol. 21, no. 14, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/14/4755

[29] Y. Lee, H. Lee, E. Lee, H. Kwon, and S. Bhattacharyya, "Exploiting simplified depth estimation for stereo-based 2d object detection," in *2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2022, pp. 1–6.

[30] C. Fan, Z. Yin, X. Huang, M. Li, X. Wang, and H. Li, "Faster 3d reconstruction by fusing 2d object detection and self-supervised monocular depth estimation," in *2022 11th International Conference of Information and Communication Technology (ICTech))*, 2022, pp. 492–497.

[31] M. Usman and Q. Ling, "Point-pixel fusion for object detection and depth estimation," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 5458–5462.

[32] H.-M. Wang and H.-Y. Lin, "A real-time forward collision warning technique incorporating detection and depth estimation networks," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 1966–1971.

[33] C. Park, H. Kim, M. Kim, J. Sung, and J. Paik, "Monocular 3d object detection of moving objects using random sampling and deep layer aggregation," in *2023 IEEE International Conference on Consumer Electronics (ICCE)*, 2023, pp. 1–2.

[34] H. Kato, F. Nagata, Y. Murakami, and K. Koya, "Partial depth estimation with single image using yolo and cnn for robot arm control," in *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2022, pp. 1727–1731.

[35] S. Pogaru, A. Bose, D. Elliott, and J. O'Keefe, "Multiple object association incorporating object tracking, depth, and velocity analysis on 2d videos," in *SoutheastCon 2021*, 2021, pp. 1–6.

[36] C. Xu, B. Huang, and D. S. Elson, "Self-supervised monocular depth estimation with 3-d displacement module for laparoscopic images," *IEEE Transactions on Medical Robotics and Bionics*, vol. 4, no. 2, pp. 331–334, 2022.

[37] *An Introduction to the Biology of Vision*. Cambridge University Press, 1996.

[38] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, 2001, pp. 131–140.

[39] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition*, ser. Texts in Computer Science. Springer, 2022. [Online]. Available: https://doi.org/10.1007/978-3-030-34372-9

[40] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[41] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," 2017.

[42] S. Bazrafkan, H. Javidnia, J. Lemley, and P. Corcoran, "Semiparallel deep neural network hybrid architecture: first application on depth from monocular camera," *Journal of Electronic Imaging*, vol. 27, no. 4, p. 043041, 2018. [Online]. Available: https://doi.org/10.1117/1.JEI.27.4.043041

[43] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and D. Puig, "Monocular depth estimation using deep learning: A review," *Sensors*, vol. 22, no. 14, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/14/5353

[44] K. Xian, C. Shen, Z. Cao, H. Lu, Y. Xiao, R. Li, and Z. Luo, "Monocular relative depth perception with web stereo data supervision," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[45] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *ICCV*, 2021.

[46] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, 2022.

[47] Y. Liu, Y. Zhang, Y. Wang, F. Hou, J. Yuan, J. Tian, Y. Zhang, Z. Shi, J. Fan, and Z. He, "A survey of visual transformers," 2022.

[48] W. Zhao, Y. Rao, Z. Liu, B. Liu, J. Zhou, and J. Lu, "Unleashing text-to-image diffusion models for visual perception," 2023.

[49] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, F. Aleotti, and S. Mattoccia, "Monocular depth perception on microcontrollers for edge applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1524–1536, 2022.

[50] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "Tood: Task-aligned one-stage object detection," 2021.

[51] T. Zhang, B. Luo, A. Sharda, and G. Wang, "Dynamic label assignment for object detection by combining predicted IoUs and anchor IoUs," *Journal of Imaging*, vol. 8, no. 7, p. 193, jul 2022. [Online]. Available: https://doi.org/10.3390%2Fjimaging8070193

[52] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[53] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 and beyond," 2023.

[54] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[55] C. Zhang, C. Zhang, C. Li, Y. Qiao, S. Zheng, S. K. Dam, M. Zhang, J. U. Kim, S. T. Kim, J. Choi, G.-M. Park, S.-H. Bae, L.-H. Lee, P. Hui, I. S. Kweon, and C. S. Hong, "One small step for generative ai, one giant leap for agi: A complete survey on chatgpt in aigc era," 2023.

[56] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[57] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," 2023.

[58] Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," 2020.

[59] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4340–4349.

[60] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind, "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *International Conference on Computer Vision (ICCV) 2021*, 2021.

[61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[62] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[63] F. Ciaglia, F. S. Zuppichini, P. Guerrie, M. McQuade, and J. Solawetz, "Roboflow 100: A rich, multi-domain object detection benchmark," 2022.

[64] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[65] B. Cetinkaya, S. Kalkan, and E. Akbas, "Does depth estimation help object detection?" 2022.

[66] A. J. Xiang, A. B. Huddin, M. F. Ibrahim, and F. H. Hashim, "An oil palm loose fruits image detection system using faster r -cnn and jetson tx2," in *2021 Inter-*

national Conference on Electrical Engineering and Informatics (ICEEI), 2021, pp. 1–6.

[67] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[68] Y. Nagaraju, Venkatesh, and K. R. Venugopal, "A fruit detection method for vague environment high-density fruit orchards," in *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, 2022, pp. 1–6.

[69] G. Jocher, "Ultralytics yolov5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[70] "Ultralytics:yolov8," https://docs.ultralytics.com/models/yolov8/, 2023, accessed: 1-08-2023.

[71] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," 2022.

[72] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[73] X. Li, C. Lv, W. Wang, G. Li, L. Yang, and J. Yang, "Generalized focal loss: Towards efficient representation learning for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3139–3153, 2023.

[74] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[75] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," 2017.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[77] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2013.

[78] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "Cspnet: A new backbone that can enhance learning capability of cnn," 2019.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. [Online]. Available: https://doi.org/10.1007%2F978-3-319-10578-9_23

[80] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with yolov8," 2023.

[81] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," 2019.

[82] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018,*

Conference Track Proceedings. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[83] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[84] "Python programming language," https://www.python.org/, accessed: 16-08-2023.

[85] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[87] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," 2021.

[88] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," 2017.

[89] I. S. Isa, M. S. A. Rosli, U. K. Yusof, M. I. F. Maruzuki, and S. N. Sulaiman, "Optimizing the hyperparameter tuning of yolov5 for underwater detection," *IEEE Access*, vol. 10, pp. 52 818–52 831, 2022.

[90] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[91] M. D. Awheda and H. M. Schwartz, "Exponential moving average q-learning algorithm," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2013, pp. 31–38.

[92] Raspberry Pi Foundation, "Raspberry pi," https://www.raspberrypi.org/, 2023, accessed: 21-08-2023.

[93] W. Gay, *Raspberry Pi Hardware Reference*, 1st ed. USA: Apress, 2014.

[94] M. Dukhan, Y. Wu, H. Lu, and B. Maher, "Qnnpack: Quantized neural network package," https://github.com/pytorch/QNNPACK, 2019, accessed: 22-08-2023.

[95] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[97] "RangeKing: Brief summary of yolov8 model structure," https://github.com/ultralytics/ultralytics/issues/189, 2023, accessed: 1-08-2023.

Anexos

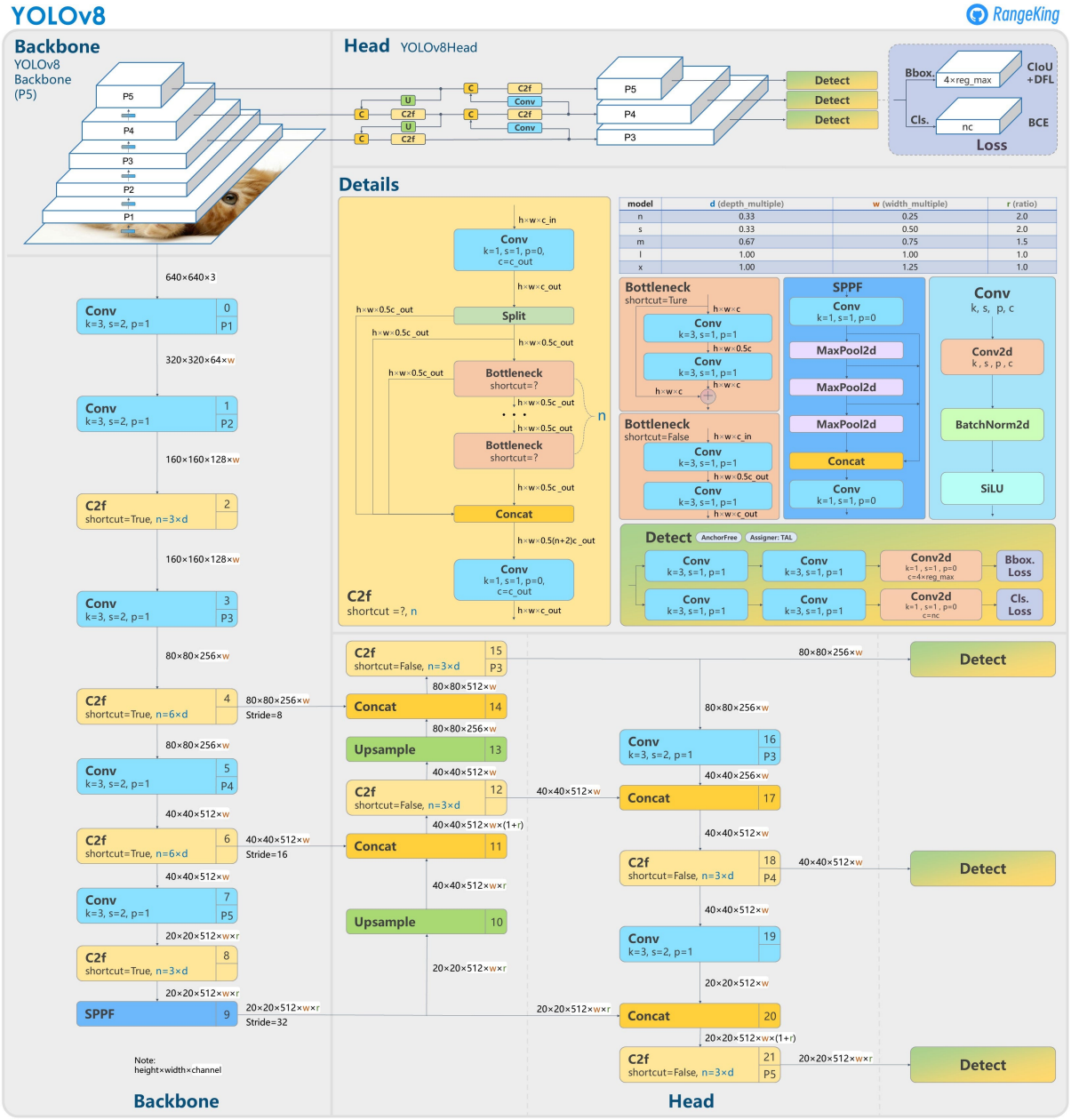


Figura 1: Arquitectura del modelo de detección de objetos YOLOv8, propuesto por Jocher *et al.* [54] en febrero de 2023. Figura tomada de [97].

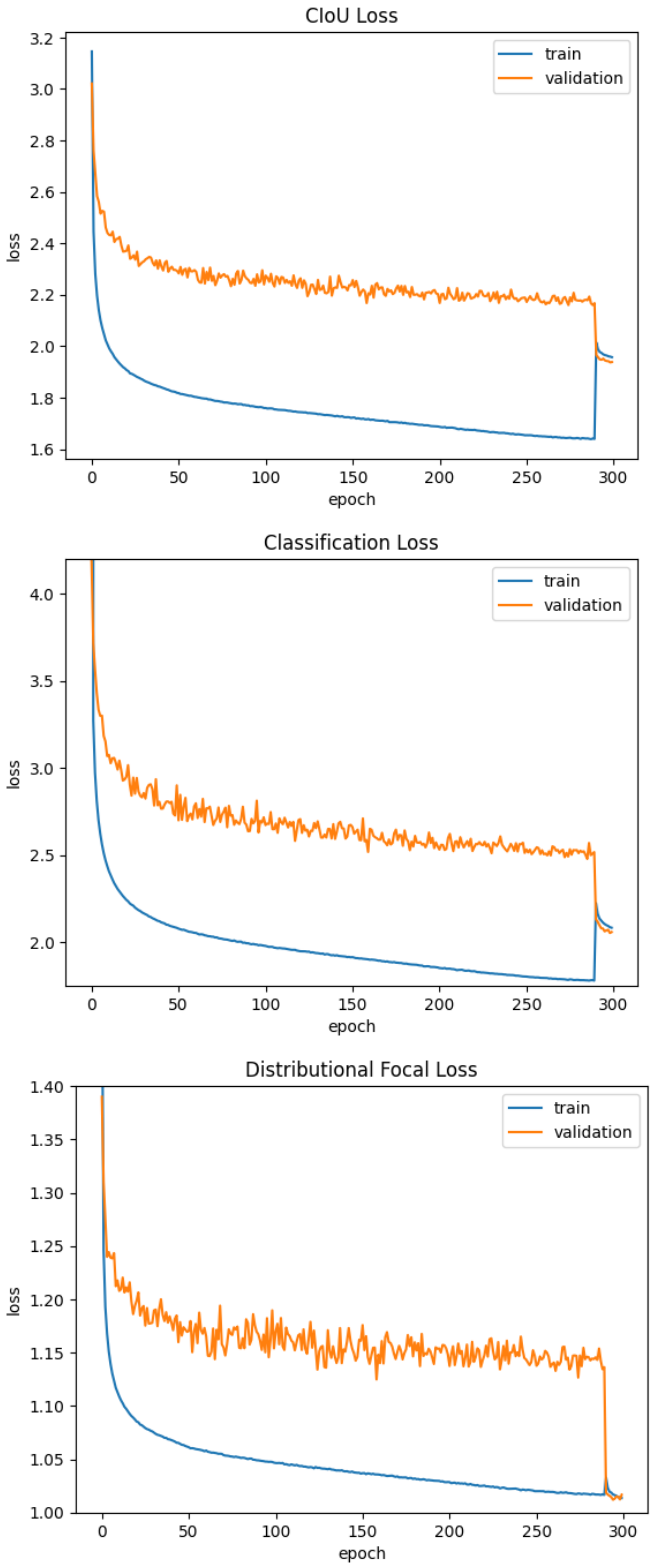


Figura 2: Curva de aprendizaje para cada factor de detección en la Función de Pérdida 3.6. Primer experimento en COCO [1] para la tarea de detección de objetos. Duración total del experimento: 87.110 segundos.

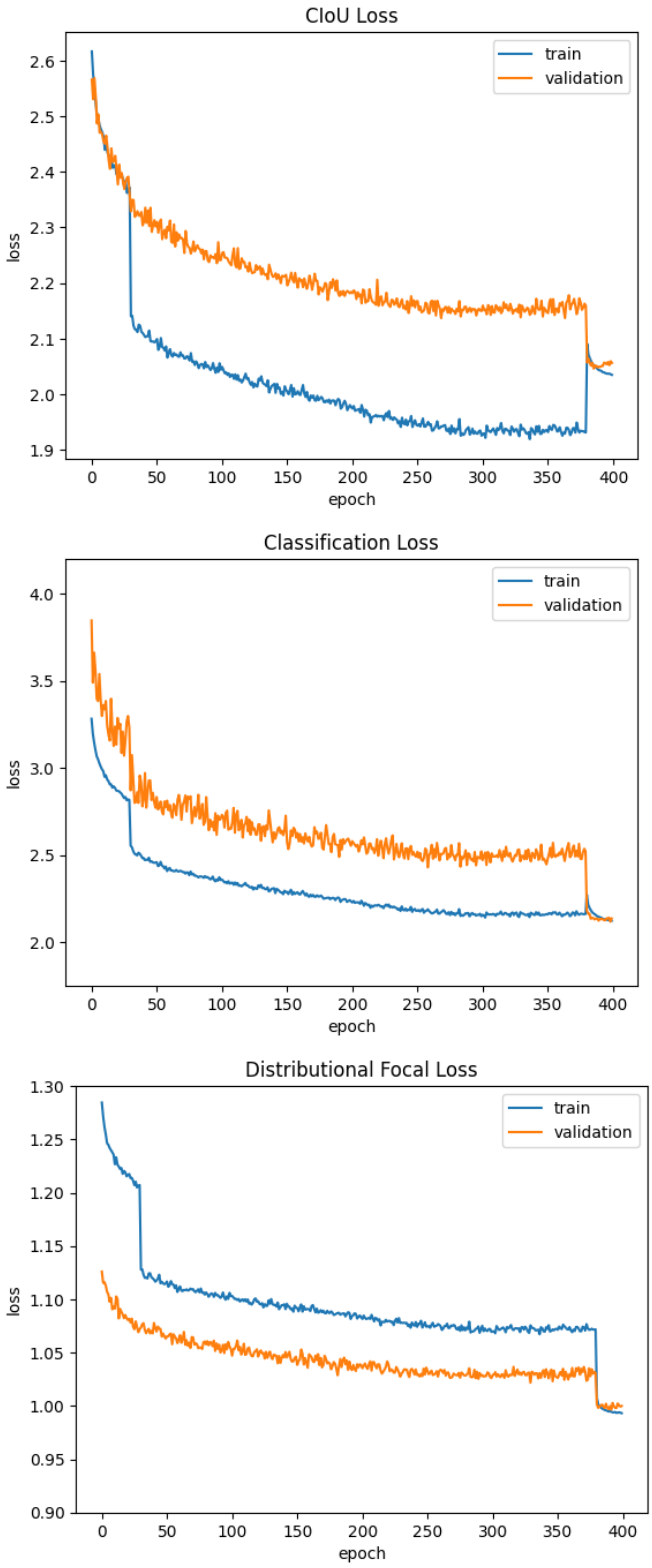
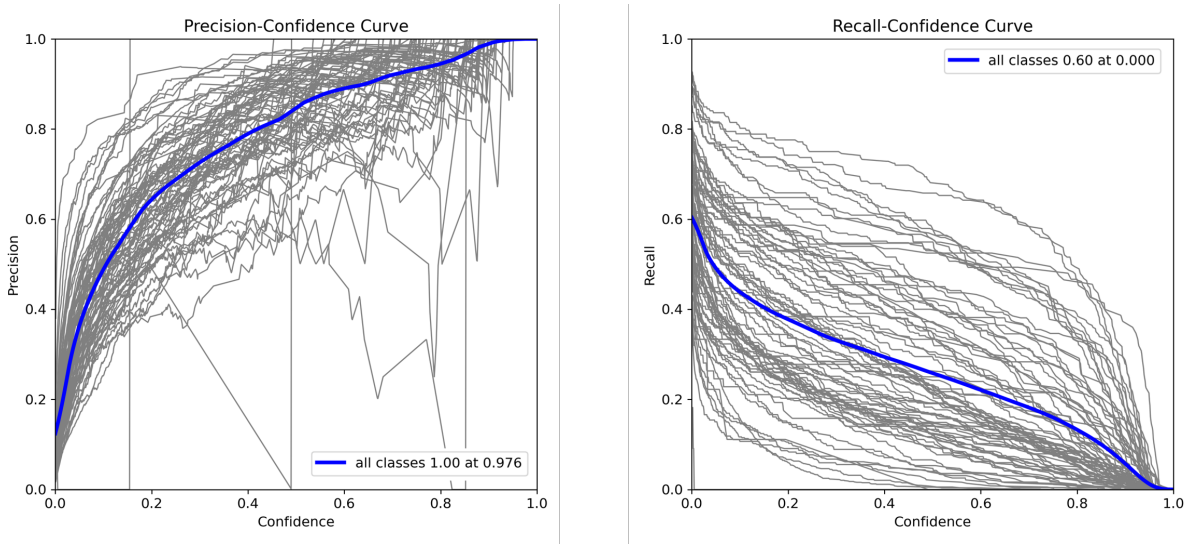


Figura 3: Curva de aprendizaje para cada factor de detección en la Función de Pérdida 3.6. Segundo experimento en COCO [1] para la tarea de detección de objetos. Duración total aproximada del experimento: 120.000 segundos.

(a) modelo YOLOv8n



(b) modelo DOD

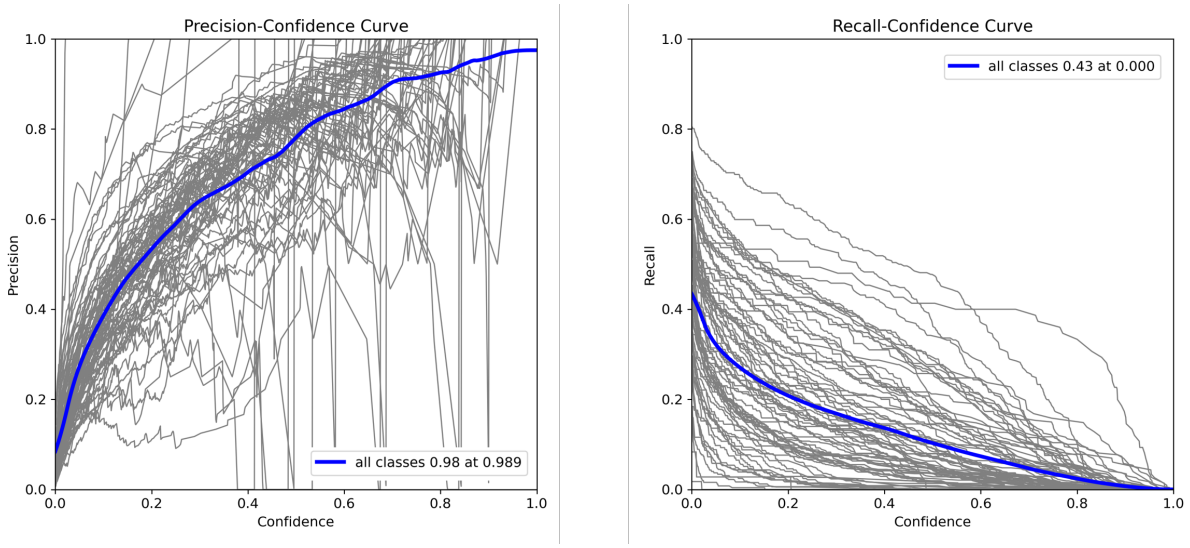


Figura 4: Curvas de Precisión-Confidencia y Recuperación-Confidencia de los modelos de detección de objetos YOLOv8n y DOD entrenados y evaluados en COCO [1].

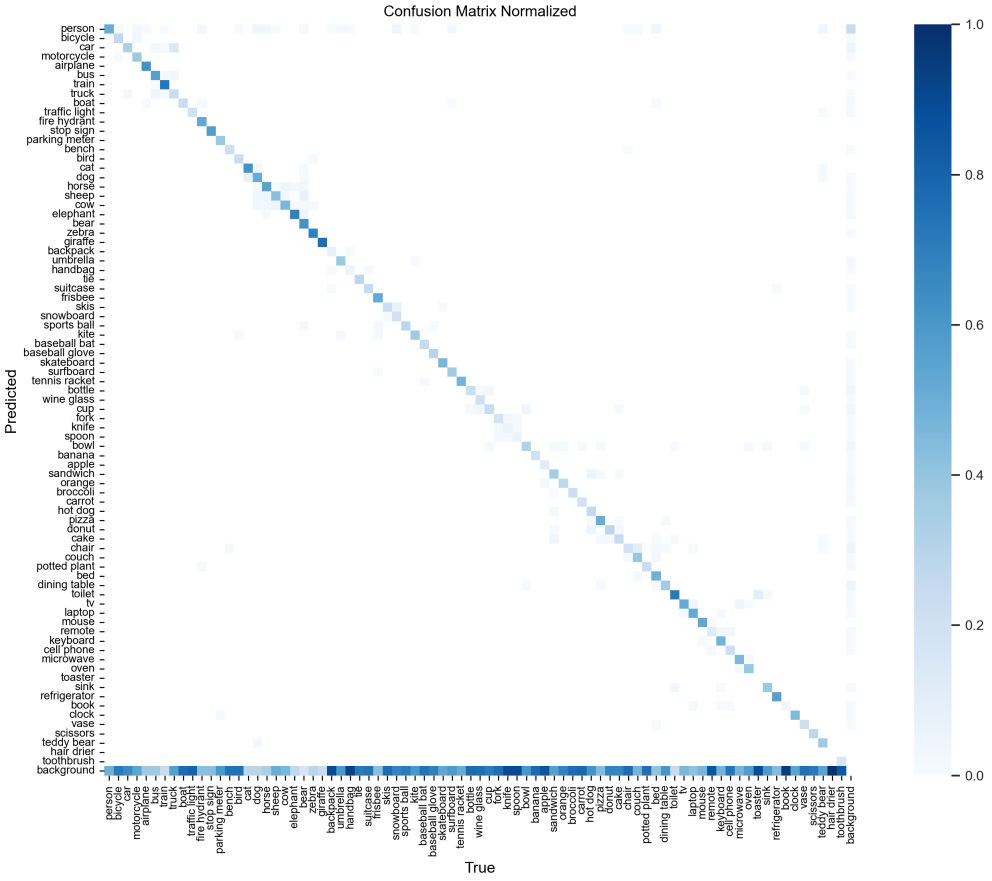


Figura 5: Modelo YOLOv8n: Matriz de Confusión normalizada que muestra los verdaderos positivos y los falsos negativos para cada clase de COCO [1].

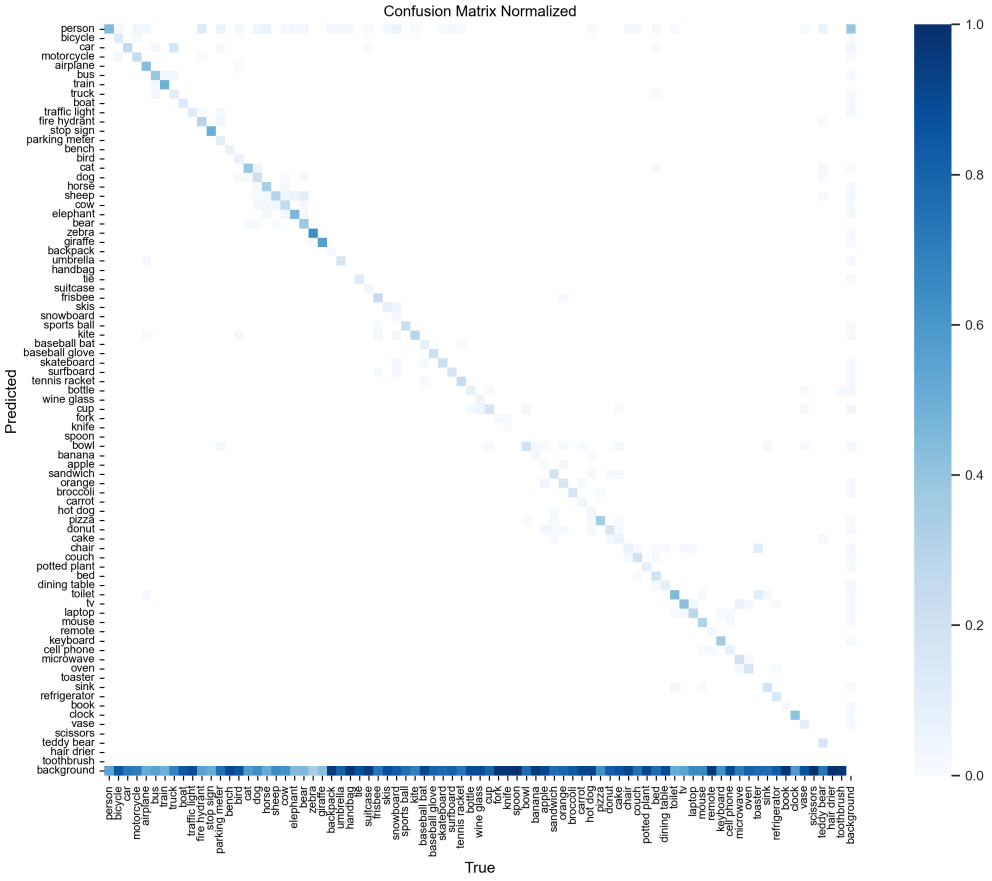


Figura 6: Modelo DOD: Matriz de Confusión normalizada que muestra los verdaderos positivos y los falsos negativos para cada clase de COCO [1].

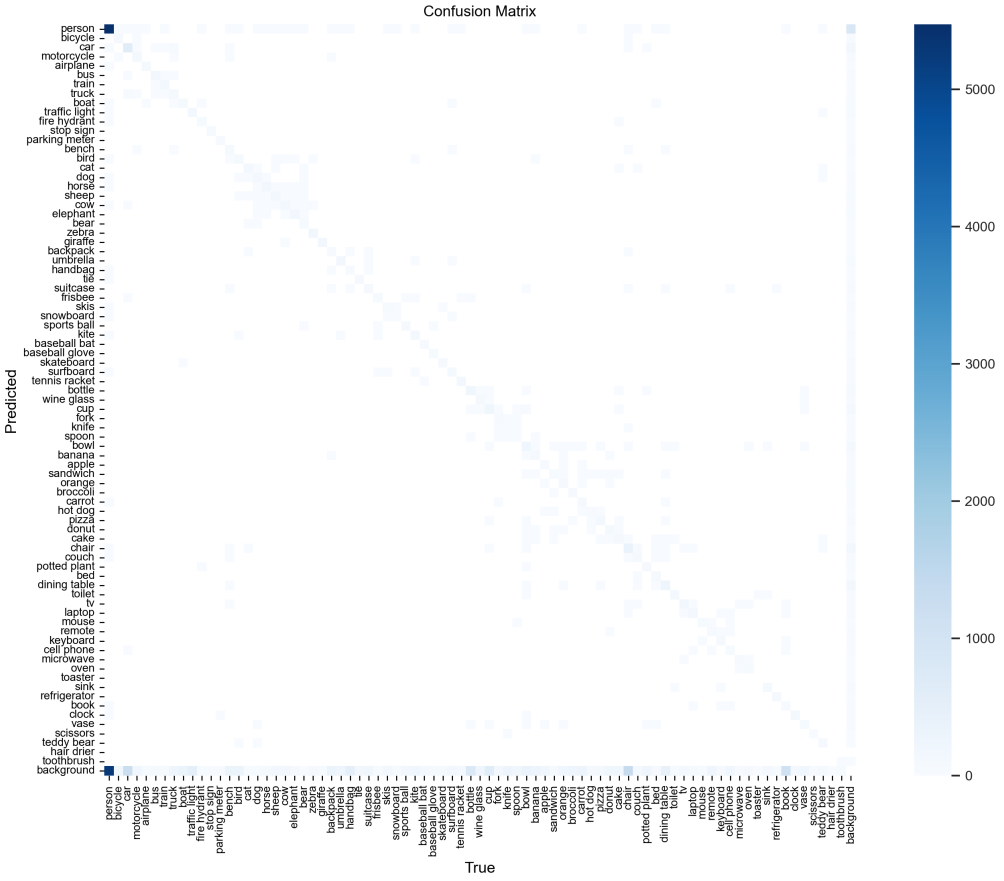


Figura 7: Modelo YOLOv8n: Matriz de Confusión sin normalizar que muestra los verdaderos positivos y los falsos negativos para cada clase de COCO [1].

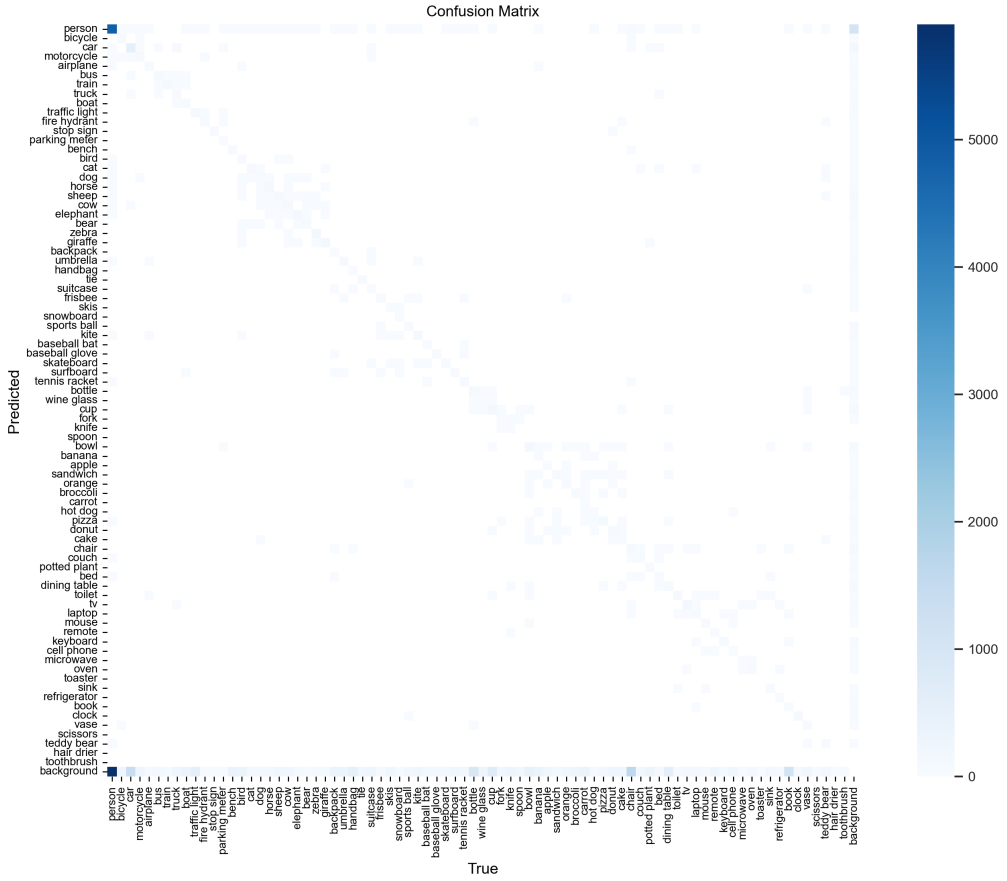


Figura 8: Modelo DOD: Matriz de Confusión sin normalizar que muestra los verdaderos positivos y los falsos negativos para cada clase de COCO [1].

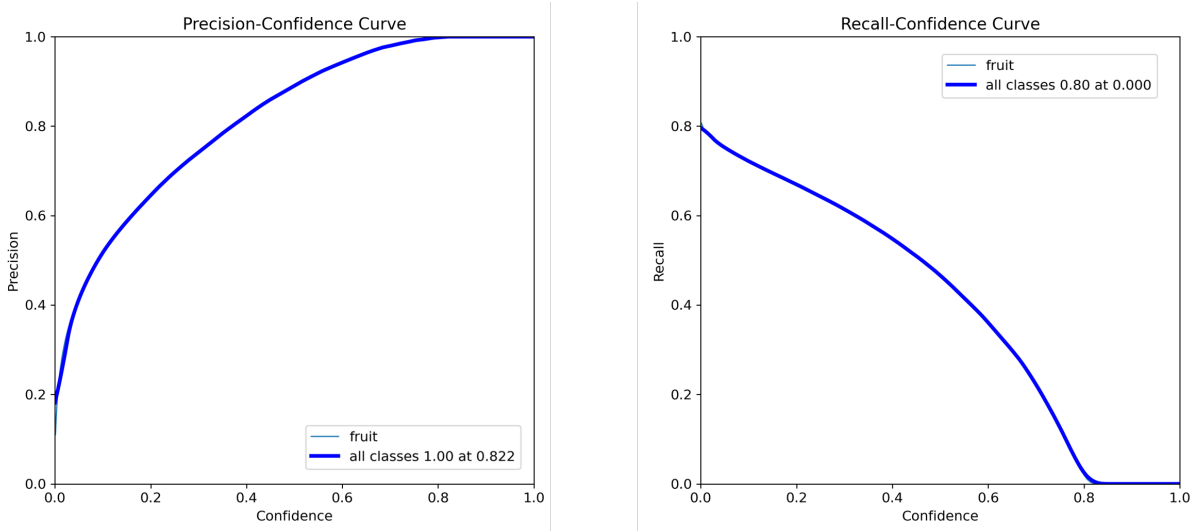


Figura 9: Curvas de Precisión-Confidencia y Recuperación-Confidencia del modelo de detección de frutos DOD entrenado y evaluado en MinneApple [2, 3, 4].

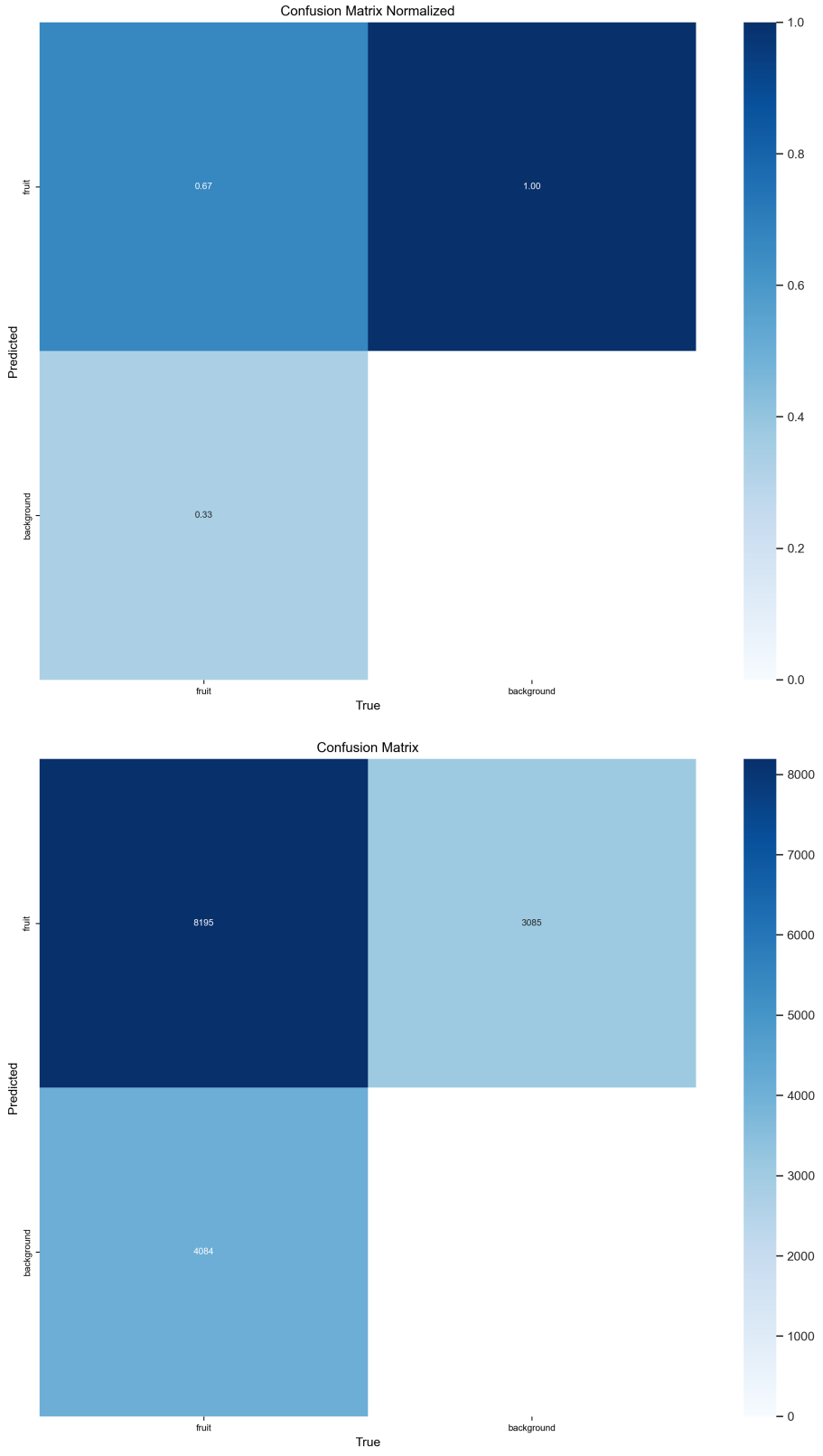
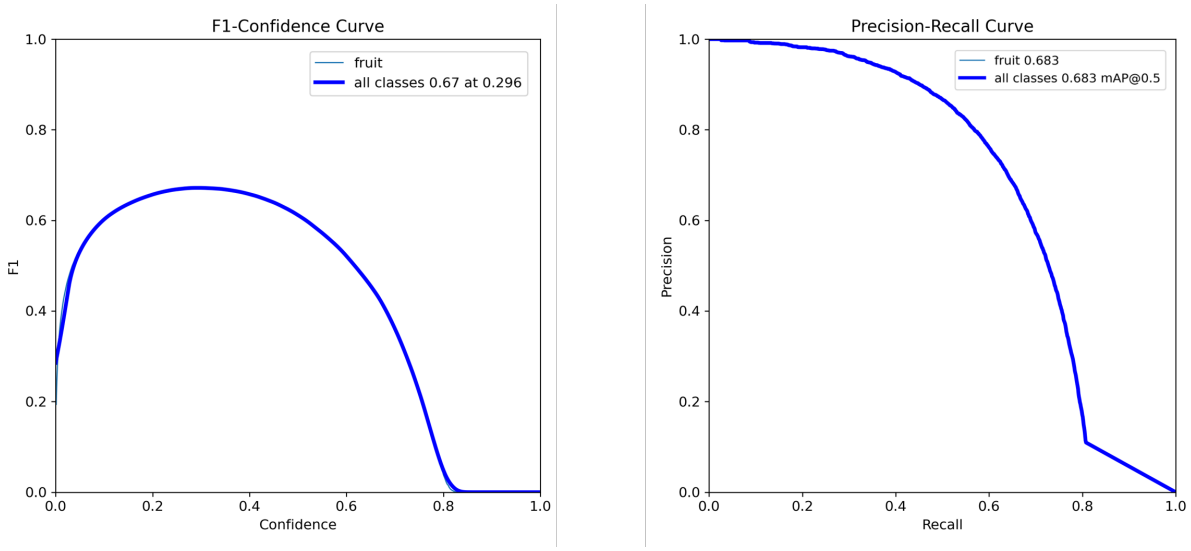


Figura 10: Modelo DOD: Matriz de Confusión que muestra los verdaderos positivos y los falsos negativos para la clase fruta en MinneApple [2, 3, 4].

(a) Entrenado en MinneApple



(b) Entrenado en MinneApple + Apples

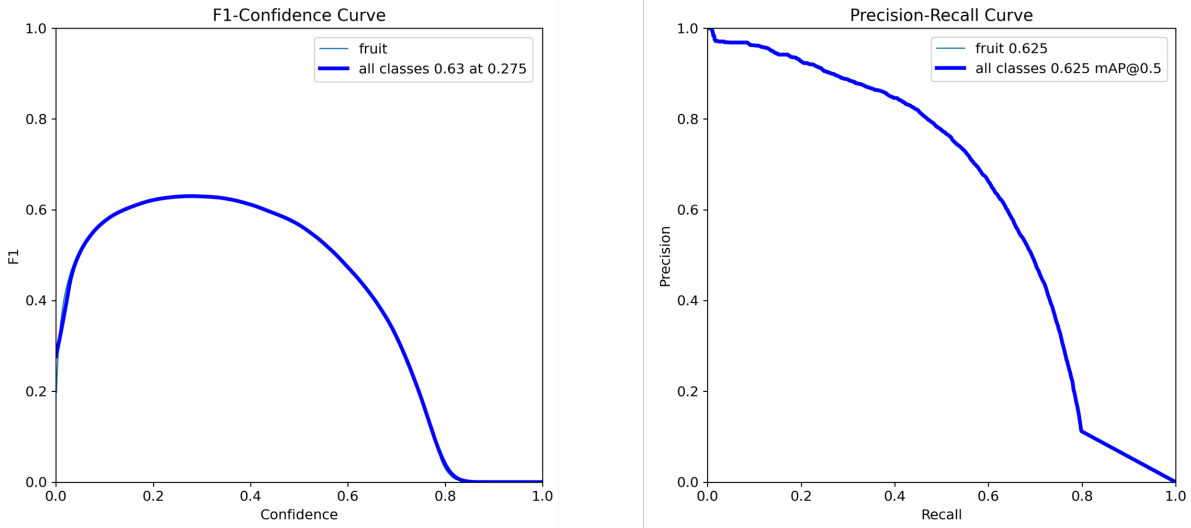


Figura 11: Comparación de las Curvas F1-Confidencia y Precisión-Recuperación de los modelos de detección de frutos DOD entrenados en MinneApple [2, 3, 4] y Apples [63] respectivamente, evaluados en MinneApple [2, 3, 4].