Análisis comparativo del rendimiento de modelos de detección de objetos para la identificación de elementos de protección personal en dispositivos de borde

Andrés Camilo Rodríguez Patarroyo

Juan Camilo Ávila Bermúdez

Universidad EAN

Facultad de Ingeniería

Maestría en Ciencia de Datos

Bogotá, Colombia

22/05/2026

Análisis comparativo del rendimiento de modelos de detección de objetos para la identificación de elementos de protección personal en dispositivos de borde

Andrés Camilo Rodríguez Patarroyo

Juan Camilo Ávila Bermúdez

Trabajo de grado presentado como requisito para optar al título de:

Magíster en Ciencia de Datos

Director (a):

Luz Maribel Guevara Ortega

Modalidad:

Monografía

Universidad EAN

Facultad de Ingeniería

Maestría en Ciencia de Datos

Bogotá, Colombia

22/05/2026

Nota de aceptación:

Firma del jurado

Firma del jurado

Firma del director del trabajo de grado

Bogotá, 22/05/2026

Agradecimientos

Queremos ofrecer nuestro más sincero agradecimiento a la ingeniera PhD. Luz Maribel Guevara Ortega, docente de la Facultad de Ingeniería, quien nos sirvió como guía durante todo el proceso de este estudio, así como al director del ecosistema de ciencia de datos de la Facultad de Ingeniería, Julián Fernando Muñoz Ordóñez, quien, con sus consejos, nos permitió madurar aún más las ideas de este estudio.

Resumen

La supervisión del uso adecuado de elementos de protección personal (EPP) continúa siendo un desafío crítico en los sectores industrial y de la construcción, especialmente en contextos con limitaciones de recursos tecnológicos. En los últimos años, la visión por computadora y el aprendizaje profundo han demostrado un alto potencial para automatizar esta tarea; sin embargo, la mayoría de las evaluaciones se han realizado en entornos de alto poder de cómputo.

Este trabajo de grado tiene como objetivo comparar el rendimiento de distintas arquitecturas de visión por computadora para la detección de EPP en dispositivos de borde de bajo costo, considerando métricas de precisión y eficiencia computacional. La investigación adopta un enfoque cuantitativo y experimental, en el que se entrenan y evalúan al menos tres modelos de detección de objetos sobre conjuntos de datos públicos. Las pruebas se ejecutan en condiciones controladas en un dispositivo de borde (Raspberry PI 5 con y sin NPU), midiendo métricas como mAP, velocidad de inferencia (FPS) y tamaño del modelo.

Los resultados permiten identificar diferencias significativas en el desempeño de las arquitecturas evaluadas, evidenciando compromisos entre precisión y eficiencia. Finalmente, se concluye que ciertas arquitecturas optimizadas presentan un equilibrio adecuado para su implementación en sistemas de bajo costo, y se aportan criterios técnicos para el despliegue de soluciones de seguridad industrial basadas en dispositivos de borde.

Palabras clave: Visión por computadora, detección de objetos, elementos de protección personal, dispositivos embebidos, dispositivos de borde, aprendizaje profundo.

Abstract

Monitoring the correct use of personal protective equipment (PPE) remains a critical challenge in industrial and construction environments, particularly in contexts with limited technological resources. In recent years, computer vision and deep learning have shown strong potential for automating this task; however, most evaluations have been conducted on high-performance computing platforms.

This thesis aims to compare the performance of different computer vision architectures for PPE detection on low-cost edge devices, considering both accuracy and computational efficiency metrics. The research follows a quantitative and experimental approach, in which at least three object detection models are trained and evaluated using public datasets. Experiments are conducted under controlled conditions on a low-cost edge device (Raspberry PI 5, with and without NPU), measuring metrics including mean Average Precision (mAP), inference speed (FPS) and model size.

The results reveal significant performance differences among the evaluated architectures, highlighting trade-offs between accuracy and efficiency. The study concludes that certain optimized architectures achieve a suitable balance for deployment on low-cost systems, providing technical guidance for implementing industrial safety solutions based on edge devices.

Keywords: Computer vision, object detection, personal protective equipment, embedded devices, edge devices, deep learning.

Contenido

Pág.

Lista de Figuras

Pág.

Lista de Tablas

Pág.

Introducción

La presente investigación se inscribe en el campo de la ciencia de datos aplicada a la visión por computadora, un área interdisciplinaria de la inteligencia artificial que combina técnicas de aprendizaje automático, procesamiento de imágenes y análisis computacional para la extracción de información relevante a partir de datos visuales. En particular, el estudio se alinea con la línea de investigación en Tecnologías de la Información y las Comunicaciones, abordando el diseño y la evaluación de modelos de detección de objetos orientados a mejorar la seguridad y la salud en el trabajo.

En los últimos años, la visión por computadora ha experimentado avances significativos gracias al desarrollo de arquitecturas de aprendizaje profundo, lo que ha permitido su aplicación en sectores como la manufactura, la construcción y la supervisión industrial, en estos sectores, el uso adecuado de elementos de protección personal (EPP) es un factor crítico para la prevención de accidentes laborales. (Spencer et al., 2019; Xu et al., 2020).

Diversos estudios, como en Sehsah et al. (2020), han demostrado que el uso adecuado de EPP reduce significativamente la incidencia y la severidad de las lesiones en entornos de alto riesgo. No obstante, la supervisión manual del cumplimiento normativo presenta limitaciones asociadas a la fatiga humana, a la cobertura espacial restringida y a la subjetividad en la evaluación. En respuesta a estas limitaciones, la literatura reciente ha explorado el uso de sistemas automatizados basados en visión por computadora para la detección de EPP, reportando resultados prometedores en términos de precisión y velocidad en entornos controlados (Xie et al., 2018; Cabrejos & Roman-Gonzalez, 2023; Velasquez et al., 2023).

A pesar de estos avances, una revisión crítica de los antecedentes empíricos y bibliográficos evidencia una brecha relevante: la mayoría de los modelos de detección de

objetos han sido entrenados y evaluados en plataformas de alto poder computacional, como servidores con GPU, lo que limita su adopción en contextos reales donde predominan restricciones de costo, energía y capacidad de procesamiento. Esta situación resulta particularmente relevante en países latinoamericanos, donde la implementación de soluciones tecnológicas de alto costo puede constituir una barrera para las pequeñas y medianas empresas (Schulz et al., 2024).

En el contexto colombiano, esta problemática adquiere mayor relevancia si se considera la persistencia de altas tasas de accidentalidad laboral, a pesar de la existencia de un marco normativo robusto como el de la Ley 1562 de 2012, el Decreto 1072 de 2015 y el Observatorio de Seguridad y Salud en el Trabajo en el que se regula el uso obligatorio de EPP.

El objeto de estudio de esta investigación se centra, por tanto, en responder la siguiente pregunta de investigación:

¿Cuáles son las diferencias estadísticamente significativas en el rendimiento y latencia de los modelos de detección de objetos basados en visión computacional para la identificación de elementos de protección personal al implementarlos en dispositivos de borde?

Esta pregunta busca identificar la solución más adecuada desde una perspectiva técnica y práctica, considerando las restricciones propias de los dispositivos de borde y los requerimientos mínimos para una implementación efectiva en entornos productivos reales. El problema central radica en la ausencia de estudios comparativos sistemáticos que analicen, bajo condiciones controladas, el equilibrio entre la precisión de detección y la eficiencia computacional de los modelos modernos de detección de objetos en este tipo de plataformas.

Si bien, de acuerdo con Redmon et al. (2015), S. Wang et al. (2024) y Peng et al. (2024), arquitecturas como YOLO, RF-DETR y DEIM han demostrado un desempeño

sobresaliente en términos de precisión y velocidad en entornos de alto cómputo, no existe consenso sobre su viabilidad técnica en escenarios con recursos limitados.

Para abordar esta problemática, este estudio se limita al análisis comparativo de los modelos derivados de las arquitecturas mencionadas en un dispositivo de borde de bajo costo como Raspberry PI 5, aplicados a la detección de los elementos de protección personal más utilizados en distintas industrias: cascos, máscaras, guantes y chalecos.

Por tanto, el mismo se estructura de la siguiente manera: en primer lugar, se presenta un marco teórico que contextualiza los conceptos fundamentales relacionados con la inteligencia artificial, el aprendizaje profundo, la visión por computadora, la detección de objetos y la normatividad asociada al uso de EPP.

Posteriormente, se describen la pregunta de investigación, las hipótesis y las variables, seguidas de un diseño metodológico de enfoque cuantitativo y experimental, en el que se detallan los modelos seleccionados, los dispositivos de prueba, las métricas de evaluación y los procedimientos de validación.

Finalmente, se presentan el trabajo de campo, las conclusiones y las contribuciones esperadas, destacando el impacto potencial del estudio en la industria, la sociedad y el desarrollo de soluciones aplicadas a la seguridad laboral.

Objetivos

Objetivo General

Analizar comparativamente el rendimiento de los modelos de detección de objetos basados en visión computacional para la identificación de elementos de protección personal (EPP), implementados en dispositivos de borde, mediante métricas de precisión y eficiencia computacional.

Objetivos Específicos

- Establecer un marco teórico sobre las arquitecturas de detección de objetos en visión computacional y su estado del arte, aplicables a la identificación de elementos de protección personal.
- Generar modelos de visión computacional basados en diferentes arquitecturas de detección de objetos para la identificación de elementos de protección personal.
- Analizar el rendimiento de los modelos en un dispositivo de borde en términos de precisión y eficiencia computacional.
- Determinar, mediante técnicas estadísticas, las diferencias en el rendimiento y la eficiencia entre los modelos analizados.

Justificación

La seguridad industrial constituye un pilar fundamental en entornos laborales de alto riesgo, donde el uso de elementos de protección personal (EPP) previene accidentes y salva vidas. Sin embargo, la supervisión humana de este cumplimiento presenta limitaciones inherentes: la fatiga visual, la subjetividad en la evaluación y la cobertura espacial limitada.

En este contexto, la visión por computadora emerge como una solución tecnológica prometedora, capaz de automatizar la detección de EPP mediante algoritmos de inteligencia artificial y de procesamiento de imágenes en tiempo real, mejorando los indicadores de accidentalidad y las cifras de mortalidad en el lugar de trabajo.

De acuerdo con Zhou et al. (2021), en los últimos años se han logrado avances significativos en el área de la visión por computadora gracias al desarrollo de técnicas novedosas que podrían mejorar el rendimiento de estos sistemas, por lo que, el desarrollo de un sistema de detección de EPP mediante visión por computadora, con características de eficiencia y bajo costo, se presenta como una solución atractiva e innovadora para mejorar la seguridad laboral en Colombia.

Estudios recientes demuestran avances significativos en modelos de detección de objetos, con arquitecturas de visión por computadora para identificación de elementos de protección personal, logrando precisiones superiores al 90% y velocidades de por encima de los 10 FPS para una cantidad baja de EPP y en dispositivos de procesamiento limitado (Cabrejos & Roman-Gonzalez, 2023; Velasquez et al., 2023).

Sin embargo, también persisten otros desafíos técnicos: la variabilidad lumínica, las oclusiones parciales de los EPP y la necesidad de operar en hardware embebido con restricciones computacionales.

Un sistema de este tipo funcionaría mediante la instalación de cámaras en áreas estratégicas de las instalaciones laborales, que capturarían imágenes o videos del entorno de trabajo, estas imágenes serían procesadas por algoritmos de visión por computadora capaces de identificar si los trabajadores utilizan adecuadamente los EPP, como cascos, guantes, chalecos reflectantes, entre otros. En caso de detectar incumplimientos, el sistema podría generar alertas en tiempo real para que se tomen las medidas correctivas necesarias de manera inmediata o bien servir como indicadores para posteriores acciones de mejora y auditoría.

Adicionalmente y de acuerdo con Khoshakhlagh et al. (2024), la implementación de este tipo de sistemas aportaría múltiples beneficios: en primer lugar, podría fortalecer la cultura de seguridad dentro de las organizaciones, promoviendo el uso constante y adecuado de los EPP. En segundo lugar, permitiría una supervisión más eficiente y menos intrusiva, ya que el monitoreo automatizado reduciría la necesidad de inspecciones manuales constantes. Además, al identificar y corregir rápidamente las infracciones, se reduciría la incidencia de accidentes laborales y, por consiguiente, los costos asociados, como indemnizaciones, pérdidas de productividad y gastos médicos.

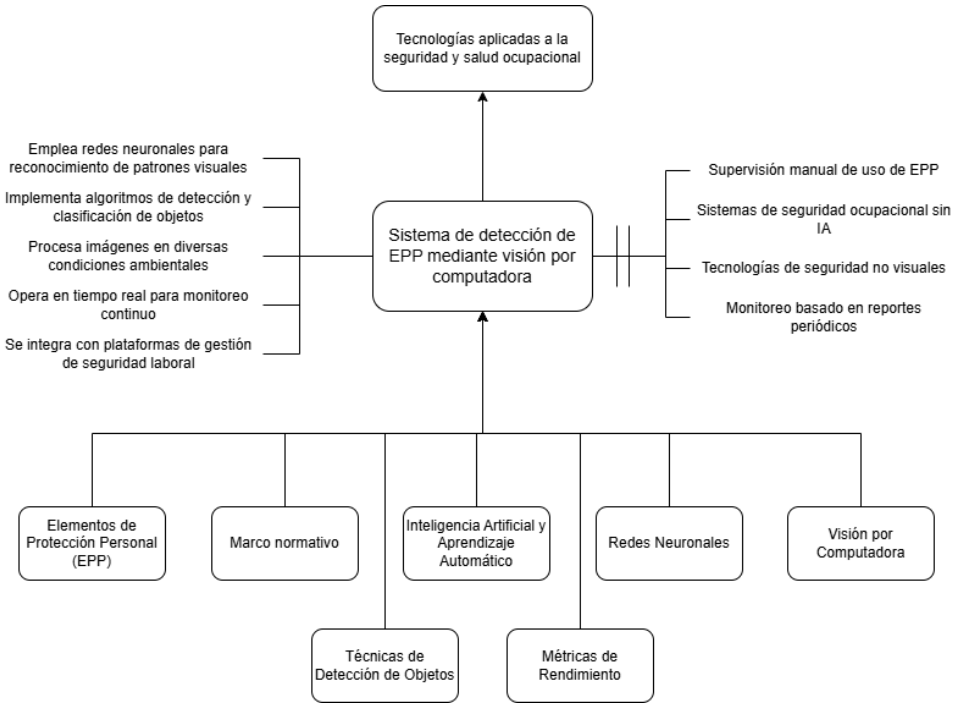
Por lo tanto, en esta investigación se propone seleccionar los elementos de protección personal más comunes —cascos, máscaras, guantes y chalecos— para su uso en el entrenamiento de un algoritmo, una arquitectura o un modelo de visión por computadora que permita validar cuantitativamente la viabilidad de implementar este tipo de sistemas en futuros proyectos en un ambiente productivo y bajo las condiciones reales de trabajo.

Marco Teórico

En esta investigación es imprescindible una comprensión básica tanto de la perspectiva técnica del problema como de los marcos normativos que regulan el uso de los EPP, tanto para diseñar criterios de validación del uso correcto como para delimitar el alcance legal del sistema.

Figura 1

Mentefacto conceptual del proyecto



Nota. El mentefacto presentado sintetiza de manera efectiva el marco teórico y destaca las relaciones jerárquicas entre los distintos componentes del sistema de detección automatizada de EPP mediante IA. Elaboración propia.

En la **Figura *1*** se presenta una visión estructurada y completa del marco teórico de esta investigación, lo que facilita la comprensión de las relaciones entre los distintos conceptos involucrados en el desarrollo de un sistema de detección automatizada de

EPP. La organización jerárquica permite identificar con claridad el posicionamiento del sistema dentro de un contexto más amplio, sus características definitorias, los elementos que lo diferencian de otros enfoques y los componentes técnicos y normativos que lo constituyen.

Elementos de Protección Personal (EPP)

Un Elemento de Protección Personal (EPP), según la legislación colombiana y los lineamientos del Ministerio de Salud y Protección Social (2024), es cualquier equipo, accesorio o dispositivo que debe ser utilizado por el trabajador con el fin de protegerlo contra uno o varios riesgos que puedan amenazar su seguridad o salud en el trabajo.

Desde el enfoque del Ministerio de Salud y Protección Social (2024), los EPP son considerados parte fundamental del Sistema de Gestión de Seguridad y Salud en el Trabajo (SG-SST) de Colombia, y su entrega, uso y mantenimiento adecuados son responsabilidad tanto del empleador como del trabajador.

Elementos Básicos de Protección Personal

Los elementos de protección personal (EPP) más comunes en los entornos laborales colombianos, según la normativa y las recomendaciones del Ministerio de Salud y Protección Social (1979), son aquellos que protegen las principales partes del cuerpo frente a riesgos frecuentes en la industria, la construcción, la salud y otros sectores.

Figura 2

Elementos de protección personal

Nota. Elementos de protección personal más comunes en entornos laborales. Elaboración propia.

Normatividad legal existente

La legislación colombiana sobre el uso de Elementos de Protección Personal (EPP) establece obligaciones claras para empleadores y trabajadores, con el objetivo de salvaguardar la salud y seguridad en el entorno laboral. Estas normas exigen identificar los riesgos presentes en cada actividad, seleccionar y suministrar los EPP adecuados, sin costo para el trabajador, y garantizar la capacitación sobre su uso y mantenimiento.

Tabla 1

Leyes, normas y decretos relacionados con el uso de EPP

Legislación	Descripción
Ley 9 de 1979	Considerada la base de la salud ocupacional en Colombia, obliga a los empleadores a proporcionar EPP adecuados y suficientes,

	ajustados a los riesgos laborales, y a cumplir con las normas técnicas oficiales.
Decreto 1072 de 2015	Compila y organiza toda la normativa laboral, consolidando la obligación de identificar riesgos, suministrar EPP certificados y garantizar su uso correcto como parte del SG-SST.
Resolución 2400 de 1979	Regula las condiciones de higiene y seguridad en los lugares de trabajo, incluyendo disposiciones específicas sobre el uso de EPP.
Resolución 0312 de 2019	Define los estándares mínimos del SG-SST, incluyendo la gestión y el control de los EPP.

Nota. Esta tabla presenta un resumen general, de algunas de las leyes, decretos y resoluciones expedidos actualmente, relacionados con el uso de elementos de protección personal en Colombia. Elaboración propia.

Inteligencia Artificial (IA)

La inteligencia artificial (IA) ha surgido como uno de los campos más transformadores de la informática, con profundas implicaciones en numerosas disciplinas. De acuerdo con Russell y Norvig (2021), autores del libro de texto de IA más utilizado: "Artificial Intelligence: A Modern Approach", en donde abordan la IA desde múltiples perspectivas, los autores clasifican los sistemas de IA como aquellos diseñados para pensar racionalmente, actuar racionalmente, pensar como humanos o actuar como humanos.

Aprendizaje Automático (Machine Learning)

El aprendizaje automático (*Machine Learning* [ML]) es un subcampo de la inteligencia artificial (IA) en el que se utilizan técnicas estadísticas para identificar patrones dentro de conjuntos de datos, permitiendo que los sistemas realicen predicciones o tomen decisiones basadas en datos nuevos y no vistos previamente (França et al., 2021).

Aprendizaje Supervisado

Según Sarker (2021) y numerosos autores en la materia, el aprendizaje automático abarca varios paradigmas principales, definidos por la naturaleza de los datos y por el proceso de aprendizaje. El tipo de aprendizaje aplicable a esta investigación es el aprendizaje supervisado, en el cual, de acuerdo con Sarker, los modelos se entrenan con conjuntos de datos etiquetados, en los que se conocen tanto las variables de entrada como las de salida. Los algoritmos aprenden a mapear las entradas a las salidas, lo que los hace adecuados para tareas como la clasificación y la regresión.

Aprendizaje Profundo (Deep Learning)

El aprendizaje profundo (*Deep Learning* [DL]) es una subespecialidad del aprendizaje automático que emplea redes neuronales artificiales con múltiples capas para modelar relaciones complejas en los datos (Wan, 2023).

De acuerdo con Wan, los modelos de aprendizaje profundo, como los perceptrones multicapa (MLP) y las redes neuronales convolucionales (CNN), han demostrado un éxito notable en áreas como el reconocimiento de imágenes, el procesamiento del lenguaje natural y el análisis del habla humana, debido a su capacidad para aprender representaciones jerárquicas de los datos.

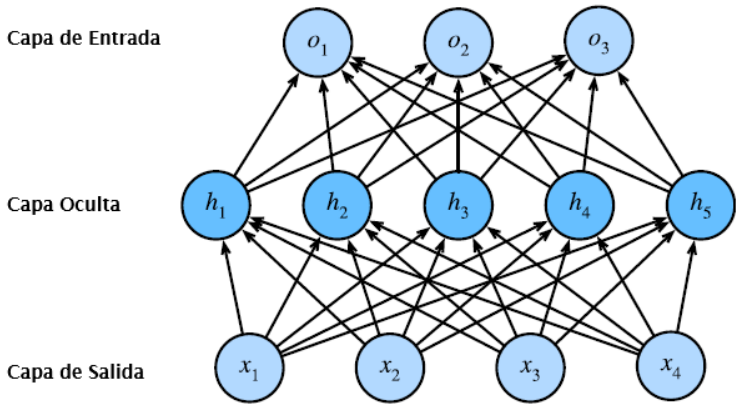
Categorías de Redes Neuronales

Tal como expone Zhang et al. (2021) expone con precisión en un reciente estudio, los principales tipos de redes neuronales que podemos encontrar en el estado del arte (SOTA). Se resumen a continuación las diferentes categorías que podemos encontrar en este estudio:

Redes Neuronales Feedforward (FNN)/Perceptrones Multicapa (MLP). De acuerdo con Chan et al. (2023), una MLP está compuesta por múltiples capas de neuronas interconectadas, que generalmente incluyen una capa de entrada, una o más capas ocultas y una capa de salida. La forma más simple en la que los datos fluyen en una sola dirección, desde la entrada hasta la salida, a través de una o varias capas ocultas. Se usan para tareas básicas de clasificación y regresión (Chan et al., 2023).

Figura 3

Red neuronal multicapa

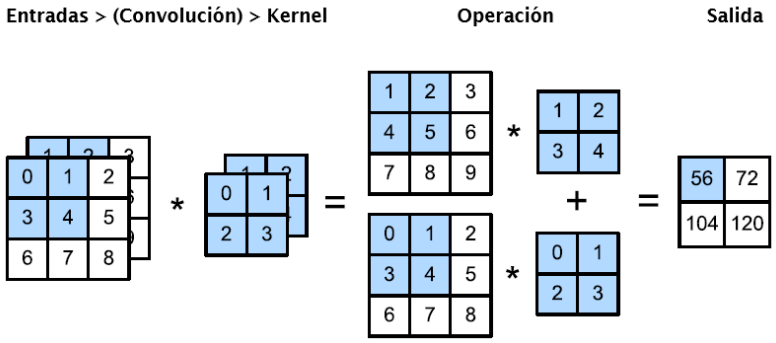


Nota. Diagrama de composición de una red neuronal multicapa, contiene capas intermedias de neuronas entre la capa de entrada y la capa de salida. Adaptado de Dive into Deep Learning (p. 172) por Zhang A. et al., 2021, *Journal of the American College of Radiology*.

Redes Neuronales Convolucionales (CNN). Especializadas en procesar datos no estructurados, como imágenes, la idea central detrás de las CNN es dividir la imagen de entrada en pequeños segmentos o parches, y procesar cada parche utilizando operaciones idénticas. Cada parche es analizado por un módulo clasificador, que, a su vez, es una red neuronal. En la práctica, esta red neuronal recorre la imagen, clasificando cada parche conforme avanza (Torralba et al., 2024).

Figura 4

Operación de una red neuronal convolucional

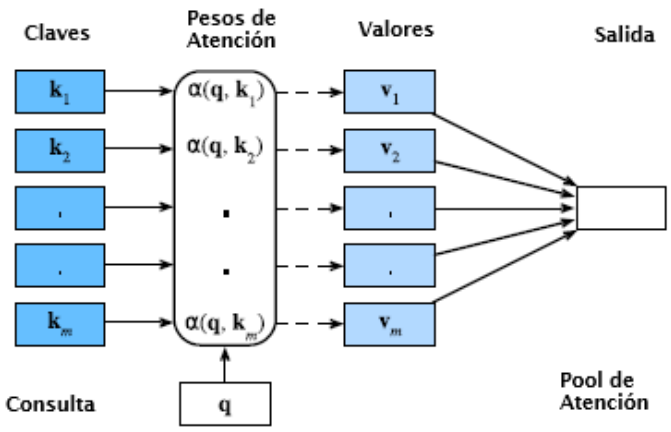


Nota. Funcionamiento general de una red neuronal convolucional, los datos de entrada generalmente se apilan en filas y columnas. Adaptado de Dive into Deep Learning (p. 247) por Zhang A. et al., 2021, *Journal of the American College of Radiology*.

Redes Transformer. Utilizan mecanismos de atención para procesar datos secuenciales de manera eficiente, emplean pesos que se ajustan de acuerdo con la importancia de cada elemento de la entrada, siendo actualmente el estado del arte en procesamiento de lenguaje natural y cada vez más en visión por computadora (Zhang et al., 2021).

Figura 5

Mecanismo de atención de una red transformer



Nota. El mecanismo de atención de una red transformer, utiliza múltiples elementos de procesamiento que son usados para ajustar unos pesos que indican la importancia de cada componente de entrada. Adaptado de Dive into Deep Learning (p. 426) por Zhang A. et al., 2021, *Journal of the American College of Radiology*.

Visión por Computadora (Visión Artificial)

La visión por computadora es un campo interdisciplinario dentro de la informática y la inteligencia artificial que se centra en permitir que las computadoras interpreten, procesen y comprendan la información visual del mundo, como imágenes y videos. El objetivo principal de la visión por computadora es replicar las capacidades perceptuales de la visión humana, permitiendo que las máquinas extraigan información significativa de datos visuales y tomen decisiones o realicen tareas basadas en esa información (Chen et al., 2024).

Reconocimiento y Detección de Objetos

El reconocimiento de objetos es el proceso de identificar objetos asignándoles etiquetas o nombres a partir de información visual. Implica clasificar los objetos en instancias o clases, y puede extenderse a reconocer acciones y actividades utilizando varios modelos de aprendizaje en Visión por Computadora (Khanday & Sofi, 2021).

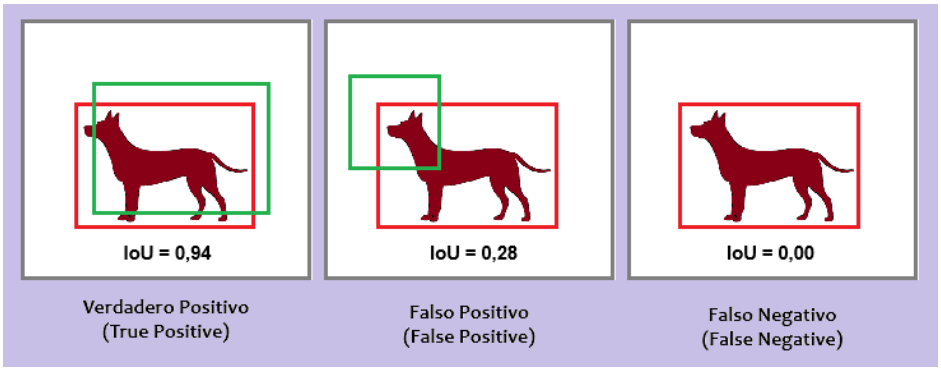
Detección de Objetos. La detección de objetos es un área de la visión por computadora cuyo objetivo es localizar objetos en imágenes digitales combinando subtareas de localización y clasificación para estimar simultáneamente la ubicación y el tipo de instancias de objetos en una o más imágenes, es una rama de inteligencia artificial que busca permitirle a las computadoras reconocer y clasificar objetos de acuerdo con categorías semánticas (Bogusław Cyganek, 2013).

Métricas de Rendimiento en Detección de Objetos

Intersección sobre Unión (IoU). La intersección sobre unión (IoU) es una métrica de evaluación común utilizada en los modelos de detección de objetos. Un cuadro delimitador es la salida rectangular que delimita un objeto detectado según lo predicho por el modelo. Calcula la relación de múltiples cajas delimitadoras de la intersección en el área de las secciones superpuestas de las cajas sobre su área de unión, es decir, área total de ambas cajas combinadas (Szeliski, 2022).

Figura 6

Tipos de casos para la clasificación por IoU



Nota. Descripción visual de la clasificación de casos para la Intersección sobre la Unión (IoU). Elaboración propia.

La fórmula de la Intersección sobre la Unión se expresa como:

$$IOU = \frac{area\ of\ equation}{area\ of\ union}$$

Precisión. La precisión mide el porcentaje de predicciones correctas respecto de todas las observaciones (o detecciones). Se calcula como el número de verdaderos

positivos (TP) sobre el número de TP más el número de falsos positivos (FP) **(Powers & Ailab, 2020)**:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Indica qué porcentaje de las detecciones del modelo son correctas al identificar objetos en imágenes.

Recall (Sensibilidad). La Recall o sensibilidad mide las predicciones correctas con respecto a la todas las clases de los datos reales **(Powers & Ailab, 2020)**.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

En resumen, mide la capacidad del modelo para identificar correctamente todos los objetos reales presentes y muestra qué porcentaje de los positivos reales fue detectado.

Promedio de Precisión (AP). Mide el área bajo la curva de precisión-recall para una clase específica. Se calcula promediando la precisión en cada punto donde se detecta un objeto relevante, reflejando tanto la precisión como la recuperación del modelo en detección de objetos **(Kaur & Singh, 2023)**. La precisión media se expresa matemáticamente como:

$$AP = \int_0^1 p(r)dr$$

Donde p es la Precisión y r es el Recall.

Para calcular esta integral se pueden utilizar varios métodos de aproximación, sin embargo, los métodos más utilizados actualmente son: el método de interpolación de 101 puntos con integración trapezoidal recomendado por el equipo del conjunto de datos Microsoft COCO (Lin et al., 2014) y el método de interpolación de todos los puntos con integración por área bajo la curva (AUC).

Media de los Promedios de Precisión (Mean Average Precision, mAP). La Media de los Promedios de Precisión (mAP) es una métrica común utilizada para evaluar la precisión de un modelo de detección de objetos y es la media de los Promedios de Precisión (AP) sobre un lote de muestras **(Kaur & Singh, 2023)**. La Media de los Promedios de Precisión se evalúa de la siguiente manera:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

Donde AP_k es el AP de la clase k y n es el número de clases.

Arquitecturas para la Detección de Objetos

En la actualidad, las arquitecturas de detección de objetos en tiempo real han evolucionado considerablemente para satisfacer demandas de alta precisión y velocidad. YOLO (You Only Look Once), desarrollado inicialmente en 2015 (Redmon et al., 2015), revolucionó el campo al procesar imágenes completas en una sola pasada, alcanzando con YOLOv8 velocidades de hasta 155 FPS.

RT-DETR, basado en redes transformers, representa un avance significativo al eliminar el posprocesamiento de las imágenes, logrando 48,1% AP mientras mantiene

latencia competitiva (S. Wang et al., 2024). DEIM combina lo mejor de ambos enfoques, implementando emparejamiento denso y refinamiento de distribución granular, alcanzando hasta 56,5% AP (S. Wang et al., 2024). Estas arquitecturas establecen nuevos estándares para aplicaciones como la vigilancia, los vehículos autónomos y el análisis de video en tiempo real.

Arquitectura ViT (Vision Transformer), es una arquitectura de red neuronal que aplica directamente el bloque Transformer, originalmente propuesto para NLP, al dominio de imágenes, tratándolas como secuencias de parches en lugar de usar convoluciones.

ViT fue introducido por Dosovitskiy et al. (2021) bajo la idea de que “una imagen vale 16×16 palabras”: se divide la imagen en parches de tamaño fijo (por ejemplo, 16×16 píxeles), y cada parche se procesa como si fuera un “token” en un modelo Transformer. El modelo utiliza un *encoder* Transformer puro (sin *decoder*) para tareas de clasificación de imágenes, y ha demostrado que, con suficientes datos de entrenamiento, puede igualar o superar a arquitecturas CNN profundas tradicionales.

Arquitectura YOLO (You Only Look Once). Es una familia de modelos de detección de objetos en tiempo real que definen la detección como un único problema de regresión, desde los píxeles de la imagen hasta las coordenadas de los cuadros delimitadores y las probabilidades de clase. Procesa la imagen completa en una sola pasada, lo que la hace rápida y eficiente para aplicaciones en tiempo real **(Bochkovskiy et al., 2020; Khanam & Hussain, 2024a, 2024b; Li et al., 2022; A. Wang et al., 2024; C. Wang et al., 2022; Yaseen, 2024; Tian Y. et al, 2025; Sapkota R. et al, 2026)**.

Tabla 2

Comparación de versiones de la arquitectura YOLO

Versión	Tronco (Backbone)	Uso de Anclaje	mAP	mAP50-95	FPS
YOLO v4	CSPDarknet53	Sí	65,7%	43,5%	~65
YOLO v5x	CSPDarknet + SPPF	Sí	68,9%	50.7%	~200
YOLO v6-L	EfficientRep	Sí	70,0%	52,3%	~121
YOLO v7	E-ELAN	Sí	71,2%	53,1%	~30
YOLO v8x	CSP + PAN/FPN	No	71,5%	~53%	~87
YOLO v9e	GELAN	No	-	55,6%	-
YOLO v10x	CSPNet	No	-	54,4%	~93
YOLO v11x	C3K2	No	-	54,7%	~88,5
YOLO v12x	C3K2	No	72,0%	55,2%	~84,8
YOLO v26x	C3K2	No	-	57,5%	~84,7

Nota. Se muestra una comparación de las versiones principales de la familia de modelos YOLO, desde la v4 hasta la v26. Elaboración propia.

En la **Tabla 2** se presenta un resumen de métricas para el conjunto de datos Microsoft COCO de las diferentes versiones de la familia de modelos de acuerdo con cada uno de sus autores (Lin et al., 2014).

Como se puede ver en C. Wang y Liao (2024), las arquitecturas entre las versiones de YOLO varían en numerosos aspectos de versión a versión, sin embargo, la versión más reciente y eficiente del conjunto en la actualidad es YOLO v26, la cual comparte la mayoría de las características con YOLO v12 y esta, a su vez, con YOLO v11.

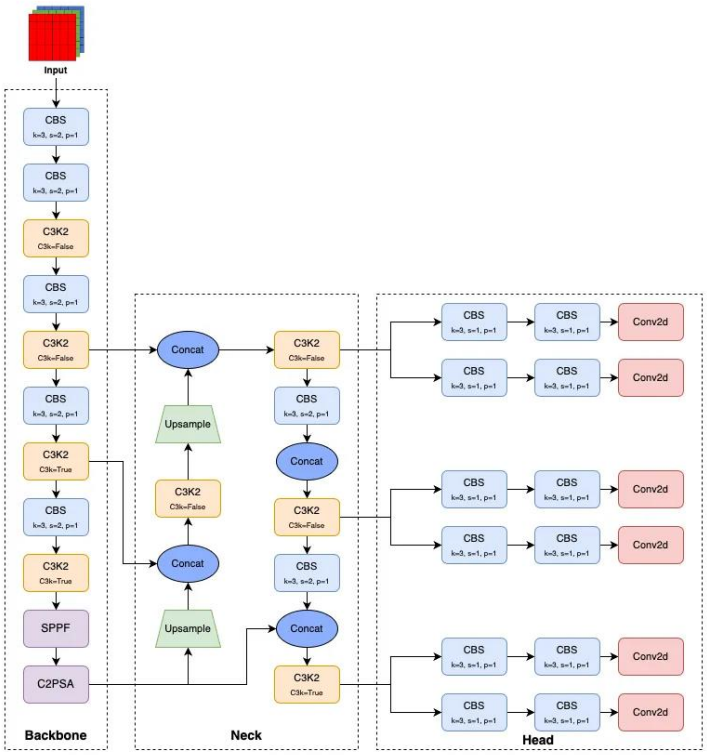
De acuerdo con Tian et al. (2025), en YOLO v12 se introduce un módulo de atención más simple y eficiente. Otro cambio destacable es la inclusión de redes de agregación eficientes de residuales, lo que optimiza el rendimiento de los modelos a gran escala.

Finalmente, se implementaron múltiples mejoras arquitecturales orientadas a optimizar el rendimiento y mejorar la eficiencia computacional.

Para la versión más reciente, YOLO v26, también se implementaron algunas mejoras orientadas a mejorar en simplicidad y eficiencia (Sapkota et al., 2026), reduce los tiempos de inferencia, cuellos de botella debido al posprocesamiento y simplifica la exportación y mejora la compatibilidad de hardware para ONNX, TensorRT, CoreML y TFLite, sin embargo, una mejora para destacar es la inclusión del optimizador híbrido entre Muon y SGD: MuSGD (Jordan et al., 2024).

Figura 7

Arquitectura de YOLO v11



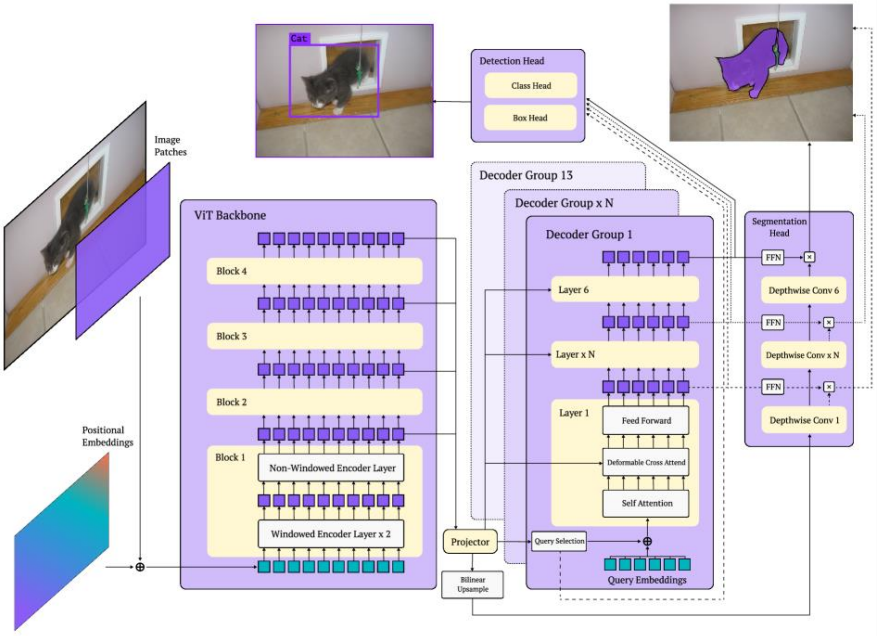
Nota. Arquitectura de YOLO v11, se muestran los diferentes bloques que componen el Tronco, Cuello y Cabeza. Tomado de “Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors” por Jegham et al., 2024, arXiv:2411.00201v4.

Arquitectura RF-DETR (Roboflow Detection Transformer), es una arquitectura de detección de objetos basada en Transformers, diseñada para lograr un equilibrio óptimo entre precisión y latencia en aplicaciones en tiempo real. Surge como una evolución de la familia DETR (*Detection Transformer*), incorporando mejoras estructurales y técnicas de optimización que permiten una mayor eficiencia computacional sin sacrificar el rendimiento.

RF-DETR representa una alternativa práctica de los detectores basados en Transformers, combinando la formulación de DETR con mejoras arquitectónicas orientadas a la eficiencia y aplicaciones en tiempo real. Se posiciona como una alternativa competitiva frente a modelos de CNN como YOLO, especialmente cuando el modelado global y la simplicidad estructural son prioritarios.

Figura 8

Arquitectura de RF-DETR



Nota. Arquitectura de RF-DETR, se muestra el backbone basado en ViT y los componentes de búsqueda neural (NAS). Tomado de “RF-DETR: Neural Architecture Search for Real-Time Detection Transformers” por Robinson I. et al., 2026, arXiv:2511.09554.

A continuación, se presenta la tabla de métricas de detección de objetos de la arquitectura RF-DETR para el conjunto de datos Microsoft COCO en distintos tamaños.

Tabla 3

Métricas de rendimiento RF-DETR

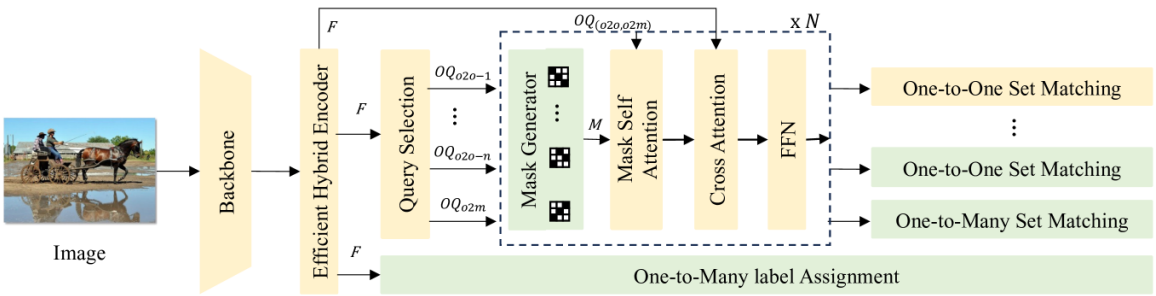
Modelo	mAP 50-95	FPS
RF-DETR-N	48,4%	~434,8
RF-DETR-S	53,0%	~285,7
RF-DETR-M	54,7%	~227,3
RF-DETR-L	56,5%	~147,1
RF-DETR-XL	58,6%	~87,0
RF-DETR-2XL	60.1%	~58,1

Nota. Métricas de rendimiento según el tipo de modelo, N (*nano*), S (*small*), M (*medium*), L (*large*), X (*extra large*). Elaboración propia.

Arquitectura RT-DETR v3, es un detector basado en redes transformer. Implementa una arquitectura basada en RT-DETR (*Real-Time Detection Transformer*), la cual refina algunos aspectos de la arquitectura original, se enfoca en mejorar algunas limitaciones que tenía la supervisión dispersa (*sparse supervision*) a través de varias estrategias sin reducir el rendimiento, como se puede detallar en la **Figura 9**. De acuerdo con S. Wang et al. (2024) y Zhao et al., (2023).

Figura 9

Arquitectura de RT-DETRv3



Nota. Arquitectura RT-DETRv3 basada en redes transformer con mecanismos de atención. Tomado de “RT-DETRv3: Real-time End-to-End Object Detection with Hierarchical Dense Positive Supervision” por Wang et al., arXiv:2409.08475v3.

Arquitectura DETR (DEtection TRansformer), introdujo un cambio de paradigma en la detección de objetos al utilizar una arquitectura basada en transformers con emparejamiento bipartito. Aunque revolucionario, de acuerdo con *S. Li et al. (2023)*, DETR presenta algunas limitaciones que motivaron a los investigadores a desarrollar sistemas mejorados, como D-FINE y DINO, manteniendo las capacidades de detección de extremo a extremo de DETR.

Arquitectura DINOv3, es la tercera generación de la familia DINO (Distillation with NO labels), un marco de aprendizaje auto-supervisado (Self-Supervised Learning, SSL) para Vision Transformers (ViT). Su objetivo es escalar el aprendizaje visual sin etiquetas hacia modelos de gran capacidad, mejorando la estabilidad, la semántica emergente y la transferibilidad en tareas de aplicación específica. Mantiene una arquitectura ViT relativamente simple, pero optimiza profundamente el régimen de entrenamiento de destilación *teacher–student* para producir representaciones visuales de alta calidad y altamente transferibles (Siméoni et al., 2025), consolidándose como uno de los estándares actuales en preentrenamiento auto-supervisado para visión por computadora.

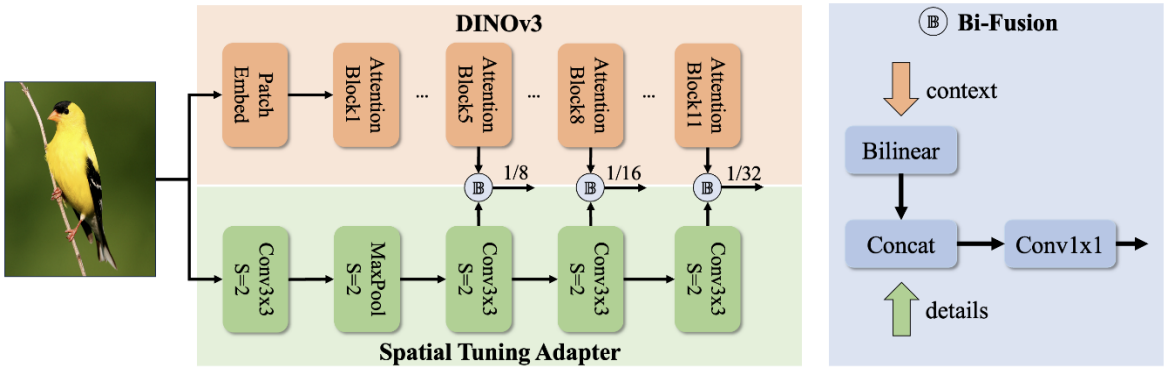
Arquitectura DEIM (DETR with Improved Matching), es un marco de entrenamiento diseñado específicamente para acelerar la convergencia en modelos de detección de objetos basados en transformers, abordando una de las principales debilidades de DETR. Según Huang et al. (2024).

Arquitectura DEIMv2, es una familia de detectores en tiempo real basada en el marco DEIM (*DETR with Improved Matching*), que integra backbones preentrenados con DINOv3. Su objetivo principal es ofrecer un mejor compromiso entre precisión y costo computacional que los detectores basados en YOLO y los DETR tradicionales, y abarcar desde GPUs de alto rendimiento hasta dispositivos edge y móviles. La serie incluye ocho tamaños de modelo (X, L, M, S, Nano, Pico, Femto y Atto), lo que permite adaptar la arquitectura a distintos presupuestos de parámetros y FLOPs.

En términos generales, la arquitectura de DEIMv2 sigue el patrón de los detectores tipo DETR, compuesta por tres bloques principales: backbone, módulo de adaptación de características y decoder de detección. No obstante, introduce modificaciones específicas en cada bloque para aprovechar las representaciones de DINOv3 (Huang et al., 2026).

Figura 10

Arquitectura de DEIMv2



Nota. Arquitectura DEIMv2, donde se muestra la integración de DINOv3. Tomado de “Real-Time Object Detection Meets DINOv3” por Huang S. et al., arXiv:2509.20787.

Para los modelos ultraligeros (Nano, Pico, Femto y Atto), DEIMv2 reemplaza el backbone de ViT por HGNetv2-B0 y aplica una estrategia de poda progresiva tanto en profundidad como en anchura. Por ejemplo, DEIMv2-Pico elimina la cuarta etapa de

HGNetv2-B0, DEIMv2-Femto reduce el número de bloques en la última etapa, y DEIMv2-Atto disminuye la dimensionalidad de canales del bloque final, con el objetivo de ajustarse a presupuestos extremadamente restringidos de parámetros.

Tabla 4

Métricas de rendimiento DEIMv2

Modelo	mAP 50-95	FPS
DEIM-Atto	23,8%	~909,0
DEIM-Femto	31,0%	~689,7
DEIM-Pico	38,5%	~469,5
DEIM-N	43,0%	~431,0
DEIM-S	50.9%	~173,0
DEIM-M	53,0%	~113,6
DEIM-L	56,0%	~95,5
DEIM-X	57,8%	~72,7

Nota. Métricas de rendimiento según el tipo de modelo, *Atto*, *Femto*, *Pico*, N (*nano*), S (*small*), M (*medium*), L (*large*), X (*extra large*). Elaboración propia.

Hipótesis

Hipótesis sobre la Precisión (H_1)

El rendimiento de los modelos de detección de objetos, medido a través de la métrica mAP@0.5:0.95, presenta variaciones entre las arquitecturas evaluadas cuando son implementadas en un dispositivo de borde de bajo costo. Dichas variaciones no solo implican posibles diferencias estadísticamente significativas, sino que también pueden caracterizarse en términos de magnitud, dispersión y relevancia práctica.

Hipótesis nula (H_{01})

No existen diferencias estadísticamente significativas en el rendimiento (mAP@0.5:0.95) entre los modelos evaluados.

Hipótesis alternativa (H_{11})

Existen diferencias estadísticamente significativas en el rendimiento (mAP@0.5:0.5:0.95) entre los modelos evaluados, las cuales pueden ser cuantificadas en términos de tamaño del efecto y variabilidad.

Hipótesis sobre latencia (H_2)

Existen diferencias en el tiempo de inferencia (latencia) entre los modelos de detección de objetos evaluados al ejecutarse en un dispositivo de borde de bajo costo.

Hipótesis nula (H_{02})

No existen diferencias estadísticamente significativas en la latencia de inferencia entre los modelos evaluados.

Hipótesis alternativa (H_{12})

Existen diferencias estadísticamente significativas en la latencia de inferencia entre los modelos evaluados.

Variables

La definición conceptual y operacional de variables constituye un proceso metodológico esencial para validar empíricamente una hipótesis científica, ya que establece un puente entre los constructos teóricos y su manifestación observable en la realidad. Este doble nivel de especificación garantiza que los conceptos abstractos inherentes a la investigación sean traducidos a términos medibles, replicables y libres de ambigüedades, permitiendo así contrastar de manera rigurosa las relaciones postuladas entre las variables (J. L. Arias, 2020; Hernández-Sampieri & Mendoza, 2018; Tamayo y Tamayo, 2003). En la **Tabla *5*** se presentan las variables de investigación identificadas durante el desarrollo de la investigación.

Tabla 5

Variables de Investigación

Tipo	Nombre	Definición Conceptual	Definición Operacional
Variables Independientes	Arquitectura del modelo	Es la organización de la red neuronal que utiliza múltiples capas de procesamiento para aprender patrones complejos a partir de datos, mediante el uso de otras capas más simples. Estas capas construyen una jerarquía de abstracciones, donde cada nivel se basa en los inferiores. (Kala, 2024)	Las arquitecturas que ofrecen buenas prestaciones, algunas de ellas consideradas estado del arte son: • YOLO • RF-DETR • DEIMv2
	Técnica de optimización	Los modelos desplegados en dispositivos de bajo costo suelen tener un	Las técnicas de optimización más utilizadas son las siguientes:

Tipo	Nombre	Definición Conceptual	Definición Operacional
		impacto en el rendimiento debido a las limitaciones de hardware, las técnicas de optimización ayudan a resolver algunos de estos problemas sin reducir demasiado sus prestaciones (Jiang et al., 2023).	• Ajuste de hiperparámetros (Raiaan et al., 2024). • Poda (*pruning*) (E. Chen et al., 2025). • Cuantización (Jiang et al., 2023).
	Fuente de datos	Hace referencia al origen o a la ubicación de la que se obtiene una colección o conjunto de datos para fines de análisis o investigación, como registros empresariales, bases de datos públicamente disponibles o instituciones de investigación.	• Conjunto local: conjunto de datos preprocesado para el entrenamiento de modelos de ML. • Conjunto sintético: conjunto de datos generado usando técnicas de aumento de datos (*data augmentation*) o distorsiones que ayudan a mejorar la precisión de los modelos (Mumuni & Mumuni, 2022).
Variables Dependientes	Eficiencia económica	Busca obtener los mejores resultados para resolver el problema con la menor cantidad de recursos posible, sin exceder un umbral establecido como óptimo (Brue & Grant, 2007).	Se establece como umbral óptimo de eficiencia: 70% del salario mínimo legal vigente (SMLV) colombiano, teniendo en cuenta que es un indicador directamente relacionado con la inflación y el costo de los servicios (Avila-Montealegre et al., 2021).

Tipo	Nombre	Definición Conceptual	Definición Operacional
	Rendimiento computacional	Se refiere a las características cuantificables de un sistema informático, con énfasis en métricas como el rendimiento, el uso de recursos y los tiempos de respuesta. Su propósito es asegurar que el sistema funcione de manera eficiente y cumpla adecuadamente con los requerimientos de las aplicaciones del usuario (Fortier & Michel, 2003).	Se mide mediante métricas de rendimiento como (Kaur & Singh, 2023): • Velocidad en FPS o ms. • Uso de memoria. • Precisión. • Sensibilidad (*recall*). • AP y mAP.
Variables de Control	Especificaciones de hardware	Son las características de hardware de los dispositivos en los que se miden las variables involucradas.	Evaluación en diferentes tipos de hardware: • Equipo de escritorio o laptop. • Raspberry PI. • Raspberry PI con módulo de NPU.
	Condiciones operativas	Se refiere a las condiciones ambientales empleadas al realizar las mediciones.	Evaluación en condiciones de iluminación óptima en exteriores e interiores con una distancia operativa de 1,5 a 4 metros.

Nota. Tabla de variables de investigación que presenta las definiciones conceptuales y operacionales de cada variable. Elaboración propia.

Metodología

Tipo de Investigación

En base a Hernández-Sampieri y Mendoza (2018), la presente investigación adopta un enfoque cuantitativo y experimental, dado que se desarrollan y evalúan modelos de inteligencia artificial basados en visión computacional para identificar elementos de protección personal (EPP) en imágenes y video. La cuantificación del rendimiento de los modelos se realiza mediante métricas objetivas como fotogramas por segundo (FPS), latencia (ms), precisión (mAP), sensibilidad (recall) y consumo de memoria (MiB), que permiten validar las hipótesis establecidas.

Diseño de Investigación

De acuerdo con F. G. Arias (2012), se emplea un diseño experimental de carácter tecnológico, basado en el desarrollo y evaluación de diferentes arquitecturas de modelos de detección de objetos, específicamente:

- **Familia YOLO**, se analizan las versiones YOLO v8, YOLO v11, YOLO v12, YOLO v26, todas en sus tamaños: *nano*(n) y *small*(s)
- **Familia RF-DETR**, se analiza la versión RF-DETR v3, en sus tamaños: *nano*(n) y *small*(s)
- **Familia DEIM**, se analiza la versión DEIM v2, en sus tamaños: *nano*(n) y *small*(s)

Las imágenes utilizadas para la generación de estos modelos se dividen en los siguientes subconjuntos:

- **Entrenamiento**, compuesto por un 70% de las imágenes
- **Validación**, compuesto por un 20% de las imágenes
- **Pruebas**, compuesto por un 10% de las imágenes

Esto permite comparar los resultados obtenidos en las distintas configuraciones experimentales.

Debido al alto costo computacional asociado a las pruebas en dispositivos embebidos, se selecciona un subconjunto de modelos representativos para su evaluación en el dispositivo edge. La selección se realiza considerando los modelos con el mayor rendimiento predictivo y según el análisis estadístico (ANOVA y prueba de Tukey), garantizando, además, la representación de las principales arquitecturas evaluadas.

Con el fin de controlar la variabilidad experimental, todas las arquitecturas se entrenan con el mismo conjunto de datos, los mismos hiperparámetros y el mismo procedimiento de entrenamiento. Las únicas variables experimentales modificadas fueron la arquitectura del modelo y su tamaño. Esto garantiza una comparación justa (fair comparison). Sin embargo, debido a la naturaleza estocástica del entrenamiento de redes neuronales profundas, el rendimiento de un modelo puede variar entre ejecuciones, incluso con los mismos hiperparámetros y los mismos datos de entrenamiento. Con el fin de obtener estimaciones robustas del rendimiento, cada arquitectura se entrena en cinco corridas independientes. Posteriormente se calculan estadísticas descriptivas (media, desviación estándar e intervalos de confianza) para cada métrica evaluada.

Las evaluaciones de rendimiento en el dispositivo se realizan exclusivamente sobre el conjunto de prueba, con el fin de garantizar la independencia respecto al entrenamiento y evitar sesgos derivados del ajuste de hiperparámetros.

Validez Interna

La validez interna en este proyecto se construye mediante un diseño experimental riguroso que garantiza que los resultados observados (eficiencia en la detección de EPP) sean atribuibles exclusivamente a las variables manipuladas (arquitecturas de redes neuronales, técnicas de optimización) y no a factores externos ni a sesgos metodológicos

(Hernández-Sampieri & Mendoza, 2018). Para lograrlo, se implementan las siguientes estrategias clave:

Control Estricto de Variables

Se identifican y neutralizan factores que pueden distorsionar la relación de causa y efecto entre el modelo de visión artificial y su rendimiento. Para efectos de esta investigación, se define como dispositivo de borde de bajo costo aquel sistema de cómputo orientado a aplicaciones de Edge Computing o Edge AI, que cumple simultáneamente los siguientes criterios:

- **Criterio económico absoluto:** precio comercial unitario aproximado < 262 USD, incluyendo los gastos de importación, con base en el 70% del salario mínimo legal vigente en Colombia en 2025, el cual corresponde a ~375 USD (Decreto 1572 de 2024, 2024).
- **Criterio arquitectónico:** basado en arquitectura ARM o SoC integrado o con Unidades de Procesamiento Neuronal (NPU).
- **Criterio de recursos:** memoria RAM ≤ 8 GB.
- **Criterio energético:** consumo máximo ≤ 25 W.
- **Criterio funcional:** orientado a la inferencia en el borde y no al entrenamiento de modelos a gran escala.

Esta definición permite diferenciar claramente las plataformas evaluadas de las estaciones de trabajo con GPU dedicada utilizadas en la fase de entrenamiento.

Para la fase de evaluación de dispositivos de borde (edge deployment), se selecciona exclusivamente el dispositivo Raspberry Pi 5 como plataforma de borde representativa de bajo costo. La selección se fundamenta en lo siguiente:

- Su arquitectura ARM de última generación.
- Disponibilidad comercial.
- Costo inferior al umbral definido (≤ 262 USD).
- Soporte para Unidad de Procesamiento Neuronal (NPU)
- Capacidad suficiente para ejecutar modelos YOLO bajo restricciones de energía.

El estudio no contempla la comparación entre múltiples plataformas de dispositivos embebidos, sino la evaluación del desempeño relativo entre distintas arquitecturas de modelos de detección ejecutandolas en una misma plataforma de hardware, con el fin de controlar variables externas asociadas al dispositivo y realizar una evaluación posterior en una Unidad de Procesamiento Neuronal (NPU).

Aleatorización en la composición del conjunto de datos

Con el fin de evitar sesgos de selección, se aplican técnicas de división estratificada, en las que el conjunto de datos se divide en: 70% para entrenamiento, 20% para validación y 10% para pruebas, procurando mantener las proporciones entre las clases. Adicionalmente, se generan imágenes sintéticas que reproducen diversas condiciones ambientales y ruido mediante técnicas de aumento de datos.

Protocolos de Medición Estandarizados

El uso de protocolos de medición estandarizados en la investigación garantiza que los datos obtenidos sean consistentes, comparables y confiables, lo que facilita la replicabilidad del estudio y la objetividad de los resultados. Al establecer procedimientos definidos y claros para la recolección y el análisis de información, se reduce la variabilidad en las mediciones y se minimizan los posibles sesgos, lo que fortalece la validez interna y externa del proyecto. Además, la estandarización permite identificar y

corregir errores de manera eficiente, asegurando que todas las etapas del proceso investigativo cumplan con los mismos criterios de calidad y rigor científico. En la **Tabla *6*** se definen las métricas precisas utilizadas para cuantificar las variables dependientes y asegurar su medición.

Tabla 6

Métricas e instrumentos de medición

Métrica	Definición	Instrumento
mAP	Promedio de AP sobre las imágenes del conjunto de datos.	PyTorch, TensorFlow, Python
FPS	Fotogramas por segundo en cada dispositivo.	PyTorch, TensorFlow, Python
Latencia (ms)	Latencia del proceso de inferencia medida en milisegundos.	PyTorch, TensorFlow, Python
Consumo de memoria (RAM)	Consumo del modelo en tiempo de ejecución medido en mebibytes (MiB) de RAM	Python, Sistema operativo
Tamaño del modelo (MiB)	Tamaño del modelo en mebibytes.	Python, Sistema operativo
Costo	Suma de los componentes de cada dispositivo de prueba.	Facturas de compra o cotizaciones

Nota. Estas mediciones se automatizaron mediante scripts en Python, con una tolerancia de error inferior al 5%. Elaboración propia.

Diferenciación respecto a validación cruzada

Es importante destacar que el enfoque adoptado no corresponde a un esquema de validación cruzada (cross-validation), ya que la partición del conjunto de datos se mantiene fija durante todas las corridas. En lugar de evaluar la variabilidad asociada a

diferentes particiones del conjunto de datos, esta estrategia se centra en cuantificar la variabilidad derivada del proceso estocástico de entrenamiento.

Este enfoque es consistente con prácticas comunes en el entrenamiento de modelos de aprendizaje profundo, donde el alto costo computacional limita la aplicación de validación cruzada tradicional, y se prioriza el análisis estadístico sobre múltiples ejecuciones independientes.

Población y Muestra

En el diseño metodológico del proyecto orientado a la detección de elementos de protección personal (EPP) mediante visión por computadora, la población y la muestra son conceptos fundamentales para asegurar la validez externa y la aplicabilidad de los resultados. Para este estudio, se considera población, a todas las imágenes y videos capturados en ambientes reales y simulados que representan la diversidad de situaciones y condiciones bajo las cuales debe operar el sistema de detección.

La muestra, en este caso, consiste en una selección representativa de esta población, diseñada para abarcar las principales variables de interés identificadas en el marco teórico y metodológico. En este caso, la muestra se compone de dos fuentes principales:

- **Imágenes y videos capturados en entornos reales:** Se recolectan fotografías de trabajadores en plantas industriales y obras de construcción, asegurando que la muestra incluya diferentes tipos de EPP seleccionados para el estudio (cascos, máscaras, guantes y chalecos), condiciones de iluminación, desde luz natural intensa hasta ambientes interiores con iluminación artificial, y situaciones de uso real, con oclusión parcial, movimiento, variaciones en la colocación del EPP.
- **Imágenes sintéticas y aumentadas:** Para robustecer la capacidad de generalización del modelo, se generan imágenes sintéticas mediante software de simulación que recrea condiciones extremas o poco frecuentes en los datos

reales, como oclusiones severas o iluminación extrema. Además, se aplican técnicas de aumento de datos (*data augmentation*) a las imágenes capturadas, lo que incrementa el tamaño del conjunto de datos y cubre un espectro de variabilidad más amplio.

La selección de la muestra probabilística es **estratificada** para garantizar que cada clase de EPP y cada tipo de condición ambiental estén representados de manera equilibrada. Esto permite que los resultados sean extrapolables a la población objetivo y minimiza el riesgo de sesgo tanto en el entrenamiento como en la validación del modelo. Además, la muestra se divide en conjuntos de entrenamiento, validación y prueba, siguiendo prácticas estándar de aprendizaje automático para evitar el sobreajuste y evaluar el desempeño real del sistema.

Técnicas e Instrumentos de Recolección de Datos

En esta investigación, la muestra está compuesta por un subconjunto de imágenes seleccionadas de bases de datos públicas, etiquetadas manualmente o semiautomáticamente. Las imágenes se clasifican según la presencia o ausencia de los EPP correspondientes y se dividen en conjuntos de entrenamiento, validación y prueba.

Tabla 7

Fuentes de datos de la investigación

Tipo	Fuente	URL
Publica	Roboflow	https://public.roboflow.com/
	Open Images	https://storage.googleapis.com/openimages/web/index.html
	Kaggle	https://www.kaggle.com/datasets
	Github	https://github.com/
	NIAID NIH	https://data.niaid.nih.gov/
	Figshare	https://figshare.com/
	Paper with Code	https://paperswithcode.com/dataset/openimages-v6

Nota. Lista de fuentes de datos de investigación que almacenan conjuntos de imágenes. Elaboración propia.

La técnica principal de recolección de datos consiste en la adquisición de imágenes a partir de fuentes públicas o mediante captura directa (descarga), seguida de un proceso de clasificación, selección y etiquetado que identifica las imágenes aptas para la generación del modelo, así como la categorización de los elementos de protección personal visibles en cada imagen. Las herramientas utilizadas para este propósito incluyen plataformas como: *LabelImg*, *Label Studio* y *Roboflow Annotate*, que permiten generar archivos de anotación compatibles con los modelos de entrenamiento. Adicionalmente, se procura asegurar la calidad y la diversidad del conjunto de datos recolectado.

Consideraciones Éticas

En esta investigación se adopta un enfoque de ética aplicada en inteligencia artificial, integrando principios de responsabilidad, transparencia, privacidad y proporcionalidad en el uso de tecnologías de visión por computadora para la detección de elementos de protección personal (EPP) en entornos laborales.

En relación con el uso de datos, se emplearon exclusivamente conjuntos de imágenes provenientes de repositorios públicos y abiertos, asegurando que estos cuentan con licencias adecuadas para su uso con fines académicos. No se realiza la recolección directa de datos en entornos reales ni el tratamiento de información personal identificable, lo que reduce significativamente los riesgos éticos asociados. Este enfoque se alinea con el principio de minimización de datos y con las buenas prácticas de investigación reproducible en el aprendizaje automático.

Desde el punto de vista normativo, la investigación se acoge a lo establecido en la Ley 1581 de 2012 (2012), que regula el tratamiento de datos personales en Colombia, así como a sus principios rectores: legalidad, finalidad, libertad, veracidad, transparencia, acceso y seguridad. Aunque el estudio no implicó el tratamiento directo de datos personales, se reconoce que una implementación real de un sistema de este tipo sí podría implicarlo, por lo que sería necesario garantizar mecanismos de consentimiento informado, anonimización y gobernanza de datos.

Adicionalmente, la investigación incorpora principios internacionales de ética en inteligencia artificial, en particular los propuestos por la OCDE, la UNESCO y el marco regulatorio emergente del Reglamento (UE) 2022/2065 sobre inteligencia artificial conocido como *AI Act* (REGLAMENTO (UE) 2024/1689, 2024). Estos marcos enfatizan aspectos clave como la supervisión humana, la equidad, la robustez técnica, la rendición de cuentas y la explicabilidad de los sistemas de IA. En coherencia con estos principios, los modelos desarrollados se plantean como una herramienta de apoyo a la toma de decisiones, evitando cualquier forma de automatización crítica que no requiera intervención humana.

En términos de privacidad y vigilancia, se reconoce que los sistemas de visión por computadora aplicados a entornos laborales pueden derivar en un monitoreo continuo, lo que plantea riesgos éticos para la autonomía y la dignidad de los trabajadores. Por ello, se adopta el principio de proporcionalidad, limitando el alcance funcional del sistema exclusivamente a la detección de EPP, sin incorporar capacidades de reconocimiento facial, de identificación biométrica ni de seguimiento individualizado. Asimismo, cualquier implementación futura deberá garantizar la transparencia en su uso, una comunicación clara a los trabajadores y mecanismos de control institucional.

Desde la perspectiva de la equidad y de los sesgos algorítmicos, se reconoce que los modelos de aprendizaje profundo pueden heredar sesgos presentes en los datos de

entrenamiento, lo que podría afectar su desempeño en contextos no representados adecuadamente. En consecuencia, se procura utilizar conjuntos de datos diversos y se documentan las limitaciones del modelo, en línea con el principio de responsabilidad científica y con las recomendaciones actuales sobre la auditoría de sistemas de IA.

Finalmente, esta investigación se orienta al desarrollo de soluciones tecnológicas con impacto social positivo, en particular en la reducción de accidentes laborales y el fortalecimiento de la cultura de seguridad. No obstante, se enfatiza que el despliegue de este tipo de sistemas debe realizarse dentro de marcos éticos, legales y organizacionales sólidos que garanticen el respeto de los derechos fundamentales y eviten usos indebidos, como la vigilancia desproporcionada o la toma de decisiones automatizadas sin supervisión humana.

Trabajo de Campo

Generación de los Modelos de Detección de Objetos

Para la generación de los modelos de detección de objetos, se elaboró inicialmente un conjunto de datos base que contiene imágenes de personas con los diferentes elementos de protección personal seleccionados para el estudio. Posteriormente, este conjunto de datos se tomó como base para entrenar los diferentes modelos seleccionados, utilizando scripts de batch y scripts en lenguaje Python para hacer este procedimiento reproducible.

A continuación, se detallan los elementos más importantes de la generación de estos modelos de detección de objetos:

Entorno de Entrenamiento

Todo el flujo de entrenamiento de los diferentes modelos se realizó en un equipo de cómputo con las siguientes características:

- **Sistema operativo:** Windows 11, Ubuntu LTS (WSL).
- **Memoria RAM:** 16 GiB
- **Disco:** 4 TiB
- **Tarjeta gráfica:** NVIDIA Geforce GTX 1080 Ti
- **Memoria VRAM:** GDDR5 11GB
- **Floating point:** FP32
- **Núcleos CUDA:** 3584
- **Entorno de ejecución:** Python 3.12 y torch 2.10+cu126

Creación del Conjunto de Datos

Inicialmente, se hizo una recopilación de conjuntos de datos disponibles públicamente con el fin de hacer una compilación y depuración de las imágenes incluidas, se

consultaron múltiples fuentes, principalmente Open Images, Kaggle, Github, NIAID NIH y Roboflow.

Producto de esta compilación inicial, se creó un conjunto de datos con las siguientes características:

- **Total de imágenes:** 13957
- **Entrenamiento:** 9770 imágenes (70%)
- **Validación:** 2791 imágenes (20%)
- **Pruebas:** 1396 imágenes (10%)
- **Resolución:** 512x512, 640x640, 1024x768
- **Incluye aumento de datos:** Sí

Varios de los conjuntos de datos recolectados contenían imágenes con diferentes tratamientos de aumento de datos, esto resulta inconveniente para el estudio ya que las herramientas de entrenamiento actuales permiten configurar este tipo de tratamiento de imágenes, en cuyo caso es realizado en el momento del entrenamiento, debido a esto, se realizó una depuración con el fin de eliminar estas imágenes y tener un conjunto de datos más limpio, así mismo se eliminaron imágenes duplicadas, imágenes poco relevantes y se unificaron las resoluciones a 640x640, finalmente se obtuvo un conjunto de datos con las siguientes características:

- **Total de imágenes:** 7634
- **Entrenamiento:** 5344 imágenes (70%)
- **Validación:** 1527 imágenes (20%)
- **Pruebas:** 763 imágenes (10%)
- **Resolución:** 640x640
- **Incluye aumento de datos:** No

Este conjunto de datos fue validado respecto a las etiquetas (labels) de las imágenes, este etiquetado y esta distribución se realizaron utilizando las herramientas Roboflow y LabelImg. Posteriormente se creó o exportó el conjunto de datos en los diferentes formatos necesarios para el entrenamiento según los modelos seleccionados, específicamente formato YOLO para cada una de las versiones y formato MS COCO, en este último se aplicó *resizing,* una técnica de preprocesamiento en la que se redimensiona la imagen a un tamaño de 576x576 con el fin de posibilitar el entrenamiento en modelos de tipo DEIM. Adicionalmente, se trató de mantener una proporción entre las etiquetas de las imágenes teniendo en cuenta las diferentes divisiones del conjunto de datos, como se muestra a continuación:

Tabla 8

Distribución de etiquetas del conjunto de datos

Etiqueta	Entrenamiento	Validación	Pruebas
Persona	4494	1366	629
Casco	4091	1207	619
Guantes	3156	851	455
Chaleco	2410	761	337
Mascara	2295	610	321

Nota. Distribución de las diferentes etiquetas del conjunto de datos utilizado respetando las proporciones. Elaboración propia.

Las imágenes utilizadas varían en sus etiquetas, en su mayoría, contienen la clase persona, ya que esta siempre va acompañada de algún elemento de protección personal. Sin embargo, cabe aclarar que algunas imágenes pueden incluir múltiples personas; por lo tanto, múltiples ocurrencias de 1 o más clases.

Análisis de distribución y balance del conjunto de datos

Para validar la distribución del conjunto de datos y calcular una métrica formal de desbalance, se utilizaron las técnicas de Proporción de desbalance (IR) y el Coeficiente de variación (CV).

$$IR = \frac{N_{max}}{N_{min}} \tag{1}$$

$$IR = \frac{4494}{2295} \approx 1{,}96$$

Como se observa, un IR de 1,96 (inferior a 3,0) indica un desbalance leve en el conjunto de datos, por lo que la distribución resulta aceptable para el estudio.

Asimismo, dentro de la distribución porcentual de las clases, la clase minoritaria “mascara”, con un porcentaje de 13,9%, no cae por debajo del 5%, lo que indica que se encuentra dentro de un rango adecuado para su estudio.

Para calcular el coeficiente de variación, se utiliza la siguiente fórmula:

$$CV = \frac{\sigma}{\mu} \tag{2}$$

$$CV = \frac{873}{3289} \approx 0{,}27$$

Dado que el coeficiente de variación es menor que 0,3, se infiere que el conjunto de datos presenta una dispersión baja y una buena uniformidad relativa entre las clases, lo cual es favorable para continuar con el estudio.

Aunque el desbalance es leve, las clases con menor representación (máscara y chaleco) podrían presentar un menor desempeño en términos de recall, especialmente si corresponden a objetos de menor tamaño relativo en la imagen.

Cálculo del Tamaño de la Muestra

En estudios comparativos de modelos de detección de objetos, las mejoras incrementales entre arquitecturas de una misma familia suelen situarse en rangos moderados, particularmente al comparar variantes optimizadas para dispositivos con recursos limitados. En benchmarks estándar como COCO (Lin et al., 2014), las diferencias en la desviación estándar reportadas entre generaciones sucesivas o variantes de modelos de detección suelen oscilar entre 1% y 5% en términos de mAP@0.5:0.95.

Arquitecturas ampliamente utilizadas como YOLO (Jegham et al., 2024), así como detectores modernos como DINO (Siméoni et al., 2025) y DETR (Zhao et al., 2023), evidencian mejoras progresivas en precisión que, si bien son estadísticamente relevantes, tienden a ser relativamente pequeñas en magnitud absoluta.

Por otra parte, la literatura sobre reproducibilidad en el aprendizaje profundo ha demostrado que los modelos basados en redes neuronales presentan variabilidad inherente asociada a la inicialización aleatoria, al ordenamiento de datos y a procesos estocásticos de optimización (Bhojanapalli et al., 2021). Esta variabilidad implica que diferencias pequeñas entre modelos pueden no ser detectables si no se controla adecuadamente el tamaño de la muestra experimental. En consecuencia, la estimación del tamaño del efecto esperado constituye un componente fundamental del diseño experimental.

En ausencia de datos preliminares propios, el tamaño del efecto se estimó con base en los rangos típicamente reportados en la literatura especializada sobre detección de objetos. Se estableció como diferencia mínima relevante una mejora del 2% en mAP@0.5:0.95, un valor consistente con los incrementos reportados entre variantes compactas de detectores modernos.

Asimismo, considerando estudios sobre la estabilidad en el entrenamiento de redes profundas, se asumió, de forma conservadora, una desviación estándar entre ejecuciones de 0,01. Bajo estas condiciones, el tamaño del efecto fue estimado mediante el índice de Cohen:

$$d = \frac{\Delta}{\sigma} \tag{3}$$

Donde:

- Δ = 0,02 (diferencia mínima relevante)
- σ = 0,01 (desviación estándar asumida)

$$d = \frac{0{,}02}{0{,}01} = 2{,}0$$

Este valor corresponde a un tamaño de efecto grande según la clasificación convencional.

El tamaño de muestra fue determinado mediante análisis de potencia estadística para comparación de medias independientes (t-test bilateral), utilizando la siguiente expresión:

$$n = \frac{2(Z_{\frac{\alpha}{2}} + Z_{\beta})}{d^2} \tag{4}$$

Considerando:

- Nivel de significancia α = 0,05
- Potencia estadística (1−β) = 0,80
- d = 2,0

Se obtuvo:

$$n \approx \frac{16}{d^2}$$

$$n = \frac{16}{4} = 4$$

Lo anterior indica que serían necesarias al menos 4 ejecuciones independientes por modelo para detectar diferencias respecto de la potencia especificada.

No obstante, con el fin de incrementar la robustez estadística del estudio y reducir posibles sesgos de subestimación de la varianza, se decidió realizar 5 ejecuciones independientes por modelo, garantizando así la estabilidad en la estimación de las medias y las desviaciones estándar.

Las ejecuciones experimentales se consideraron independientes al variar únicamente la semilla aleatoria del proceso de entrenamiento, manteniendo constantes la arquitectura, los hiperparámetros, el conjunto de datos, la resolución y el dispositivo de entrenamiento. Esta estrategia permitió estimar la variabilidad inherente al proceso de optimización estocástica, reducir el riesgo de error tipo II y fortalecer la validez inferencial del estudio.

Entrenamiento de los Modelos YOLO

Usando el conjunto de datos establecido, se entrenaron múltiples versiones de esta familia de arquitecturas con el fin de evaluarlas posteriormente en el estudio. Las versiones seleccionadas para el entrenamiento son: YOLOv8, YOLOv11, YOLOv12 y YOLOv26, se seleccionaron también los tamaños S (*small*) y N (*nano*) ya que son tamaños que facilitan su implementación en dispositivos de bajas prestaciones. Estas son las versiones más recientes y utilizadas de este conjunto de arquitecturas.

El entrenamiento de todos estos modelos se realizó bajo las mismas condiciones experimentales, para esto, se fijaron los mismos hiperparámetros para todos los modelos, con la excepción de la semilla (*seed*), la cual fue cambiada entre cada entrenamiento. En el **Anexo A** se presenta un resumen de los principales hiperparámetros configurados.

Adicionalmente, se implementaron técnicas de aumento de datos (data augmentation) y se aplicaron al entrenamiento de todas las versiones del modelo. En el **Anexo B**, se muestran los parámetros establecidos para el entrenamiento.

Resultados del Entrenamiento

Cada modelo se entrenó cinco veces, variando aleatoriamente la semilla de inicialización para capturar la variabilidad inherente al proceso de optimización estocástica. Los resultados de la métrica mAP@0.5:0.95 para cada ejecución de entrenamiento se presentan a continuación en el **Anexo C**.

En el **Anexo D** y **Anexo E**, se muestran las métricas estadísticas: promedio, desviación estándar e intervalos de confianza del 95%, tanto para tamaños *nano* y como para tamaños *small*. Las comparaciones se realizaron sobre mAP@0.5:0.95 como métrica principal, sin embargo, se presentan otras métricas importantes para el análisis de los resultados.

La métrica principal y más adecuada para la selección de los mejores modelos es mAP50-95, por ser más estricta, evalúa múltiples IoU y es estándar en MS COCO.

Teniendo en cuenta esto, se pueden ordenar los mejores modelos de tamaño *nano* y *small*, en la **Tabla *9*** se muestran de manera ordenada los modelos con mejor desempeño estadístico.

Tabla 9

Modelos YOLO con mejor desempeño estadístico de tamaño nano y small

No.	Modelo	mAP50-95	Std	CV (%)
1	YOLOv12n	0,583706	0,002291	0,39
2	YOLOv8n	0,581758	0,004990	0,45
3	YOLOv11n	0,581346	0,002637	0,86

No.	Modelo	mAP50-95	Std	CV (%)
4	YOLOv26n	0,566732	0,004108	0,72
1	YOLOv12s	0,598518	0,001487	0,25
2	YOLOv11s	0,596454	0,001353	0,23
3	YOLOv8s	0,593358	0,001548	0,26
4	YOLOv26s	0,586010	0,002120	0,36

Nota. Modelos YOLO con mejor desempeño estadístico en los tamaños nano y small. Elaboración propia.

Con esta información se pueden comparar los modelos con el mejor desempeño estadístico de cada tamaño (nano y small) para identificar si existe una diferencia apreciable en la precisión.

Tabla 10

Comparación estadística de modelos YOLO

Modelos	mAP_1	mAP_2	Dif. Abs. (Δ)	Dif. Rel. (%)
YOLOv12n vs YOLOv8n	0,583706	0,581758	0,001948	0,33%
YOLOv12n vs YOLOv11n	0,583706	0,581346	0,00236	0,41%
YOLOv12n vs YOLOv26n	0,583706	0,566732	0,016974	2,99%
YOLOv12s vs YOLOv11s	0,598518	0,596454	0,002064	0,35%
YOLOv12s vs YOLOv8s	0,598518	0,593358	0,005160	0,87%
YOLOv12s vs YOLOv26s	0,598518	0,586010	0,012508	2,13%

Nota. Modelos YOLO con mejor desempeño estadístico en los tamaños nano y small. Elaboración propia.

Teniendo en cuenta que los modelos YOLO v26 tanto para nano como para small presentan diferencias estadísticas evidentes respecto a los demás, se verifica si existen diferencias significativas entre los primeros 3 modelos de tamaño nano y small: YOLO

v12, YOLO v8 y YOLO v11, con el fin de concluir si todos se seleccionan para un análisis posterior mediante otras métricas, como FPS, y, de, la arquitectura YOLO óptima.

Análisis de Varianzas de un Factor para Arquitecturas Nano

Al realizar el análisis de varianzas, se busca determinar si existen diferencias estadísticamente significativas entre las arquitecturas de tamaño nano excluyendo YOLO v26n.

$$F = \frac{MSG}{MSE} \tag{5}$$

$$F = \frac{0{,}00000795}{0{,}00001237} = 0{,}642$$

$$F \gg F_{crit}$$

$$0{,}642 \ll 3{,}8853$$

Donde α = 0,05 y $F_{crit} = F(2{,}12) = 3{,}8853$

Al ser F mucho menor que el valor crítico F(2,12), se puede concluir que no hay diferencias estadísticamente significativas entre las arquitecturas de tamaño nano y que se deben obtener métricas adicionales, como FPS, para seleccionar el modelo óptimo.

Análisis de Varianzas de un Factor para Arquitecturas Small

Al realizar el análisis de varianzas, se busca determinar si existen diferencias estadísticamente significativas entre las arquitecturas de tamaño small, esto lo podemos hacer por medio de la ecuación (5).

$$F = \frac{0{,}00003373}{0{,}00000215} = 15{,}72$$

$$F \gg F_{crit}$$

$$15{,}72 \gg 3{,}8853$$

Donde α = 0,05 y $F_{crit} = F(2{,}12) = 3{,}8853$

Al ser F moderadamente mayor que el valor crítico F(2,12), se puede concluir que existen diferencias estadísticamente significativas entre las arquitecturas de tamaño small, incluso al excluir YOLO v26s.

Para abordar este inconveniente, se debe realizar una prueba de Tukey HSD de todas las versiones (k=4) para identificar las diferencias entre estos modelos.

$$HSD = q_{\alpha,k,df} \times \sqrt{\frac{MSE}{n}} \quad (6)$$

Donde:

k = 4

df = 16

n = 5

q = q(0,05,4,16) = 4,0464

MSE = 0,0000027324 (con k=4)

$$HSD = 4{,}0464 \times \sqrt{\frac{0{,}0000027324}{5}} = 0{,}002991$$

Tabla 11

Tabla de resultados de la prueba Tukey HSD

Pares	Dif. Absoluta	Dif. Absoluta > HSD
v12s vs v11s	0,002064	No significativo
v12s vs v8s	0,005160	Significativo
v12s vs v26s	0,012508	Significativo
v11s vs v8s	0,003096	Significativo
v11s vs v26s	0,010444	Significativo
v8s vs v26s	0,007348	Significativo

Nota. Resultados de la prueba de Tukey HSD para modelos small. Elaboración propia.

Con estos resultados se puede concluir lo siguiente:

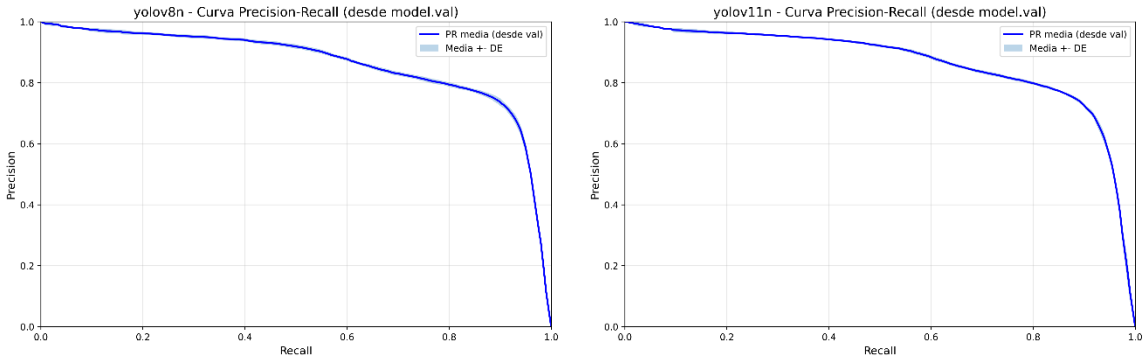
- YOLO v26s es inferior a todos.
- YOLO v12s y YOLO v11s son mejores que YOLO v8s.
- YOLO v12s no es significativamente mejor que YOLO v11s.

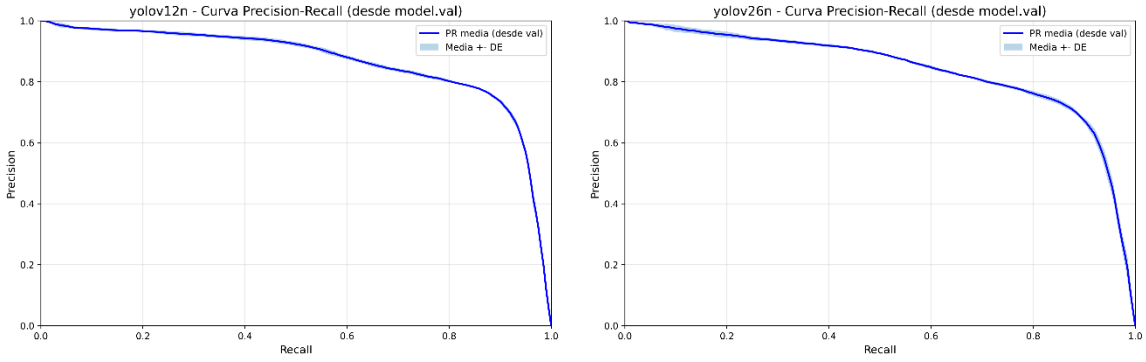
Otras Métricas de Entrenamiento

En el análisis de la fase de entrenamiento se pueden verificar métricas adicionales que nos ayudan a comprender las fortalezas y limitaciones de los modelos evaluados. A continuación, en la **Figura *11*** se muestran las curvas Precision-Recall para los modelos evaluados de tamaño *nano*, donde cada una corresponde al promedio de cinco ejecuciones de entrenamiento independientes. Adicionalmente, se reporta la banda de la media y la desviación estándar.

Figura 11

Curvas PR para modelos YOLO de tamaño nano





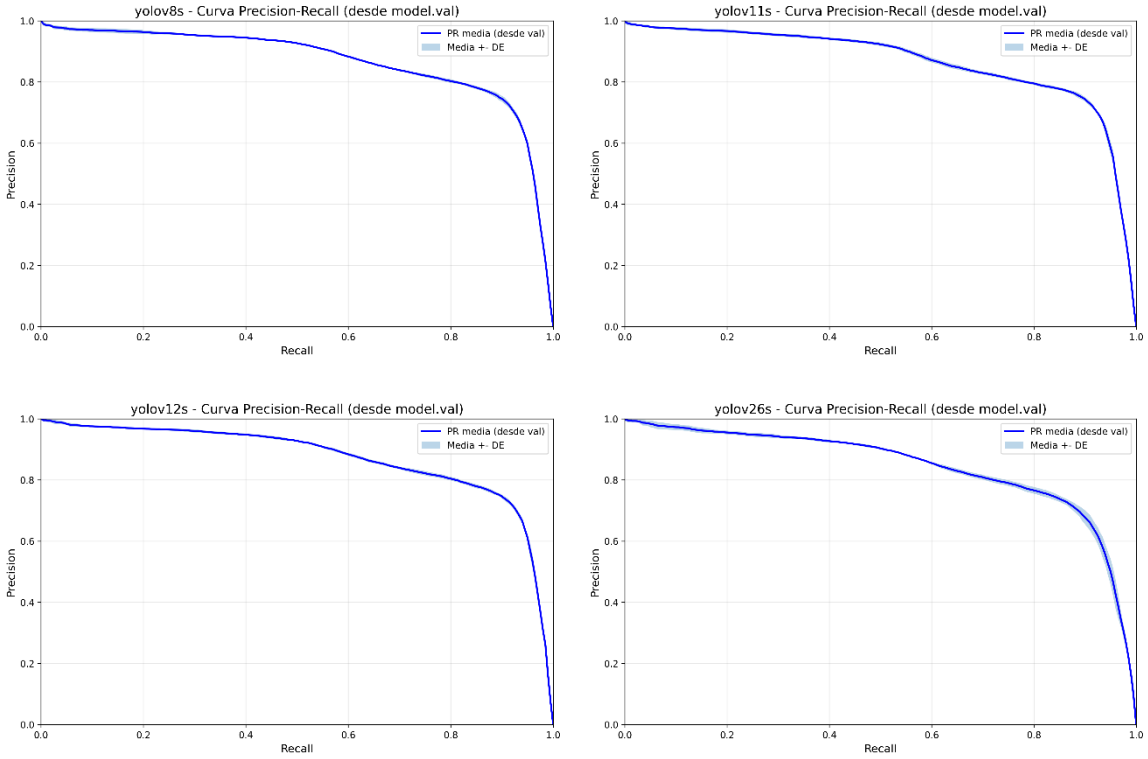
Nota. Se muestran las curvas de Precision-Recall de los modelos YOLO de tamaño nano. Elaboración propia.

Las curvas Precision–Recall de los modelos YOLO nano presentan un comportamiento similar entre arquitecturas, caracterizado por valores de precisión cercanos a 1,0 en niveles bajos de recall, lo que indica una baja tasa de falsos positivos cuando se emplean umbrales de confianza altos. A medida que el recall aumenta, la precisión disminuye gradualmente, manteniéndose generalmente entre 0,80 y 0,90 en el rango medio del recall (≈0,6–0,8). Este comportamiento refleja un equilibrio adecuado entre la capacidad de detección y el control de falsas alarmas en las distintas versiones del modelo. En todos los casos, se observa una caída pronunciada de la precisión cuando el recall se aproxima a 1, fenómeno típico de los detectores de objetos cuando se incluyen predicciones de menor confianza. La dispersión observada en las curvas indica una baja variabilidad entre las ejecuciones, lo que sugiere estabilidad en el proceso de entrenamiento de los modelos.

En la **Figura *12*** se muestran las curvas Precision-Recall para los modelos evaluados de tamaño *small*, donde cada una corresponde al promedio de cinco ejecuciones de entrenamiento independientes. Adicionalmente, se reporta la banda de la media y la desviación estándar.

Figura 12

Curvas PR para modelos YOLO de tamaño small



Nota. Se muestran las curvas de Precision-Recall de los modelos YOLO de tamaño small. Elaboración propia.

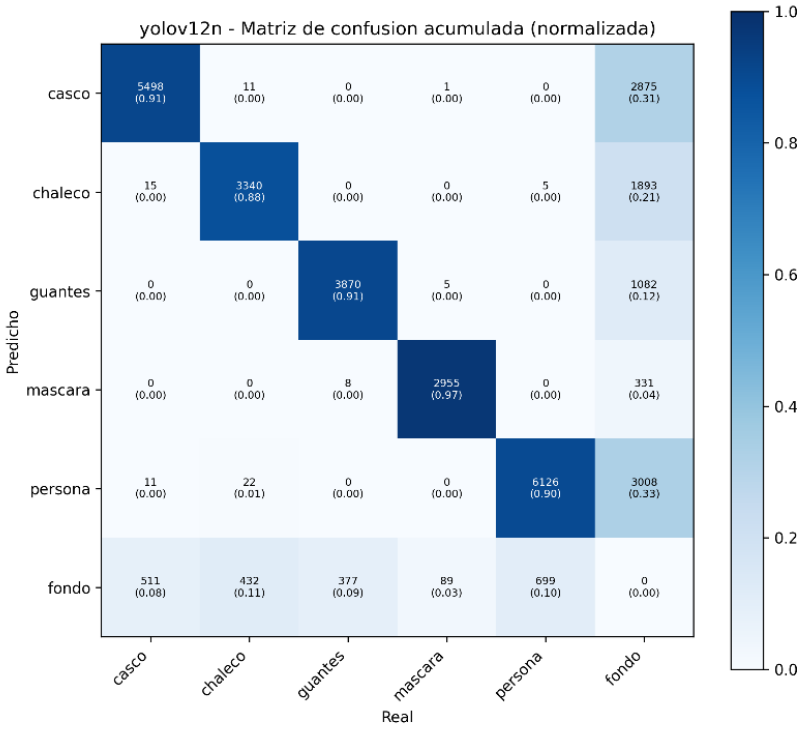
Las curvas Precision–Recall de los modelos YOLO en su variante Small muestran un comportamiento consistente entre las distintas arquitecturas. En todos los casos, la precisión se mantiene cercana a valores de 0,95–1,0 en niveles bajos de recall, lo que indica una baja tasa de falsos positivos cuando se emplean umbrales de confianza altos. A medida que el recall aumenta, la precisión disminuye gradualmente, manteniéndose generalmente entre 0,80 y 0,90 en el rango medio del recall (≈0,6–0,8). Este comportamiento evidencia un equilibrio adecuado entre la capacidad de detección y el control de las falsas alarmas.

En todos los modelos se observa una caída pronunciada de la precisión cuando el recall se aproxima a 1, un fenómeno común en detectores de objetos que incorporan predicciones de menor confianza. La dispersión observada en las curvas es relativamente baja, lo que sugiere estabilidad en el desempeño entre diferentes corridas de entrenamiento.

Igualmente, en las **Figura *13*** y **Figura *14*** se muestran, respectivamente, las matrices de confusión normalizadas para los modelos con mejor desempeño estadístico y menor variabilidad entre las ejecuciones evaluadas de tamaños *nano y small*, donde cada una corresponde al promedio de cinco ejecuciones de entrenamiento independientes.

Figura 13

Matriz de confusión normalizada para YOLO v12 nano



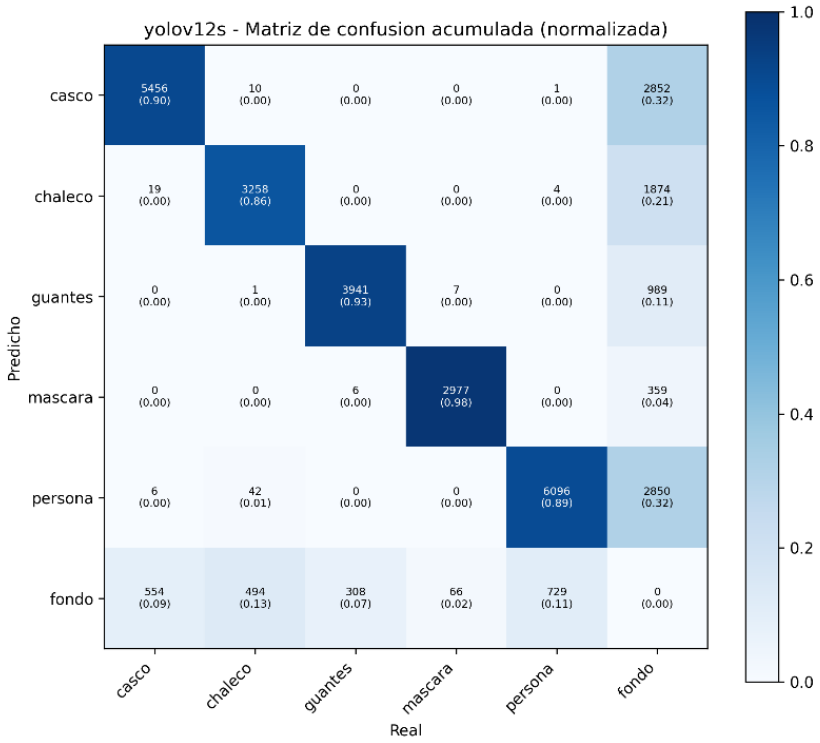
Nota. Matriz de confusión normalizada para el modelo YOLO v12 nano. Elaboración propia.

La matriz de confusión acumulada muestra un alto nivel de aciertos en la diagonal principal para todas las clases, lo que indica que los modelos logran identificar correctamente la mayoría de los elementos de protección personal. La clase máscara presenta consistentemente las tasas de detección más altas (≈0,96–0,97), seguida por guantes y casco, mientras que las clases chaleco y persona presentan un desempeño ligeramente inferior en algunas arquitecturas.

La confusión entre clases es mínima en todos los modelos, lo que evidencia que las redes aprenden representaciones visuales adecuadas para diferenciar los distintos tipos de EPP. Los errores observados se concentran principalmente en falsos negativos clasificados como fondo, especialmente en las clases chaleco, casco y persona, lo cual es un comportamiento común en detectores de objetos cuando algunas instancias no alcanzan el umbral de confianza requerido.

Figura 14

Matriz de confusión normalizada para YOLO v12 small



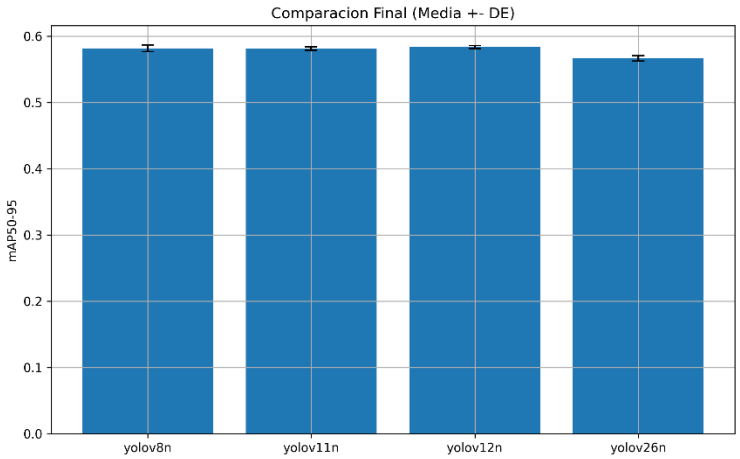
Nota. Matriz de confusión normalizada para el modelo YOLO v12 small. Elaboración propia.

Las matrices de confusión acumuladas muestran altas tasas de acierto en la diagonal principal para todas las clases, lo que indica que los modelos identifican correctamente la mayoría de las instancias de elementos de protección personal. La clase máscara presenta el mejor desempeño (≈0,97–0,98) en todas las arquitecturas, seguida por guantes (≈0,92–0,93). Las clases casco y persona muestran valores ligeramente inferiores, mientras que chaleco presenta consistentemente el menor desempeño relativo entre las clases evaluadas.

La confusión entre clases es mínima, lo que indica que las redes aprenden representaciones visuales lo suficientemente discriminativas como para distinguir los distintos elementos de protección personal. Los errores observados se concentran principalmente en falsos negativos clasificados como fondo, especialmente en las clases chaleco, casco y persona, lo cual es un patrón común en modelos de detección cuando existen regiones visualmente ambiguas en la escena.

Figura 15

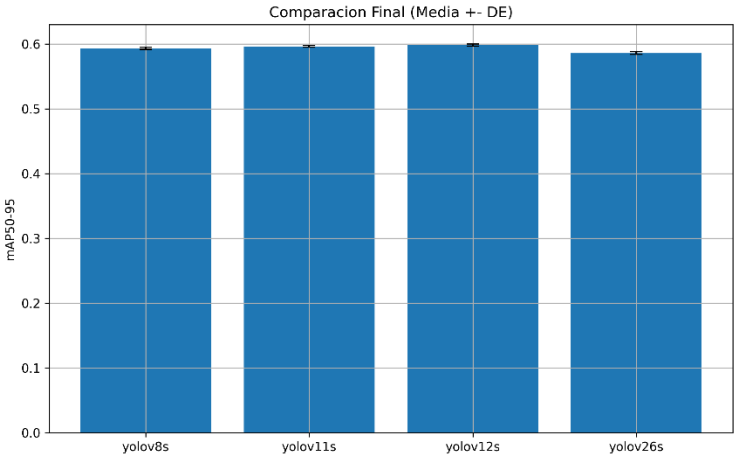
Comparación de los diferentes modelos de YOLO de tamaño nano



Nota. Gráfico de barras que compara los diferentes modelos de YOLO de tamaño nano, incluyendo el indicador de desviación estándar. Elaboración propia.

Figura 16

Comparación de los diferentes modelos de YOLO de tamaño small



Nota. Gráfico de barras que compara los diferentes modelos de YOLO de tamaño small, incluyendo el indicador de desviación estándar. Elaboración propia.

Finalmente, en la **Figura *15*** y **Figura *16*** se muestra una comparación de los modelos en sus tamaños *nano* y *small*, que incluye la media y la desviación estándar de la métrica de precisión mAP50-90.

Análisis de Escalamiento entre Tamaños

Con este análisis se busca calcular el impacto de realizar un escalamiento de los tamaños de arquitectura YOLO nano a YOLO small, para ello, se calculan las diferencias absolutas y relativas de cada par de modelos en sus diferentes tamaños:

Tabla 12

Tabla de diferencias en el escalamiento entre tamaños de YOLO

Modelo	Nano	Small	Dif. Absoluta	Dif. Relativa
v8	0,581758	0,593358	0,011600	+1,99%
v11	0,581346	0,596454	0,015108	+2.60%
v12	0,583706	0,598518	0,014812	+2.54%
v26	0,566732	0,586010	0,019278	+3,40%

Nota. Se muestran las diferencias en el escalamiento entre los tamaños de YOLO. Elaboración propia.

Con lo anterior, se puede concluir que la mejora por escalamiento (≈2–3,4%) es 6–8 veces mayor que la mejora entre generaciones (≈0,3–0,4%), es decir, el impacto del escalamiento del modelo es significativamente mayor que el del cambio de versión arquitectónica.

Para analizar más rigurosamente la interacción entre las variables de estos modelos se emplea un ANOVA factorial 2x4 (tamaño x arquitectura) con 5 replicaciones. En la **Tabla *13*** se presenta la lista de parámetros necesarios para calcularla.

La ANOVA factorial se calcula usando las siguientes fórmulas según su variable (A para arquitectura, B para tamaño):

$$F_A = \frac{MS_A}{MS_E}$$
$$F_B = \frac{MS_B}{MS_E} \quad (7)$$

Tabla 13

Tabla de resultados para ANOVA factorial 2x4 de modelos YOLO

Origen	Suma de cuadrados (SS)	Grados de libertad (df)	Promedio de los cuadrados (MS)	F	F Crítico
Arquitectura	0,001296867	3	0,000432289	53,2757	2.90112
Tamaño	0,002310248	1	0,002310248	284,7175	4,14910
Interacción	7,43727E-05	3	2.47909E-05	3,05526	2.90112
Error	0,000259654	32	8,11418E-06	-	-

Nota. Se muestran los parámetros necesarios para calcular la ANOVA factorial 2x4 de los modelos YOLO. Elaboración propia.

En cuanto a la arquitectura se puede observar que:

$$F_A \gg 2.90112$$

$$53{,}2757 \gg 2.90112$$

El valor es muy significativo, lo que indica que el rendimiento depende significativamente de la arquitectura del modelo.

En cuanto al tamaño se puede observar que:

$$F_B \gg 4{,}14910$$

$$284{,}7175 \gg 4{,}14910$$

El valor es extremadamente significativo, lo que indica que el tamaño del modelo (nano vs small) tiene un impacto muy fuerte en el mAP.

En cuanto al Tamaño x Arquitectura (Interacción) se puede observar que:

$$F_B \gg 4{,}14910$$

$$3{,}05526 \gg 2.90112$$

El valor es ligeramente significativo (al 5%), lo que indica que el efecto de aumentar el tamaño del modelo no es el mismo en todas las arquitecturas.

El análisis ANOVA factorial (Arquitectura × Tamaño) evidenció efectos significativos tanto de la arquitectura del modelo ($F(3, 32) = 53{,}28$, $p < 0{,}001$), como del tamaño del modelo ($F(1, 32) = 284{,}72$, $p < 0{,}001$). Asimismo, se observó una interacción significativa entre la arquitectura y el tamaño ($F(3, 32) = 3{,}06$, $p < 0{,}042$), lo que indica que el beneficio de aumentar el tamaño del modelo depende de la arquitectura específica.

Entrenamiento de los Modelos RF-DETR

Para el entrenamiento de los modelos de la familia RF-DETR se seleccionaron los tamaños S (*small*) y N (*nano*), ya que facilitan su implementación en dispositivos de bajas prestaciones. Estos fueron entrenados con el conjunto de datos ya establecido. Sin embargo, en este caso se utilizó una resolución ligeramente mayor: 670 x 670. Esto se debe a que es la resolución más cercana soportada por su arquitectura.

El entrenamiento de estos modelos se realizó bajo las mismas condiciones experimentales, para esto, se fijaron los mismos hiperparámetros para todos los modelos, con la excepción de la semilla (*seed*), la cual fue cambiada aleatoriamente entre cada entrenamiento. En el **Anexo F**, se muestra un resumen de los principales hiperparámetros configurados.

Adicionalmente, se utilizaron técnicas de aumento de datos (data augmentation) aplicándolas al entrenamiento de los distintos tamaños del modelo, si bien estas no son las mismas que las utilizadas en el entrenamiento de las otras arquitecturas, se utilizaron las recomendadas por su equipo de creadores. A continuación, se muestran los parámetros de aumento de datos establecidos para el entrenamiento:

- **Random Horizontal Flip**.
- **Square Resize:** 448, 512, 576, 640, 704, 768, 832, 896
- **Random Resize:** 400, 500, 600

- **Random Size Crop:** 384, 600

Resultados del Entrenamiento

Cada modelo se entrenó cinco veces, variando aleatoriamente la semilla de inicialización. Desafortunadamente, debido a las características de la arquitectura (retropropagación en grid) y a su naturaleza multihilo, no es posible garantizar la reproducibilidad exacta a pesar de que este modelo utiliza una semilla. Los resultados de la métrica mAP@0.5:0.95 para cada ejecución de los entrenamientos se presentan a continuación en la **¡Error! No se encuentra el origen de la referencia.**.

Tabla 14

Resultados de los modelos entrenados de tamaño nano y small (RF-DETR)

Modelo	Ejecución 1	Ejecución 2	Ejecución 3	Ejecución 4	Ejecución 5	Promedio
Nano	0,612283	0,613173	0,613959	0,612348	0,608816	0,612116
Small	0,616952	0,619155	0,613462	0,612895	0,615867	0,615666

Nota. Tabla de resultados de mAP@0.5:0.95 de los productos del entrenamiento de los diferentes tamaños de RF-DETR: nano y small. Elaboración Propia.

A continuación, se muestran las métricas estadísticas: promedio, desviación estándar e intervalos de confianza del 95%, en la **¡Error! No se encuentra el origen de la r eferencia.** tanto para tamaños *nano* como para tamaños *small*. Las comparaciones se realizaron sobre mAP@0.5:0.95 como métrica principal, sin embargo, se presentan otras métricas importantes para el análisis de los resultados.

Tabla 15

Métricas estadísticas de los modelos de tamaño nano y small (RF-DETR)

Modelo	Métrica	Promedio	STD	IC inf.	IC sup.	CV (%)
Nano	mAP50-95	0,612116	0,001968	0,611344	0,612887	0,32%

Modelo	Métrica	Promedio	STD	IC inf.	IC sup.	CV (%)
	Precision	0,775233	0,007323	0,772362	0,778103	0,94%
	Recall	0,908917	0,010768	0,904696	0,913138	1,18%
Small	mAP50-95	0,615666	0,002569	0,614659	0,616674	0,42%
	Precision	0,773781	0,004375	0,772066	0,775496	0,56%
	Recall	0,921117	0,004366	0,919405	0,922828	0,47%

Nota. Tabla de métricas estadísticas del entrenamiento de los modelos RF-DETR de tamaño nano y small. Elaboración propia.

El modelo RF-DETR Small presenta un mAP50-95 promedio ligeramente superior (0,6157) en comparación con RF-DETR Nano (0,6121). La diferencia absoluta es de aproximadamente 0,0035 (~0,57%), lo que sugiere una mejora marginal en la capacidad de detección global del modelo Small. Además, los intervalos de confianza (IC 95%) de ambos modelos son muy estrechos y presentan mínima superposición, lo que indica alta estabilidad en las mediciones y sugiere que la diferencia observada es consistente entre corridas experimentales.

El coeficiente de variación (CV) es bajo en ambos casos (Nano: 0,32%; Small: 0,42%), lo que evidencia una alta reproducibilidad tanto en el entrenamiento como en la evaluación.

La diferencia de Precision es prácticamente despreciable (~0,0014). Esto indica que ambas variantes mantienen una tasa de falsos positivos comparable, sin que el incremento del tamaño del modelo Small produzca mejoras sustanciales en esta métrica. El CV inferior al 1% en ambos modelos confirma una baja variabilidad experimental.

El modelo Small mejora el recall en aproximadamente 1,22 puntos porcentuales, lo que sugiere que detecta un mayor número de objetos verdaderos en comparación con la versión Nano. Esto puede atribuirse a una mayor capacidad representacional del modelo Small, lo que permite capturar más instancias del objeto objetivo. Además, el CV del

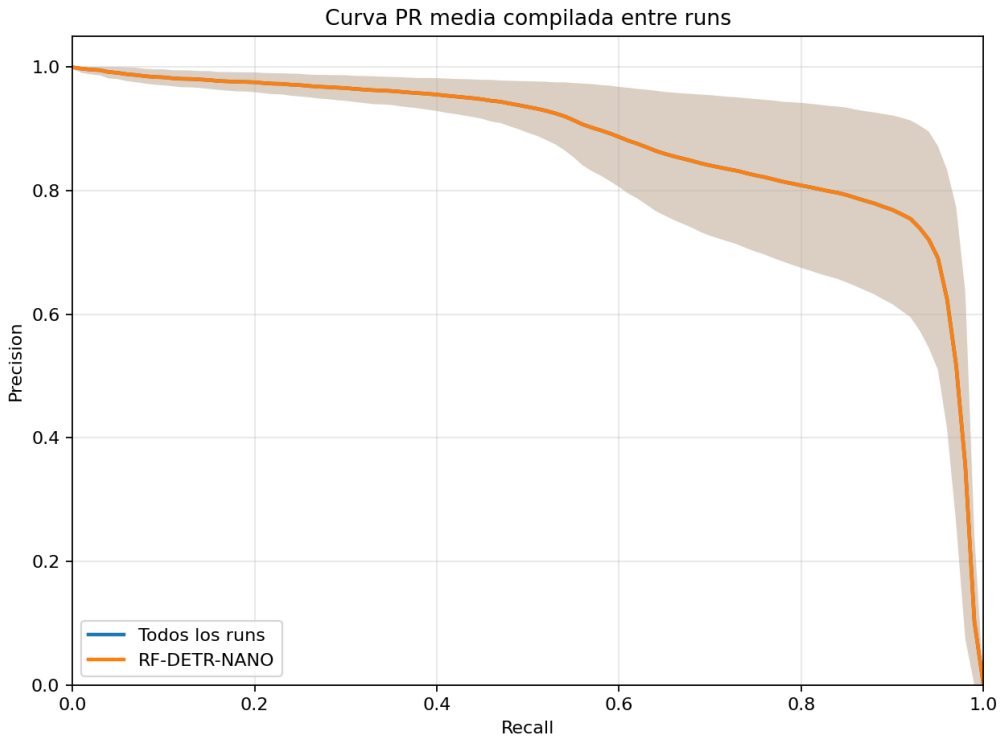
recall es muy bajo (Nano: 1,18%, Small: 0,47%), lo que evidencia la consistencia de los resultados entre ejecuciones.

Otras métricas de Entrenamiento

En el análisis de la fase de entrenamiento se pueden verificar métricas adicionales que nos ayudan a comprender las fortalezas y limitaciones de los modelos evaluados. A continuación, en la **Figura *21*** se muestra la curva Precision-Recall del modelo evaluado de tamaño *nano*, correspondiente al promedio de cinco ejecuciones de entrenamiento independientes. Adicionalmente, se reportan la banda de la media y la desviación estándar.

Figura 17

Curva PR para el modelo RF-DETR de tamaño nano



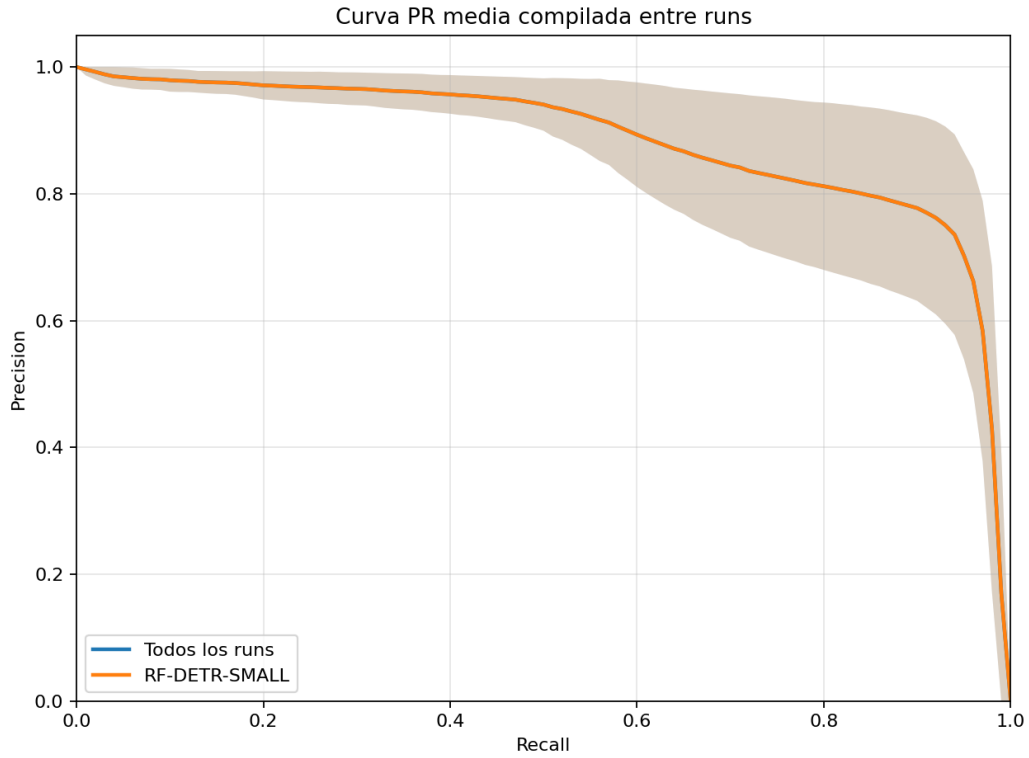
Nota. Se muestra la curva de Precision-Recall del modelo RF-DETR de tamaño nano. Elaboración propia.

La curva Precision–Recall promedio, obtenida a partir de múltiples ejecuciones, muestra un comportamiento consistente del modelo RF-DETR en la detección de elementos de protección personal. Se observa que la precisión se mantiene cercana a 0,95–1,0 en los niveles de recall bajos y medios, lo que indica que el modelo genera un número reducido de falsos positivos cuando las detecciones se realizan con alta confianza. A medida que el recall aumenta, la precisión disminuye gradualmente, alcanzando valores cercanos a 0,8 para valores de recall superiores a 0,8, lo cual refleja el comportamiento esperado en sistemas de detección de objetos al intentar recuperar un mayor número de instancias. Finalmente, se evidencia una caída abrupta de la precisión cuando el recall se aproxima a 1, asociada a detecciones de baja confianza que incrementan los falsos positivos. La región sombreada entre ejecuciones muestra baja variabilidad en la mayor parte del rango de recall, lo que sugiere estabilidad en el proceso de entrenamiento y buena reproducibilidad del modelo.

En la **Figura *22*** se muestra la curva Precision-Recall del modelo evaluado de tamaño *small*, correspondiente al promedio de cinco ejecuciones de entrenamiento independientes. Adicionalmente, se reportan la banda de la media y la desviación estándar.

Figura 18

Curva PR para el modelo RF-DETR de tamaño small



Nota. Se muestra la curva de Precision-Recall del modelo RF-DETR de tamaño small. Elaboración propia.

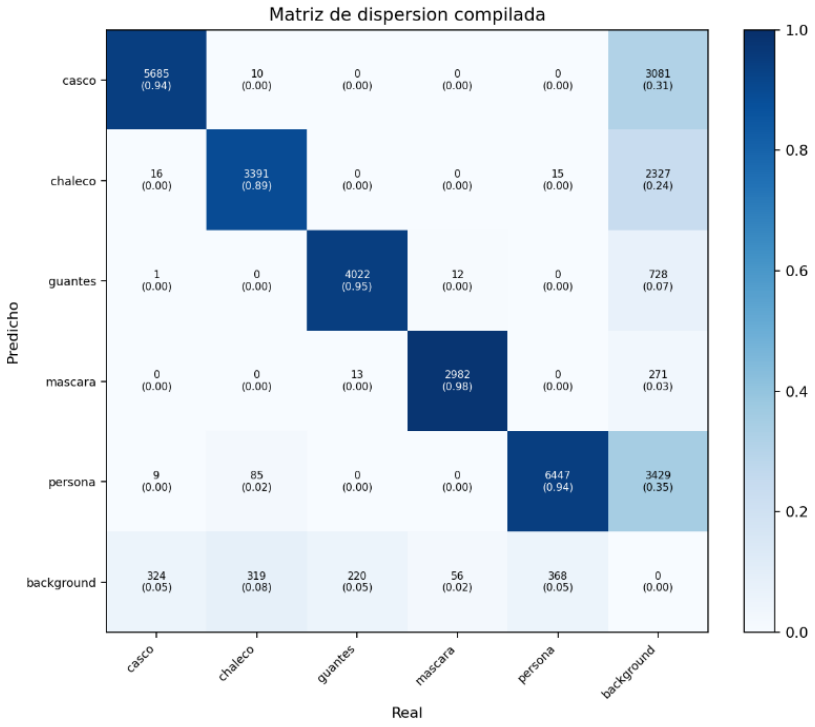
La curva de Precision–Recall promedio del modelo RF-DETR Small muestra un comportamiento estable a lo largo de su rango de operación. La precisión se mantiene cercana a 1,0 en valores bajos de recall, lo que indica que el modelo genera un número muy reducido de falsos positivos cuando opera con umbrales de confianza altos. A medida que el recall aumenta, la precisión disminuye de forma gradual, manteniéndose aproximadamente entre 0,80 y 0,90 para valores de recall entre 0,6 y 0,9, lo que refleja un equilibrio adecuado entre la capacidad de detección y el control de falsos positivos. Al igual que en otros detectores de objetos, se observa una caída abrupta de la precisión cuando el recall se aproxima a 1, asociada a detecciones de baja confianza que incrementan la tasa de falsos positivos. La banda de dispersión entre ejecuciones

muestra baja variabilidad en la mayor parte del rango de recall, lo que indica estabilidad en el entrenamiento y consistencia en el desempeño del modelo entre distintas corridas.

Igualmente, en las **Figura *23*** y **Figura *24*** se muestran, respectivamente, las matrices de confusión normalizadas para los modelos evaluados de tamaños *nano y small*, donde cada una corresponde al promedio de cinco ejecuciones de entrenamiento independientes.

Figura 19

Matriz de confusión normalizada para DEIMv2 nano



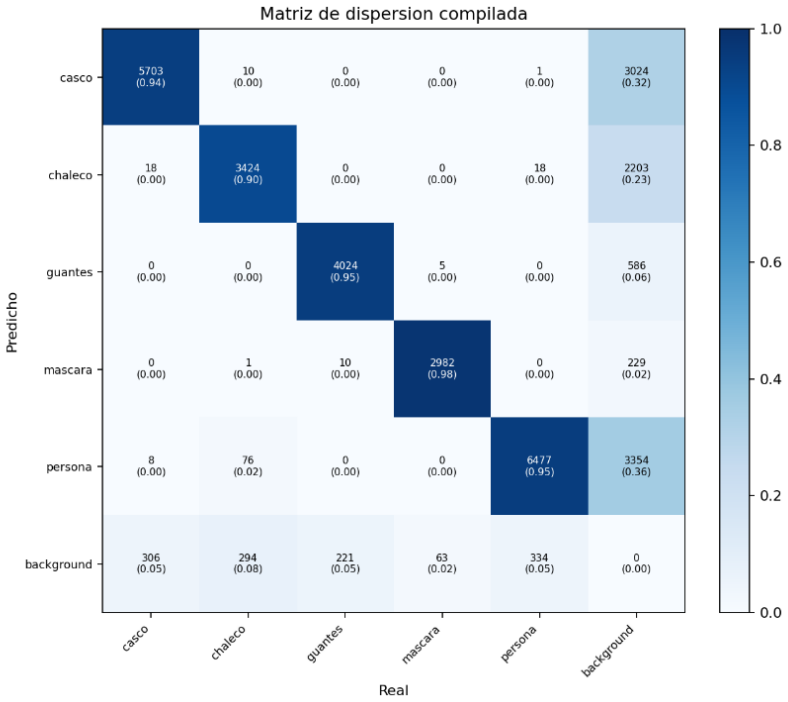
Nota. Matriz de confusión normalizada del modelo RF-DETR nano. Elaboración propia.

La matriz de confusión evidencia un alto nivel de acierto en la diagonal principal, lo que indica que el modelo identifica correctamente la mayoría de las instancias de cada clase. Las tasas de detección más altas se observan en las clases máscara (≈0,98) y guantes (≈0,95), seguidas de casco y persona (≈0,94), mientras que chaleco presenta el

desempeño relativamente menor (≈0,89). Los errores de clasificación entre clases son poco frecuentes, lo que sugiere que el modelo logra aprender características visuales discriminativas entre los distintos elementos de protección personal. La principal fuente de error es la clasificación de falsos negativos como fondo, lo cual es un comportamiento común en detectores de objetos cuando algunas instancias no alcanzan el umbral de confianza requerido. Asimismo, se observan algunos falsos positivos asociados al fondo, particularmente en las clases persona y casco, posiblemente relacionados con patrones visuales similares del entorno.

Figura 20

Matriz de confusión normalizada para DEIMv2 small



Nota. Matriz de confusión normalizada del modelo RF-DETR small. Elaboración propia.

La matriz de confusión evidencia un alto nivel de aciertos en la diagonal principal, lo que indica que el modelo identifica correctamente la mayoría de las instancias de cada

clase de elemento de protección personal. Las tasas de detección correctas más altas se observan en máscara (≈0,98) y guantes (≈0,95), seguidas por persona (≈0,95) y casco (≈0,94), mientras que chaleco presenta un desempeño ligeramente menor (≈0,90).

La confusión entre clases es muy limitada, lo que sugiere que el modelo logra aprender representaciones discriminativas adecuadas para diferenciar entre los distintos tipos de EPP. Los errores observados corresponden principalmente a falsos negativos clasificados como fondo, especialmente en las clases casco, chaleco y persona, lo cual es un comportamiento habitual en sistemas de detección de objetos cuando ciertas instancias no alcanzan el umbral de confianza requerido. Asimismo, se identifican algunos falsos positivos provenientes del fondo, particularmente en las clases persona y casco, probablemente asociados a patrones visuales similares presentes en el entorno.

Análisis de Escalamiento entre Tamaños

Con este análisis se busca calcular el impacto de realizar un escalamiento del tamaño de arquitectura RF-DETR nano a RF-DETR small, para ello, se calcula las diferencias absolutas y relativas de cada par de modelos en sus diferentes tamaños:

Se halla la diferencia absoluta para los dos modelos:

$$\Delta = a - b \tag{8}$$

$$\Delta = 0{,}615666 - 0{,}612116 = 0{,}00355$$

Esto indica una diferencia absoluta apreciable de aproximadamente 0,3%. Ahora se puede hallar la diferencia porcentual relativa:

$$DRP = \frac{a - b}{a} \times 100 \tag{9}$$

$$\frac{0{,}00355}{0{,}615666} \times 100 = 0{,}58\%$$

El modelo de tamaño small mejora en aproximadamente 0,58% de mAP50-95 respecto al de tamaño nano, claramente inferior.

La comparación entre los modelos RF-DETR Nano y RF-DETR Small muestra comportamientos muy similares en términos de precisión y capacidad de detección, lo que se refleja en las curvas Precision–Recall prácticamente equivalentes. Ambos modelos mantienen niveles de precisión superiores a 0,9 en gran parte del rango de recall, evidenciando una alta confiabilidad en las detecciones. No obstante, el modelo RF-DETR Small presenta ligeras mejoras en la detección de algunas clases, en particular de persona y chaleco, donde se observa un incremento marginal en la tasa de aciertos en la diagonal de la matriz de confusión. Asimismo, el modelo Small muestra una ligera reducción de algunos falsos negativos y falsos positivos asociados al fondo, lo que sugiere una mayor capacidad de generalización.

A pesar de estas mejoras, las diferencias entre ambos modelos son relativamente pequeñas, lo que indica que el modelo Nano logra mantener un desempeño competitivo a pesar de su menor complejidad arquitectónica. En consecuencia, mientras el modelo Small ofrece una leve ventaja en precisión y robustez, el modelo Nano puede resultar más eficiente en escenarios de computación en el borde, donde las restricciones de recursos computacionales son un factor determinante.

Entrenamiento de los Modelos DEIMv2

Usando el conjunto de datos inicialmente establecido, se entrenaron dos modelos, uno para el tamaño N (nano) y otro para el tamaño S (small) de esta arquitectura, con imágenes de 640x640.

El entrenamiento de todos estos modelos, como en los casos anteriores, se realizó bajo las mismas condiciones experimentales, para esto, se fijaron los mismos hiperparámetros para todos los modelos, con la excepción de la semilla (*seed*), la cual

cambia entre cada entrenamiento. En el **Anexo G**, se muestra un resumen de los principales hiperparámetros configurados.

Adicionalmente, se implementaron técnicas de aumento de datos (data augmentation) aplicándolas al entrenamiento de los distintos tamaños del modelo. En el **Anexo H**, se muestran los parámetros establecidos para el entrenamiento.

Resultados del Entrenamiento

Cada modelo se entrenó cinco veces, variando aleatoriamente la semilla de inicialización, con el fin de capturar la variabilidad inherente al proceso de optimización estocástica. Los resultados de la métrica mAP@0.5:0.95 para cada ejecución de los entrenamientos se presentan a continuación en la **¡Error! No se encuentra el origen de l a referencia.**.

Tabla 16

Resultados de los modelos entrenados de tamaño nano y small (DEIMv2)

Modelo	Ejecución 1	Ejecución 2	Ejecución 3	Ejecución 4	Ejecución 5	Promedio
Nano	0,509899	0,487240	0,477505	0,490450	0,489223	0,490862
Small	0,489519	0,503287	0,483195	0,514690	0,534512	0,505041

Nota. Tabla de resultados de mAP@0.5:0.95 productos del entrenamiento de los diferentes tamaños DEIMv2: nano y small. Elaboración Propia.

A continuación, se muestran las métricas estadísticas: promedio, desviación estándar e intervalos de confianza del 95%, en la **Tabla *17*** tanto para tamaños *nano* como para tamaños *small*. Las comparaciones se realizaron sobre mAP@0.5:0.95 como métrica principal, sin embargo, se presentan otras métricas importantes para el análisis de los resultados.

Tabla 17

Métricas estadísticas de los modelos de tamaño nano y small (DEIMv2)

Modelo	Métrica	Promedio	STD	IC inf.	IC sup.	CV (%)
Nano	mAP50-95	0,490862	0,029632	0,479246	0,502477	6%
	Precision	0,774957	0,121314	0,727403	0,822511	15%
	Recall	0,887200	0,046772	0,868865	0,905534	5%
Small	mAP50-95	0,505041	0,063831	0,480019	0,530062	13%
	Precision	0,779373	0,138417	0,725114	0,833632	18%
	Recall	0,915600	0,029450	0,904055	0,927144	3%

Nota. Tabla de métricas estadísticas del entrenamiento de los modelos DEIMv2 de tamaño nano y small. Elaboración propia.

El modelo DEIMv2 Small presenta un mAP50–95 promedio de 0,5050, mientras que DEIMv2 Nano alcanza 0,4909, lo que representa una mejora aproximada de 0,014 puntos (≈2.9%) a favor de la versión Small. Esta diferencia sugiere que el incremento del tamaño del modelo proporciona una ligera mejora en la capacidad global de detección.

Sin embargo, a diferencia de lo observado en otros modelos, los intervalos de confianza se superponen considerablemente entre ambos modelos, lo que indica que la diferencia observada podría no ser estadísticamente significativa y requerir un análisis adicional para confirmarla.

Asimismo, el coeficiente de variación (CV) es relativamente mayor que en otros modelos evaluados, especialmente en el caso de Small (13%), lo que indica una mayor variabilidad entre corridas experimentales y una menor estabilidad en los resultados de mAP.

En lo referente a la Precision, la diferencia es mínima (~0,004), lo que indica que el aumento del tamaño del modelo no produce mejoras sustanciales en la reducción de falsos positivos. Sin embargo, se observa una alta variabilidad en esta métrica, con CV

del 15% en Nano y del 18% en Small, lo que sugiere que la precisión es inestable entre corridas. Esto podría estar relacionado con la sensibilidad del modelo a la inicialización del entrenamiento, o con la dificultad de ciertas clases.

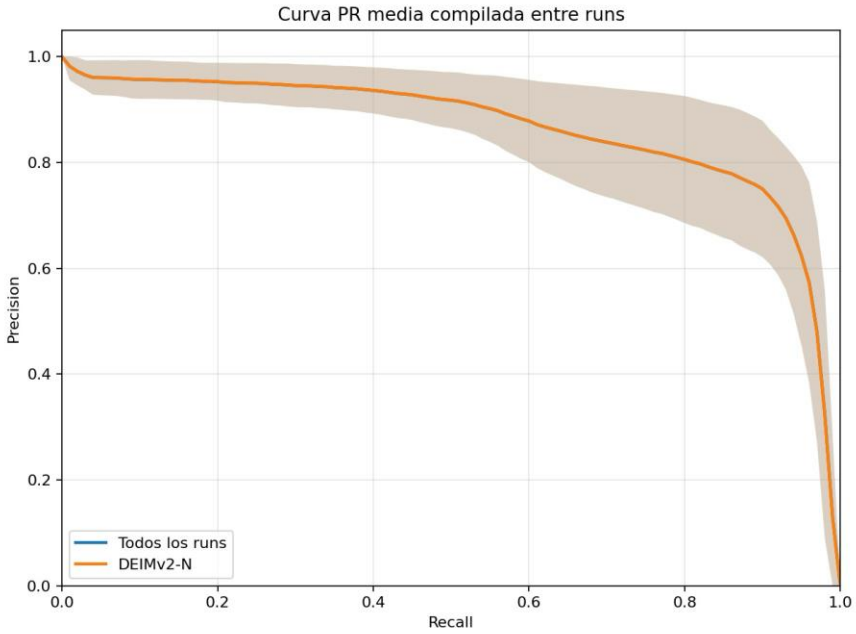
El modelo Small supera al Nano en aproximadamente 2,84 puntos porcentuales de recall, lo que indica mayor capacidad para detectar instancias reales de los objetos, reduciendo la proporción de falsos negativos. Además, el CV del recall es relativamente bajo, especialmente en el modelo Small (3%), lo que evidencia una mayor estabilidad en la capacidad de detección del modelo de mayor tamaño.

Otras Métricas de Entrenamiento

A continuación, en la **Figura *21*** se muestra la curva Precision-Recall que incluye la banda de la desviación estándar del modelo evaluado de tamaño *nano*, correspondiente al promedio de cinco ejecuciones de entrenamiento independientes.

Figura 21

Curva PR para el modelo DEIMv2 de tamaño nano



Nota. Se muestra la curva de Precision-Recall del modelo DEIMv2 de tamaño nano. Elaboración propia.

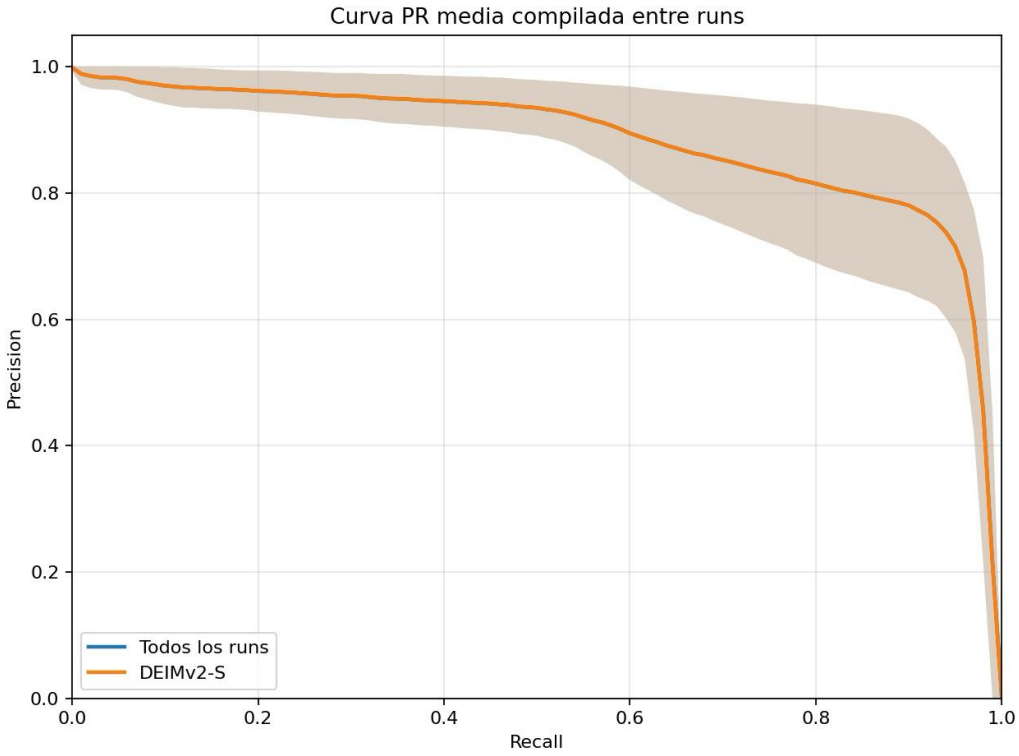
La curva de Precision–Recall promedio del modelo DEIMv2-Nano muestra un comportamiento estable en gran parte del rango de recall. La precisión se mantiene cercana a 0,95–1,0 en valores bajos de recall, lo que indica que el modelo presenta una baja tasa de falsos positivos cuando opera con umbrales de confianza altos. A medida que el recall aumenta, la precisión disminuye progresivamente, manteniéndose entre 0,80 y 0,90 para valores de recall entre 0,6 y 0,9, lo que evidencia un equilibrio adecuado entre la capacidad de detección y el control de falsas alarmas.

Asimismo, se observa una caída pronunciada de la precisión cuando el recall se aproxima a 1, lo cual es un comportamiento típico en modelos de detección de objetos que incluyen detecciones de menor confianza. La región sombreada indica la variabilidad entre corridas de entrenamiento y muestra una baja dispersión en gran parte del rango de recall, lo que sugiere consistencia en el desempeño del modelo entre diferentes ejecuciones.

En la **Figura *22*** se muestra la curva Precision-Recall del modelo evaluado de tamaño *small*, correspondiente al promedio de cinco ejecuciones de entrenamiento independientes. Adicionalmente, se reporta la banda de la desviación estándar.

Figura 22

Curva PR para el modelo DEIMv2 de tamaño small



Nota. Se muestra la curva de Precision-Recall del modelo DEIMv2 de tamaño small. Elaboración propia.

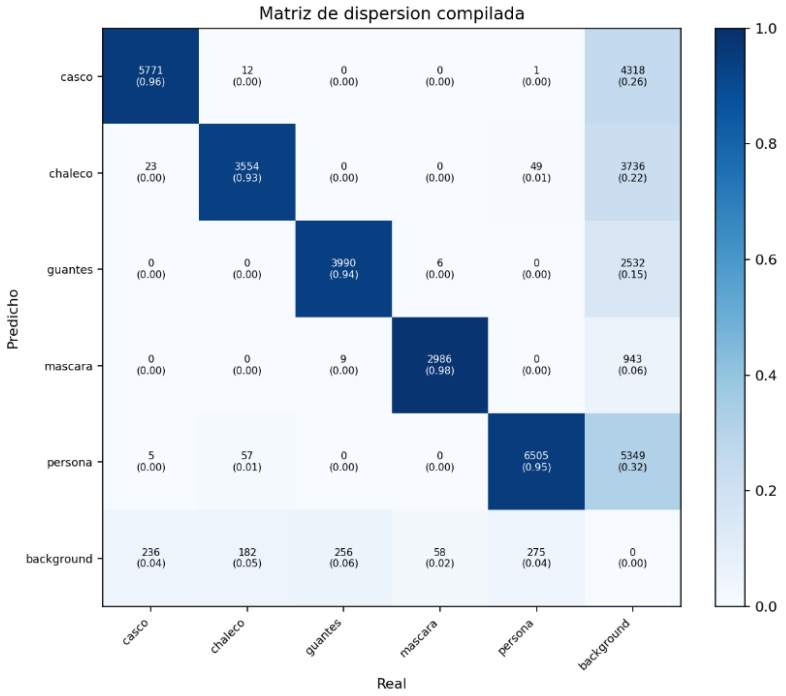
La curva de Precision–Recall promedio del modelo DEIMv2-Small presenta un comportamiento estable en la mayor parte del rango de recall. La precisión se mantiene cercana a valores de 0,95–1,0 en niveles bajos de recall, lo que indica que el modelo produce un número reducido de falsos positivos cuando opera con umbrales de confianza altos.

A medida que el recall aumenta, la precisión disminuye progresivamente, manteniéndose aproximadamente entre 0,80 y 0,90 para valores de recall entre 0,6 y 0,9, lo que refleja un equilibrio adecuado entre la capacidad de detección y el control de falsos positivos. Al igual que en otros detectores de objetos, se observa una caída pronunciada de la precisión cuando el recall se aproxima a 1, lo que se asocia a la

inclusión de detecciones de menor confianza, que incrementan la tasa de falsos positivos. La banda de dispersión entre corridas de entrenamiento indica una variabilidad moderada en valores altos de recall, mientras que en la mayor parte del rango el comportamiento del modelo es consistente, lo que sugiere estabilidad en el proceso de entrenamiento.

Figura 23

Matriz de confusión normalizada para DEIMv2 nano



Nota. Matriz de confusión normalizada del modelo DEIMv2 nano. Elaboración propia.

Igualmente, en las **Figura *23*** y **Figura *24*** se muestran, respectivamente, las matrices de confusión normalizadas para los modelos evaluados de tamaños *nano y small*, donde cada una corresponde al promedio de cinco ejecuciones de entrenamiento independientes.

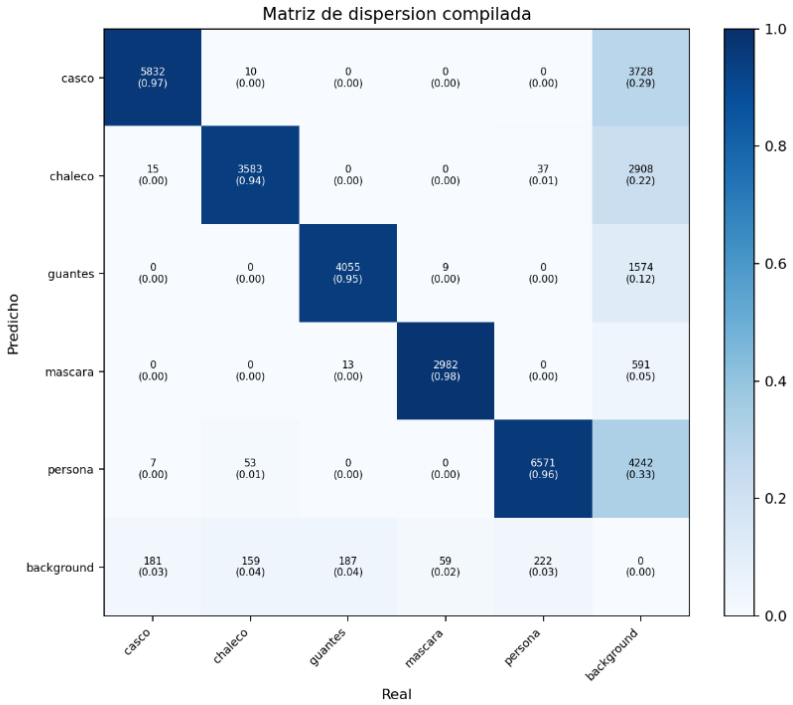
La matriz de confusión compilada muestra un alto nivel de aciertos en la diagonal principal, lo que indica que el modelo identifica correctamente la mayoría de las

instancias de cada clase evaluada. Las tasas de detección más altas se observan en máscara (≈0,98) y casco (≈0,96), seguidas por persona (≈0,95) y guantes (≈0,94), mientras que chaleco presenta una tasa de detección ligeramente menor (≈0,93).

La confusión entre clases es mínima, lo que indica que el modelo aprende representaciones visuales suficientemente discriminativas para diferenciar los distintos elementos de protección personal. Sin embargo, al igual que en otros detectores de objetos, la principal fuente de error corresponde a falsos negativos clasificados como fondo, particularmente en las clases guantes y chaleco. Además, se observan falsos positivos provenientes del fondo, especialmente en las clases persona, casco y chaleco, lo cual puede estar asociado a patrones visuales similares presentes en el entorno o a detecciones generadas en regiones con baja calidad visual.

Figura 24

Matriz de confusión normalizada para DEIMv2 small



Nota. Matriz de confusión normalizada del modelo DEIMv2 small. Elaboración propia.

La matriz de confusión muestra un alto nivel de aciertos en la diagonal principal, indicando que el modelo identifica correctamente la mayoría de las instancias de cada clase evaluada. Las tasas de detección más altas se observan en máscara (≈0,98), seguidas por casco (≈0,97), persona (≈0,96) y guantes (≈0,95), mientras que chaleco presenta un desempeño ligeramente menor (≈0,94).

La confusión entre clases es mínima, lo que indica que el modelo aprende representaciones visuales discriminativas adecuadas para diferenciar los distintos elementos de protección personal. La principal fuente de error corresponde a falsos negativos clasificados como fondo, en particular en las clases guante y casco. Asimismo, se observan falsos positivos provenientes del fondo, especialmente en las clases persona, casco y chaleco, lo cual puede estar asociado a patrones visuales similares presentes en el entorno o a detecciones generadas en regiones de baja calidad visual.

Análisis de Escalamiento entre Tamaños

Con este análisis se busca calcular el impacto de realizar un escalamiento del tamaño de arquitectura DEIMv2 nano a DEIMv2 small, para ello, calculamos las diferencias absolutas y relativas de cada par de modelos en sus diferentes tamaños:

Hallamos hallando la diferencia absoluta para los dos modelos usando la ecuación (8):

$$\Delta = 0{,}505041 - 0{,}490862 = 0{,}014179$$

Esto indica una diferencia absoluta apreciable de aproximadamente 1,4%. Ahora se puede hallar la diferencia porcentual relativa usando la ecuación (9):

$$\frac{0{,}014179}{0{,}505041} \times 100 = 2.8\%$$

El modelo de tamaño small mejora en aproximadamente 2.8% de mAP50-95 respecto al de tamaño nano, claramente inferior.

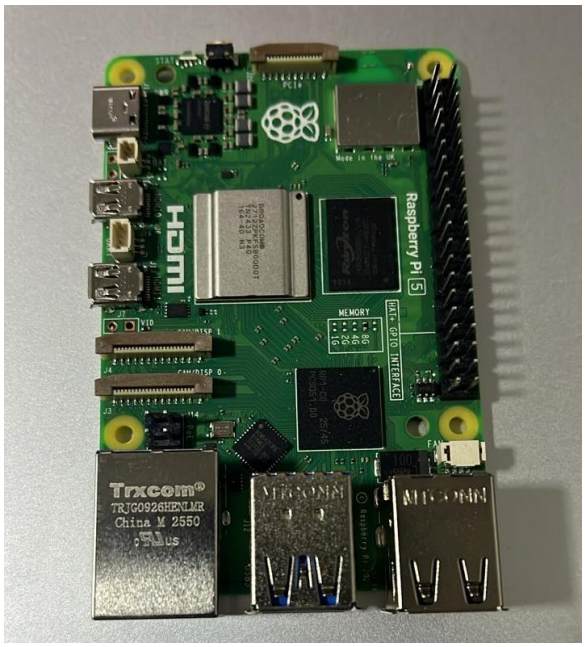
La comparación entre los modelos DEIMv2 Nano y DEIMv2 Small evidencia un comportamiento similar en términos generales de precisión y capacidad de detección, como se observa en la forma comparable de sus curvas Precision–Recall. No obstante, el modelo DEIMv2 Small presenta un desempeño ligeramente superior, reflejado en una mayor tasa de aciertos en la diagonal de la matriz de confusión para varias clases, en particular casco, chaleco y persona, donde se observa un incremento marginal en la proporción de detecciones correctas. Asimismo, el modelo Small muestra una leve reducción de los falsos negativos clasificados como fondo, lo que indica una mejor capacidad para recuperar instancias presentes en la escena. A pesar de estas mejoras, las diferencias entre ambos modelos no son sustanciales, lo que sugiere que el modelo DEIMv2 Nano mantiene un desempeño competitivo con menor complejidad computacional, mientras que el DEIMv2 Small ofrece una ligera ventaja en precisión y robustez, lo cual puede resultar beneficioso en escenarios donde los recursos de cómputo no constituyen una limitación crítica.

Pruebas de Desempeño en Dispositivos de Borde

Las pruebas de inferencia y desempeño computacional se realizaron en el dispositivo Raspberry Pi 5, lo que aseguró la uniformidad de las condiciones experimentales y permitió atribuir las diferencias de desempeño únicamente a las variaciones arquitectónicas entre los modelos evaluados. En el **Anexo I**, podemos ver las características de hardware del dispositivo.

Figura 25

Dispositivo Raspberry Pi 5



Nota. Se muestra una imagen del dispositivo Raspberry Pi 5 utilizado en el estudio. Elaboración propia.

Para cada modelo se evaluaron tres configuraciones de precisión numérica: Precisión Completa (FP32), Precisión Media (FP16) y Cuantización (INT8), lo que permitió analizar el impacto de técnicas de reducción de precisión tanto en el desempeño computacional como en la precisión predictiva.

Los modelos seleccionados fueron exportados en múltiples formatos, teniendo en cuenta las diferentes precisiones que soporta cada uno para su ejecución en la CPU (FP32, FP16, INT8). En el **Anexo J** se puede ver esta distribución.

Los formatos utilizados para cada caso se seleccionaron de acuerdo con la capacidad de sus entornos de ejecución correspondientes para soportar los modelos evaluados, así como con las recomendaciones de los autores sobre su uso.

Pruebas de Desempeño para Modelos YOLO en CPU

Para la ejecución de las pruebas de desempeño en CPU, se seleccionó el formato NCNN, ya que es totalmente de código abierto y ofrece un equilibrio entre la velocidad de inferencia y la precisión frente a ONNX. Para evaluar los modelos cuantizados (INT8) en CPU, se utilizó el formato MNN, ya que NCNN no admite actualmente este nivel de precisión. De esta manera, las diferencias observadas en el rendimiento se atribuyen exclusivamente a las arquitecturas evaluadas y a las configuraciones de precisión numérica.

Se realizaron 5 ejecuciones independientes en el dispositivo Raspberry PI 5, usando el subconjunto de pruebas (test set) del conjunto de datos principal, este subconjunto contiene un total de 763 imágenes. En el **Anexo K** se presenta el rendimiento promedio de los modelos evaluados bajo distintas configuraciones de precisión. Los valores reportados corresponden a la media y a la desviación estándar obtenidas a partir de múltiples ejecuciones experimentales.

Podemos verificar las variaciones del mAP en cada modelo debido al cambio de precisión, esto nos ayuda a evaluar su selección. En la **Tabla *18*** se muestran las variaciones del mAP y de la latencia según el tamaño del modelo para las precisiones FP32 y FP16.

Tabla 18

Tabla de diferencias variacionales para mAP y Latencia en YOLO

Modelo	FP32	FP16	Latencia FP32	Latencia FP16	∆mAP	∆Latencia
yolov11n	0,5532	0,5541	66,35	66,60	+0,0009	+0,25
yolov12n	0,5586	0,5587	142.71	145,49	+0,0001	+2.78
yolov11s	0,5739	0,5742	163,83	165,23	+0,0003	+1,4
yolov12s	0,5771	0,5778	345,75	342.27	+0,0007	-3,48
yolov8n	0,5629	0,5631	67,41	67,55	+0,0002	+0,14

Nota. Se muestran las diferencias variacionales de la mAP y la latencia. Elaboración propia.

En teoría, FP16 debería ofrecer una precisión igual o ligeramente inferior a la de FP32. Sin embargo, en la práctica pueden ocurrir pequeñas variaciones positivas como las anteriores, las cuales son extremadamente pequeñas (≈0,02–0,1 %), esto suele deberse a ruido numérico o variabilidad del pipeline, variación en Non-Maximum Suppression (NMS) o variabilidad del conjunto de datos, lo cual no afecta su validez.

En cuanto a la latencia, ocurre algo similar: se espera una disminución de esta en la mayoría de los casos, por lo que pueden verse diferencias muy pequeñas, de ≈ 0,14–2.8 ms. Esto se debe a limitaciones en la implementación de optimizaciones de CPU en los entornos de ejecución para FP16, y es posible que muchas operaciones se ejecuten sin optimización o se promuevan a FP32.

Tabla 19

Tabla de diferencias variacionales en el tamaño de modelos YOLO

Modelo	FP32	FP16	ΔTamaño
yolov11n	10 MB	5,1 MB	-4,9
yolov12n	10 MB	5,1 MB	-4,9
yolov11s	36,1 MB	18,2 MB	-17,9
yolov12s	35,4 MB	17,9 MB	-17,5
yolov8n	11,6 MB	5,9 MB	-5,7

Nota. Se muestran las diferencias variacionales en el tamaño de los modelos de YOLO. Elaboración propia.

Los resultados muestran que la reducción de la precisión a FP16 disminuye significativamente el tamaño del modelo (≈50 %), manteniendo prácticamente inalterada la precisión de detección, sin embargo, no se observaron mejoras significativas en la latencia de inferencia entre estos tipos de precisión en la Raspberry Pi 5.

Pruebas de Desempeño para Modelos RF-DETR en CPU

Para la ejecución de las pruebas de desempeño en la CPU del dispositivo Raspberry PI 5, tanto para las precisiones FP32 y FP16 como para INT8, se seleccionaron los formatos ONNX, OpenVINO y MNN. De esta manera, las diferencias observadas en el rendimiento se atribuyen exclusivamente a las arquitecturas evaluadas y a las configuraciones de precisión numérica.

Se realizaron 5 ejecuciones independientes en el dispositivo Raspberry PI 5, usando el subconjunto de pruebas (test set) del conjunto de datos principal, este subconjunto contiene un total de 763 imágenes. En la **Tabla *20*** se presenta el rendimiento promedio de los modelos evaluados bajo distintas configuraciones de precisión. Los valores reportados corresponden a la media y a la desviación estándar obtenidas a partir de múltiples ejecuciones experimentales.

Tabla 20

Tabla de métricas evaluadas para los diferentes tamaños de RF-DETR

Tamaño	Precisión	mAP50-95	Latencia (ms)	FPS	Tamaño (MB)
Nano	FP32	0,6017	1621,071 ±0,23	~0,617	107,2
	FP16	0,5850	2549,1 ±0,46	~0,392	53,8
	INT8	0,6061	2542.596 ±8,94	~0,393	30,2
Small	FP32	0,6076	1638,539 ±0,65	~0,610	113,1
	FP16	0,5930	2574,895 ±12.9	~0,39	56,8
	INT8	0,6058	2578,336 ±0,81	~0,39	31,7

Nota. Se muestran las métricas evaluadas con base en la precisión de cada tamaño de modelo. Elaboración propia.

Podemos verificar las variaciones del mAP en cada modelo debido al cambio de precisión, descartando la precisión FP16 debido a su bajo mAP50-95, su mayor latencia

y su mayor tamaño respecto a los demás, esto nos ayuda a evaluar la selección del mejor modelo. En la **Tabla *21*** se muestran las variaciones del mAP y de la latencia según el tamaño del modelo para las precisiones FP32 e INT8.

Tabla 21

Tabla de diferencias variacionales para mAP y latencia en RF-DETR

Modelo	FP32	INT8	Latencia FP32	Latencia FP16	∆mAP	∆Latencia
Nano	0,6017	0,6061	1621,071	2542.596	+0,0044	-921,525
Small	0,6076	0,6058	1638,539	2578,336	-0,0018	-939,797

Nota. Se muestran las diferencias variacionales de la mAP y la latencia para modelos RF-DETR. Elaboración propia.

Los resultados obtenidos para RF-DETR muestran que la cuantización a INT8 no produce mejoras en la latencia de inferencia, sino que incrementa significativamente el tiempo de ejecución (≈57%). Este comportamiento se atribuye a la naturaleza basada en transformers del modelo, cuyas operaciones de atención no están optimizadas para ejecutarse con precisión reducida en CPUs ARM. A pesar de que la precisión se mantiene estable, el incremento de la latencia limita su aplicabilidad en escenarios de edge computing, incluso en este tipo de aplicaciones.

Tabla 22

Tabla de diferencias variacionales en el tamaño de modelos RF-DETR

Modelo	FP32	FP16	∆Tamaño
Nano	107,2 MB	30,2 MB	-77,0
Small	113,1 MB	31,7 MB	-81,4

Nota. Se muestran las diferencias variacionales en el tamaño de los modelos de RF-DETR. Elaboración propia.

Los resultados muestran que la reducción de la precisión a INT8 disminuye significativamente el tamaño del modelo (≈57 %), manteniendo prácticamente inalterada la precisión de detección. Sin embargo, no se observaron mejoras en la latencia de inferencia de la Raspberry Pi 5 que por el contrario aumentó.

Pruebas de Desempeño para Modelos DEIMv2 en CPU

Para la ejecución de las pruebas de desempeño en la CPU del dispositivo Raspberry PI 5, tanto para las precisiones FP32 y FP16 como para INT8, se seleccionaron los formatos ONNX y OpenVINO, ya que NCNN no está soportado actualmente para este tipo de arquitecturas. De esta manera, las diferencias observadas en el rendimiento se atribuyen exclusivamente a las arquitecturas evaluadas y a las configuraciones de precisión numérica.

Se realizaron 5 ejecuciones independientes en el dispositivo Raspberry PI 5, usando el subconjunto de pruebas (test set) del conjunto de datos principal, este subconjunto contiene un total de 763 imágenes. En la **Tabla *23*** se presenta el rendimiento promedio de los modelos evaluados bajo distintas configuraciones de precisión. Los valores reportados corresponden a la media y a la desviación estándar obtenidas a partir de múltiples ejecuciones experimentales.

Tabla 23

Tabla de métricas evaluadas para los diferentes tamaños de DEIMv2

Tamaño	Precisión	mAP50-95	Latencia (ms)	FPS	Tamaño (MB)
Nano	FP32	0,5626	115,591 ±0,39	~8,65	14,1
	FP16	0,5630	115,721 ±0,46	~8,64	7,0
	INT8	0,5019	198,086 ±0,36	~5,04	3,8
Small	FP32	0,5856	657,895 ±0,32	~1,52	38,2
	FP16	0,5854	653,594 ±0,43	~1,53	19,1

Tamaño	Precisión	mAP50-95	Latencia (ms)	FPS	Tamaño (MB)
	INT8	0,5798	1144,732 ±1,26	~0,87	11,2

Nota. Se muestran las métricas evaluadas con base en la precisión de cada tamaño de modelo. Elaboración propia.

Podemos verificar las variaciones del mAP en cada modelo debido al cambio de precisión, descartando la precisión INT8 debido a su bajo mAP50-95 y su mayor latencia respecto a los demás, esto nos ayuda a evaluar la selección del mejor modelo. En la **Tabla *24*** se muestran las variaciones del mAP según la precisión.

Tabla 24

Tabla de diferencias variacionales para mAP y latencia en DEIMv2

Modelo	FP32	FP16	Latencia FP32	Latencia FP16	∆mAP	∆Latencia
Nano	0,5626	0,5630	115,591	115,721	+0,0004	-0,13
Small	0,5856	0,5854	657,895	653,594	-0,0002	-4,3

Nota. Se muestran las diferencias variacionales de la mAP y la latencia para modelos DEIMv2. Elaboración propia.

Los resultados obtenidos con los modelos DEIMv2 muestran que la reducción de la precisión a FP16 no genera cambios significativos ni en la precisión de detección (+0,023 mAP) ni en la latencia de inferencia. Esto se debe, muy posiblemente, a limitaciones en la implementación de optimizaciones de CPU en los entornos de ejecución de los formatos empleados. Asimismo, se observa que el modelo Small incrementa considerablemente la latencia (≈5,7×) con respecto al modelo Nano, mientras que la mejora en la precisión es limitada, lo que evidencia un compromiso desfavorable para su uso en dispositivos embebidos.

Tabla 25

Tabla de diferencias variacionales en el tamaño de modelos DEIMv2

Modelo	FP32	FP16	∆Tamaño
Small	14,1 MB	7,0 MB	-7,1
Nano	38,2 MB	19,1MB	-19,1

Nota. Se muestran las diferencias variacionales en el tamaño de los modelos de DEIMv2. Elaboración propia.

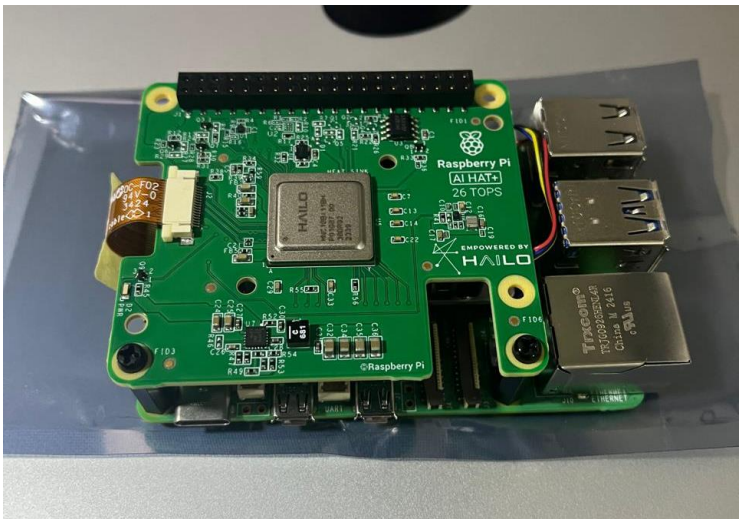
Los resultados muestran que la reducción de la precisión a FP16 disminuye significativamente el tamaño del modelo (≈50 %), manteniendo prácticamente inalterada la precisión de detección. Sin embargo, no se observaron mejoras significativas en la latencia de inferencia de la Raspberry Pi 5.

Pruebas de Desempeño para Modelos YOLO en NPU

Adicionalmente a las pruebas en CPU, se realizaron pruebas de desempeño computacional de los modelos YOLO en un módulo de procesamiento neural (NPU) compatible con dispositivos Raspberry PI 5. Se evaluaron únicamente los modelos YOLO, ya que el dispositivo NPU (Hailo8) disponible y compatible con Raspberry PI 5 en el momento de la realización de este estudio no soporta modelos basados en Transformers, sino solo modelos basados en CNN. En el **Anexo L**, se muestra un resumen de las características del módulo NPU utilizado:

Figura 26

Dispositivo NPU Hailo 8 HAT+



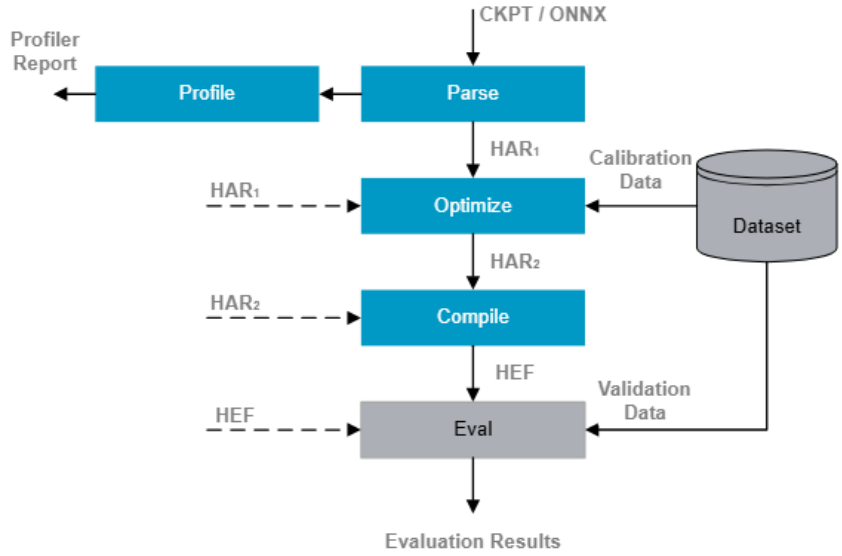
Nota. Dispositivo NPU Hailo 8 para Raspberry Pi 5. Elaboración propia.

Procedimiento de Cuantización de los Modelos

Para su ejecución en la unidad de procesamiento neural (NPU) Hailo 8 de 26 TOPS, se tomaron como base los modelos YOLO en formato ONNX, ya que este formato ofrece mayor portabilidad y compatibilidad con la plataforma de destino. Luego, se realizaron una conversión, una optimización y una cuantización utilizando el SDK y herramientas del fabricante, con el resultado de un modelo de cada candidato en formato HEF que corre directamente en este módulo de hardware. En la figura podemos ver un resumen del proceso de conversión, optimización y compilación (cuantización) de los modelos.

Figura 27

Diagrama del proceso de compilación para la NPU Hailo 8



Nota. Procedimiento de optimización, compilación y evaluación de modelos en la plataforma Hailo8. Tomado de “Getting started”, HAILO, 2025, Hailo Model Zoo (https://github.com/hailo-ai/hailo_model_zoo/).

Resultados de las Pruebas de Desempeño

Para ejecutar las pruebas de desempeño en la NPU Hailo8 del dispositivo Raspberry PI 5, se convierte el formato de los modelos base de ONNX a HEF, el único formato soportado por el dispositivo. Se realizaron 5 ejecuciones por cada modelo para evaluar tanto el mAP como la latencia de estos, los resultados los podemos ver en la tabla.

Tabla 26

Resultados de las pruebas de desempeño para modelos YOLO en NPU

Modelo	mAP50-95	Latencia	FPS
yolov11n	0,540	7,8367	~104,131
yolov12n	0,522	8,883	~134,060

Modelo	mAP50-95	Latencia	FPS
yolov11s	0,556	17,869	~43,905
yolov12s	0,485	24,854	~42,242
yolov8n	0,550	3,016	~591,727

Nota. Se muestran las diferencias métricas obtenidas mediante las pruebas de desempeño de los modelos YOLO en el NPU. Elaboración propia.

Cabe aclarar que el cálculo de FPS no corresponde a la medida de latencia en el ratio de los 1000 ms, dado que la NPU procesa las observaciones en un pipeline en paralelo y a nivel de hardware, por lo tanto, no es posible aplicar el cálculo tradicional de FPS por lo que esta medida de FPS es tomada directamente de los resultados arrojados por el hardware en las pruebas de rendimiento.

En particular, el modelo YOLOv8n presenta el mejor desempeño global, alcanzando una latencia de aproximadamente 3 ms y un rendimiento cercano a 592 FPS, manteniendo una precisión competitiva (mAP50-95 ≈ 0,55). Los modelos YOLOv11n y YOLOv12n muestran latencias ligeramente superiores (≈7–9 ms), con un desempeño aún adecuado para tiempo real (≈100–134 FPS), aunque con una leve reducción de la precisión en el caso de YOLOv12n.

Por otro lado, las variantes small (YOLOv11s y YOLOv12s) presentan un incremento considerable en la latencia (≈18–25 ms) sin mejoras proporcionales en la precisión, lo que reduce su eficiencia en comparación con las variantes nano.

Análisis de los Resultados

En los resultados experimentales obtenidos a partir del entrenamiento y la evaluación de modelos de detección de objetos en dispositivos embebidos de bajo costo, como la Raspberry PI 5, se analizaron tanto las métricas de precisión como el rendimiento computacional, considerando diferentes arquitecturas y configuraciones de precisión

numérica. Se busca identificar el modelo que ofrece el mejor equilibrio entre precisión y eficiencia en escenarios de edge computing.

Resultados de Precisión en Entrenamiento

Los modelos evaluados incluyen arquitecturas basadas en redes convolucionales (YOLO), modelos híbridos (DEIMv2) y modelos basados en transformers (RF-DETR). La evaluación se realizó utilizando la métrica mAP50-95.

Podemos resumir los resultados de las pruebas en lo siguiente:

- RF-DETR obtuvo los valores más altos de mAP (~0,60–0,61)
- DEIMv2 mostró un desempeño intermedio (~0,56–0,58)
- YOLO presentó valores ligeramente inferiores (~0,55–0,57)

A pesar de las diferencias observadas, la variación en mAP entre arquitecturas se mantiene en un rango reducido (≈3 - 5%), lo que sugiere que las mejoras en la precisión son marginales frente a cambios significativos en la complejidad computacional.

Resultados de Rendimiento en Dispositivo Edge

Las pruebas de inferencia se realizaron en el dispositivo Raspberry Pi 5, midiendo la latencia promedio por imagen. Dado que la precisión FP32 muestra resultados evidentemente mejores en las pruebas de rendimiento realizadas, a continuación, se muestra un resumen comparativo del comportamiento de la latencia para cada arquitectura:

- **YOLO:** 66–163 ms
- **DEIMv2:** 115–657 ms
- **RF-DETR:** 1621–2578 ms

Los resultados evidencian diferencias significativas en el tiempo de inferencia, con incrementos de hasta 25 veces entre los modelos YOLO y RF-DETR.

Aunque RF-DETR alcanza una mayor precisión, su elevado costo computacional lo hace inviable para aplicaciones en tiempo real en dispositivos embebidos. Por el contrario, YOLO mantiene un equilibrio adecuado entre precisión y velocidad.

Impacto de la Reducción de Precisión (FP16 e INT8)

Los resultados muestran que la reducción a FP16 no produce cambios significativos ni en la precisión ni en la latencia. Esto se debe a la ausencia de aceleración en los entornos de ejecución específicos disponibles para operaciones en FP16 en CPU ARM.

Por ejemplo, en YOLO se observa un comportamiento inesperado en el que no se evidencian mejoras claras en la latencia con INT8, y en el caso de RF-DETR se observa un aumento significativo de la latencia del ~57%.

En modelos basados en transformers, la cuantización introduce sobrecarga debido a conversiones entre formatos numéricos, lo que afecta negativamente el rendimiento.

Análisis de Eficiencia Computacional

Para este análisis, utilizaremos la siguiente métrica que nos ayuda a evaluar la eficiencia de cada arquitectura:

$$Eficiencia = \frac{mAP}{Latencia} \tag{10}$$

A continuación, se muestra un resumen de los resultados de esta métrica sobre cada una de las arquitecturas:

- **YOLO:** 0,0083, mayor eficiencia
- **DEIMv2:** 0,0049, eficiencia intermedia

- **RF-DETR:** 0,00037, muy baja eficiencia

Podemos observar que YOLO es ~2 veces más eficiente que DEIMv2 y ~20 veces más eficiente que RF-DETR. Esto demuestra que pequeñas mejoras en la precisión pueden implicar incrementos desproporcionados en el costo computacional.

Dados los resultados obtenidos, la **Tabla *27*** presenta un resumen de las dos métricas principales, ordenadas por la eficiencia de cada modelo.

Tabla 27

Resultados de las pruebas ordenadas por modelo y eficiencia

Modelo	mAP50-95	Latencia (ms)
YOLOv11n	~0,55	66,35
YOLOv11s	~0,57	163,83
DEIMv2 Nano	~0,56	115,591
DEIMv2 Small	~0,58	657,895
RF-DETR Nano	~0,60	1621,071
RF-DETR Small	~0,61	1638,539

Nota. Se muestran los modelos ordenados por la eficiencia obtenida, junto con las métricas principales de cada uno. Elaboración propia.

Dados estos resultados, se aplica una prueba no paramétrica de Friedman para comparar las métricas de eficiencia entre estos modelos, al tratarse de mediciones relacionadas bajo condiciones experimentales controladas y sin asumir normalidad.

$$X_F^2 = \frac{12}{nk(k+1)} \sum R_j^2 - 3n(k+1) \quad (11)$$

Donde:

n = 5 (ejecuciones)

k = 6 modelos

Rj = suma de rangos.

Para el tamaño del efecto se aplica Kendall's mediante la siguiente fórmula:

$$W = \frac{X^2}{n(k-1)} \tag{12}$$

Tras la identificación de diferencias globales mediante la prueba de Friedman, se aplicó la prueba post-hoc de Nemenyi para realizar comparaciones múltiples entre modelos, dado que este método es adecuado para datos relacionados y controla el error de tipo I en comparaciones múltiples.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{13}$$

Donde:

qα = valor crítico de la distribución Studentized range

k = número de grupos (modelos)

N = número de bloques (ejecuciones)

Aplicando estas pruebas, se obtienen los siguientes resultados:

$$X_F^2(5) \approx 25.0$$

$$p < 0.0001$$

$$W = 1.0$$

Los resultados evidencian diferencias estadísticamente significativas, $\chi^2(5) = 25.00$, $p < 0.001$, con un tamaño del efecto máximo (Kendall's W = 1.00), lo que indica una consistencia total en el orden de desempeño de los modelos a lo largo de las ejecuciones.

Los siguientes son los resultados del análisis post-hoc.

Tabla 28

Resultados de la prueba post-hoc para los modelos evaluados

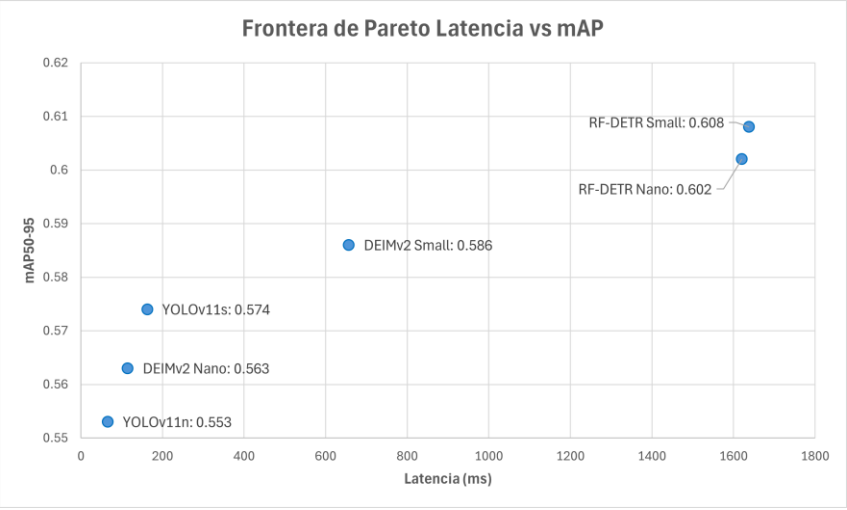
Modelo	YOLOv11n	YOLOv11s	DEIMv2-N	DEIMv2-S	RFDETR-N	RFDETR-S
YOLOv11n	-	0,538193	0,958997	0,113891	0,009435	0,000343
YOLOv11s	0,538193	-	0,958997	0,958997	0,538193	0,113891
DEIMv2-N	0,958997	0,958997	-	0,538193	0,113891	0,009435
DEIMv2-S	0,113891	0,958997	0,538193	-	0,958997	0,538193
RFDETR-N	0,009435	0,538193	0,113891	0,958997	-	0,958997
RFDETR-S	0,000343	0,113891	0,009435	0,538193	0,958997	-

Nota. Se muestran los resultados de la prueba post-hoc de Nemenyi sobre cada uno de los modelos. Elaboración propia.

Dado que valores menores de latencia indican un mejor desempeño, los resultados muestran que los modelos YOLOv11, en particular la variante nano, presentan el mejor rendimiento temporal, mientras que los modelos basados en RF-DETR exhiben los mayores tiempos de inferencia.

Figura 28

Gráfico de frontera de Pareto para latencia vs precisión (mAP)



Nota. Gráfico de frontera de Pareto que muestra, de izquierda a derecha, los modelos con mejor rendimiento computacional, considerando la relación entre la latencia y la precisión (mAP). Elaboración propia.

Adicionalmente, en la **Figura *28*** podemos ver los resultados de una forma más clara utilizando un gráfico de frontera de Pareto para la Latencia vs Precisión (mAP). Se puede observar que los modelos YOLO se ubican en la región de alta eficiencia (baja latencia), asimismo, DEIMv2 representa soluciones intermedias y, finalmente, RF-DETR maximiza la precisión, pero con alto costo computacional. Esto demuestra que pequeñas mejoras en la precisión pueden implicar incrementos desproporcionados en el costo computacional.

Análisis de Resultados para Modelos YOLO en NPU

Los resultados obtenidos evidencian una aceleración significativa en la inferencia de modelos YOLO al utilizar la NPU Hailo-8, con mejoras sustanciales en latencia y FPS respecto a la ejecución en CPU. En particular, el modelo YOLOv8n presenta el mejor desempeño global, alcanzando una latencia de aproximadamente 3 ms y un rendimiento cercano a 592 FPS, manteniendo una precisión competitiva (mAP50-95 ≈ 0,55). Esto lo posiciona como la opción más eficiente para aplicaciones en tiempo real que usan NPU.

Los modelos YOLOv11n y YOLOv12n muestran latencias ligeramente superiores (≈7–9 ms), con un desempeño aún adecuado para tiempo real (≈100–134 FPS), aunque con una leve reducción de la precisión en el caso de YOLOv12n.

Por otro lado, las variantes small (YOLOv11s y YOLOv12s) presentan un incremento considerable en la latencia (≈18–25 ms) sin mejoras proporcionales en la precisión, lo que reduce su eficiencia en comparación con las variantes nano. Adicionalmente, se observa que YOLOv12 no supera de forma consistente a YOLOv11 e incluso presenta

menor precisión en algunos casos, lo que sugiere que las mejoras arquitectónicas no están optimizadas para la ejecución en NPU.

Tabla 29

Comparación de resultados en CPU vs NPU (YOLO)

Modelo	mAP	Latencia CPU	Latencia NPU	Speed-up
YOLOv8n	0,563	67,4	3,0	~22×
YOLOv11n	0,553	66,3	7,8	~8,5×
YOLOv12n	0,559	142.7	8,9	~16×
YOLOv11s	0,574	163,8	17,8	~9×
YOLOv12s	0,577	345,7	24,8	~14×

Nota. Se muestra una comparación de resultados de ejecución de los modelos YOLO en CPU vs NPU. Elaboración propia.

Finalmente, es importante destacar que la NPU Hailo-8 no soporta modelos basados en transformers, lo que limita su uso a arquitecturas tipo CNN, en las que demuestra un rendimiento altamente eficiente.

Resumen del Análisis

Los resultados obtenidos confirman que la elección del modelo en sistemas de visión computacional para edge computing debe considerar un enfoque multiobjetivo. Sin embargo, se destacan los siguientes hallazgos:

- Las arquitecturas más complejas no garantizan mejoras proporcionales en precisión
- El costo computacional crece significativamente con modelos basados en transformers
- Las técnicas de cuantización no siempre son efectivas en CPU de propósito general

- Los modelos YOLO presentan el mejor equilibrio entre precisión y eficiencia

En particular, se evidencia que el uso de modelos como RF-DETR en dispositivos embebidos sin aceleración especializada resulta poco viable, pese a su mayor precisión.

Para aplicaciones en tiempo real, como la detección de elementos de protección personal, es fundamental garantizar tiempos de respuesta adecuados. Por lo que podemos concluir en este contexto, lo siguiente:

- YOLO permite inferencia cercana a tiempo real
- DEIMv2 puede ser viable en escenarios menos exigentes
- RF-DETR requiere hardware más potente

Según los resultados de YOLO, para los modelos de tamaño *Small* se observa una mejora de aproximadamente 1,63% respecto a *Nano*. Eso es una mejora pequeña pero real. Los resultados evidencian que la optimización de modelos para edge computing no depende únicamente de la precisión alcanzada, sino también de la eficiencia computacional global del sistema, la cual está directamente influenciada por la arquitectura del modelo y las capacidades del hardware de ejecución.

En conclusión, los modelos YOLO se posicionan como la alternativa más adecuada para su despliegue en dispositivos embebidos de bajo costo, al ofrecer el mejor equilibrio entre precisión y rendimiento computacional. Si bien modelos más complejos, como RF-DETR, alcanzan niveles más altos de precisión, su elevado costo computacional limita su aplicabilidad práctica en escenarios de edge computing.

Discusión

En primer lugar, se evidenció que la arquitectura del modelo tiene un impacto determinante en su desempeño al desplegarse en dispositivos embebidos. Los modelos basados en redes convolucionales, particularmente las variantes de YOLO, presentaron el mejor equilibrio entre precisión y latencia, lo que los posiciona como la opción más adecuada para aplicaciones en tiempo real. Por el contrario, los modelos híbridos como DEIMv2, si bien ofrecen mejoras marginales en la precisión, introducen incrementos considerables en el tiempo de inferencia, especialmente en sus versiones de mayor tamaño. Asimismo, los modelos basados en transformers, como RF-DETR, alcanzaron los mayores valores de mAP, pero a costa de latencias significativamente superiores, lo que limita su aplicabilidad en escenarios de edge computing.

En relación con lo anterior, se concluye que las arquitecturas más complejas no necesariamente garantizan un mejor desempeño en dispositivos embebidos, ya que el incremento de la precisión suele ser marginal frente al aumento exponencial del costo computacional. Este comportamiento fue particularmente evidente en los modelos basados en transformers, que presentaron latencias hasta 25 veces mayores que las de los modelos YOLO al ejecutarse en CPU. Adicionalmente, se observó que estos modelos no son compatibles con la NPU empleada, lo que limita aún más su aplicabilidad en entornos reales.

Otro aspecto relevante identificado en este estudio es la importancia del trade-off entre la precisión y la latencia. Los resultados muestran que mejoras relativamente pequeñas en la precisión, del orden de 3% a 5%, pueden implicar incrementos sustanciales en el tiempo de inferencia. Esto confirma que la selección de modelos en contextos de edge computing no debe basarse únicamente en la maximización de la precisión, sino en un enfoque multiobjetivo que considere simultáneamente la eficiencia computacional.

En este sentido, se evidenció que los modelos ligeros, en particular las variantes nano, presentan una ventaja significativa en eficiencia. Estos modelos logran mantener niveles de precisión competitivos mientras reducen considerablemente la latencia, lo que los hace especialmente adecuados para aplicaciones en tiempo real en dispositivos embebidos de bajo costo. Por el contrario, los modelos de mayor tamaño no justifican su incremento de complejidad desde una perspectiva práctica, dado que la ganancia en precisión es limitada.

En cuanto a las técnicas de optimización mediante la reducción de la precisión, se observó que su impacto en la CPU ARM es limitado. La utilización de FP16 permitió mantener la precisión sin generar mejoras significativas en la latencia, mientras que la cuantización a INT8 presentó resultados inconsistentes e incluso degradaciones del rendimiento en algunos modelos, especialmente en aquellos basados en arquitecturas más complejas. Esto pone de manifiesto que la efectividad de estas técnicas depende no solo del modelo, sino también de los entornos de ejecución, del aprovechamiento de las optimizaciones disponibles en la CPU por parte de estos y de las características del hardware de ejecución.

Por otro lado, uno de los hallazgos más relevantes de este trabajo fue el impacto del hardware especializado en el rendimiento de los modelos. La implementación en la NPU Hailo-8 permitió reducir la latencia de inferencia hasta en un factor de 22 veces en comparación con la ejecución en CPU, manteniendo niveles de precisión similares. Este resultado demuestra que la eficiencia en sistemas de visión por computador en edge computing depende en gran medida de la disponibilidad de aceleradores de hardware adecuados. No obstante, también se identificó como una limitación la falta de soporte de la NPU para modelos basados en transformers, lo que limita su uso a arquitecturas de tipo CNN.

Adicionalmente, el análisis mediante la frontera de Pareto permitió identificar con claridad los modelos que ofrecen el mejor equilibrio entre precisión y latencia. En este análisis, los modelos YOLO se ubicaron de forma consistente en la región de mayor eficiencia, mientras que los modelos más complejos, aunque más precisos, resultaron dominados en términos de eficiencia computacional. Esto valida la utilidad de enfoques multiobjetivos para la selección de modelos en escenarios de edge computing.

Los resultados obtenidos permiten concluir que el desempeño de los sistemas de detección de objetos en dispositivos embebidos no depende únicamente de la arquitectura del modelo, sino también de la interacción entre múltiples factores, entre ellos la complejidad computacional, las técnicas de optimización empleadas y las capacidades del hardware de ejecución. En este contexto, la co-optimización entre modelo y hardware se presenta como un elemento clave para el desarrollo de soluciones eficientes. En conjunto, los resultados demuestran que la selección de modelos para entornos edge debe abordarse como un problema multiobjetivo, en el que el equilibrio entre precisión y latencia resulta más relevante que la optimización de una sola métrica.

Finalmente, los modelos YOLO en sus variantes ligeras se posicionan como la alternativa más adecuada para su implementación en dispositivos embebidos de bajo costo, al ofrecer el mejor equilibrio entre precisión y eficiencia computacional. Si bien modelos más complejos, como RF-DETR, alcanzan niveles más altos de precisión, su elevado costo computacional y sus limitaciones de compatibilidad con hardware especializado restringen su aplicabilidad práctica en escenarios de edge computing.

Conclusiones y Trabajo Futuro

En el presente trabajo se llevó a cabo una evaluación integral de diferentes arquitecturas de detección de objetos orientadas a su implementación en dispositivos embebidos de bajo costo, considerando tanto métricas de precisión como de eficiencia computacional. Los resultados obtenidos permiten establecer una serie de conclusiones relevantes en el contexto de edge computing y visión por computador.

Conclusiones

En la presente investigación se analiza comparativamente el rendimiento de diferentes arquitecturas de detección de objetos aplicadas a la identificación de elementos de protección personal en dispositivos de borde de bajo costo, evidenciando diferencias estadísticamente significativas tanto en precisión como en eficiencia computacional.

El análisis ANOVA factorial evidenció efectos significativos de la arquitectura del modelo ($F(3,32)=53.28$, $p<0.001$) y del tamaño ($F(1,32)=284.72$, $p<0.001$), así como una interacción significativa entre ambos factores ($p\approx0.042$), lo que confirma que el rendimiento depende tanto del tipo de arquitectura como de su escala.

En relación con la hipótesis sobre el desempeño (precisión), en la que se planteó que el rendimiento medido mediante mAP@0.5:0.95 presenta variaciones entre arquitecturas, los resultados obtenidos permiten rechazar la hipótesis nula (H_{01}), que establecía la ausencia de diferencias significativas, y aceptar la hipótesis alternativa (H_{11}). En efecto, se evidenciaron diferencias estadísticamente significativas ($p<0.05$) en el desempeño de los modelos, las cuales además pueden ser caracterizadas en términos de magnitud y variabilidad. En términos prácticos, los modelos alcanzaron tasas de detección superiores al 90% en la mayoría de las clases, con valores cercanos a ≈0.97–0.98 en máscara y ≈0.92–0.95 en guantes, mientras que otras clases como chaleco presentaron

desempeños ligeramente inferiores (≈0.90), evidenciando diferencias reales pero acotadas entre arquitecturas.

En términos de precisión, los modelos evaluados demostraron un alto nivel de desempeño en la detección de elementos de protección personal, alcanzando tasas de acierto superiores al 90% en la mayoría de las clases. Específicamente, se observaron valores cercanos a ≈0.97–0.98 en la clase máscara y ≈0.92–0.95 en guantes, seguidos por persona (≈0.94–0.95) y casco (≈0.94), mientras que la clase chaleco presentó un desempeño ligeramente inferior (≈0.90). Estos resultados evidencian una alta capacidad de discriminación entre clases, con errores concentrados principalmente en falsos negativos asociados al fondo.

Los resultados obtenidos permitieron identificar diferencias significativas en el desempeño de las arquitecturas evaluadas, evidenciando compromisos entre precisión y eficiencia. En particular, el análisis de escalamiento mostró que el incremento del tamaño de los modelos genera mejoras marginales en precisión, del orden de ≈0.58% en arquitecturas tipo transformer y ≈1.63% en modelos YOLO, lo que indica un comportamiento de rendimientos decrecientes frente al aumento del costo computacional.

En relación con la hipótesis sobre la eficiencia (latencia), los resultados permiten igualmente rechazar la hipótesis nula (H_{02}) y aceptar la hipótesis alternativa (H_{12}), al evidenciarse diferencias estadísticamente significativas en los tiempos de inferencia entre los modelos evaluados. Estas diferencias no solo son significativas, sino también relevantes en términos prácticos, ya que los modelos YOLO alcanzaron latencias cercanas a ≈3 ms en NPU frente a valores de ≈67 ms en CPU, logrando aceleraciones de hasta ~22×. Asimismo, se observaron latencias en el rango de ≈7–9 ms (≈100–134 FPS) en configuraciones optimizadas, mientras que las variantes de mayor tamaño (small) incrementan la latencia hasta ≈18–25 ms sin mejoras proporcionales en precisión,

lo que evidencia diferencias claras en magnitud y eficiencia computacional entre arquitecturas.

Finalmente, se concluye que ciertas arquitecturas optimizadas ofrecen un equilibrio adecuado para su implementación en sistemas de bajo costo. En particular, los modelos de la familia YOLO destacan por ofrecer un balance favorable entre precisión (mAP ≈0.55–0.57) y eficiencia computacional, mientras que arquitecturas más complejas, como las basadas en Transformers, presentan limitaciones en entornos de edge computing debido a sus mayores requerimientos de hardware. Estos resultados aportan criterios técnicos relevantes para el despliegue de soluciones de seguridad industrial basadas en dispositivos de borde.

Trabajo Futuro

A partir de los resultados obtenidos en este trabajo, se identifican diversas líneas de investigación futuras orientadas a mejorar el desempeño de los modelos de visión por computador en dispositivos embebidos de bajo costo.

En primer lugar, se propone explorar la implementación de modelos basados en transformers en hardware especializado que soporte estas arquitecturas, como GPUs embebidas o NPUs con capacidades avanzadas. Esto permitiría evaluar si las ventajas de precisión observadas en modelos como RF-DETR pueden aprovecharse sin incurrir en penalizaciones significativas de latencia. Ya en el primer semestre de 2026, el fabricante Hailo planea liberar una nueva versión del dispositivo NPU que soporta modelos basados en Transformers, lo que podría ofrecer una visión más completa de las capacidades reales de estos nuevos modelos. Otra potencial fuente de exploración, muy cercana al límite especificado como de bajo costo en este estudio, son los dispositivos

NVIDIA Jetson Nano, los cuales cuentan con una NPU incorporada con mucha más potencia, y que se muestran como un competidor fuerte en el mercado.

En segundo lugar, se recomienda investigar técnicas avanzadas de optimización de modelos, tales como el entrenamiento con *quantization-aware training* (QAT), el *pruning* estructurado y la destilación del conocimiento, con el objetivo de reducir la complejidad computacional sin afectar significativamente la precisión. Estas técnicas podrían permitir que modelos más complejos sean viables en entornos edge.

Adicionalmente, sería relevante analizar el impacto del uso de diferentes motores de inferencia (como OpenVINO, TensorRT o TFLite) en el rendimiento de los modelos, dado que la eficiencia computacional no solo depende de la arquitectura, sino también del framework de ejecución.

Otra línea de investigación consiste en la evaluación del consumo energético de los modelos, lo que permite incorporar métricas como la eficiencia energética (mAP/Watt), fundamentales en aplicaciones reales de edge computing.

Asimismo, se propone ampliar el estudio a diferentes dispositivos embebidos, incluyendo plataformas heterogéneas con CPU, GPU y NPU, a fin de analizar el comportamiento de los modelos bajo distintas configuraciones de hardware.

Finalmente, se sugiere evaluar estos modelos en escenarios de aplicación reales, considerando variables como la iluminación, las oclusiones y los cambios en el entorno, con el objetivo de validar su robustez en situaciones del mundo real.

Referencias

Arias, F. G. (2012). *El Proyecto de Investigación. Introducción a la Metodología Científica. 6ta. Edición* (6ta. Edición). FIDIAS G. ARIAS ODÓN. https://books.google.com.co/books?id=W5n0BgAAQBAJ

Arias, J. L. (2020). Proyecto de Tesis: Guia para la elaboración. In *Repositorio CONCYTEC*. https://hdl.handle.net/20.500.12390/2236

Avila-Montealegre, O., Bonilla, L., Botero-García, J. A., Caicedo-García, E., Dávalos, E., Flórez, L. A., Gómez-Pineda, J. G., Grajales-Olarte, A., Guarín-López, A., Hamann, F., Hermida-Giraldo, D., Julio-Román, J. M., Lasso-Valderrama, F. J., Martínez-Cortés, N., Méndez-Vizcaíno, J. C., Morales-Zurita, L. F., Ospina-Tejeiro, J. J., Pulido-Mahecha, K. L., Ramos-Veloza, M. A., … Arango-Thomas, L. E. (2021). Efectos macroeconómicos del salario mínimo en Colombia. *Ensayos Sobre Politica Economica*, *2021*(103), 1–117. https://doi.org/10.32468/ESPE103

Bhojanapalli, S., Wilber, K., Veit, A., Rawat, A. S., Kim, S., Menon, A., & Kumar, S. (2021). *On the Reproducibility of Neural Network Predictions*. http://arxiv.org/abs/2102.03349

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. https://arxiv.org/pdf/2004.10934

Bogusław Cyganek. (2013). Object Detection and Recognition in Digital Images. *Object Detection and Recognition in Digital Images*. https://doi.org/10.1002/9781118618387

Brue, S. L. ., & Grant, R. R. . (2007). *The evolution of economic thought*. 543. https://books.google.com/books/about/The_Evolution_of_Economic_Thought.html?id=gOQzAAAACAAJ

Cabrejos, J. A. L., & Roman-Gonzalez, A. (2023). Artificial Intelligence System for Detecting the Use of Personal Protective Equipment. *International Journal of*

Advanced Computer Science and Applications, *14*(5), 580–585. https://doi.org/10.14569/IJACSA.2023.0140561

Chan, K. Y., Abu-Salih, B., Qaddoura, R., Al-Zoubi, A. M., Palade, V., Pham, D. S., Ser, J. Del, & Muhammad, K. (2023). Deep neural networks in the cloud: Review, applications, challenges and research directions. *Neurocomputing*, *545*, 126327. https://doi.org/10.1016/J.NEUCOM.2023.126327

Chen, E., Wang, H., Shi, Z., & Zhang, W. (2025). Channel pruning for convolutional neural networks using l0-norm constraints. *Neurocomputing*, *636*, 129925. https://doi.org/10.1016/J.NEUCOM.2025.129925

Chen, Y., Wang, S., Lin, L., Cui, Z., & Zong, Y. (2024). Computer Vision and Deep Learning Transforming Image Recognition and Beyond. *International Journal of Computer Science and Information Technology*, *2*(1), 45–51. https://doi.org/10.62051/IJCSIT.V2N1.06

Decreto 1072 de 2015 (2015). https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=72173

Decreto 1572 de 2024 (2024). https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=257156

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. http://arxiv.org/abs/2010.11929

Fortier, P. J., & Michel, H. E. (2003). Fundamental Concepts and Performance Measures. *Computer Systems Performance Evaluation and Prediction*, 107–126. https://doi.org/10.1016/B978-155558260-9/50003-5

França, R. P., Borges Monteiro, A. C., Arthur, R., & Iano, Y. (2021). An overview of deep learning in big data, image, and signal processing in the modern digital age. *Trends*

in Deep Learning Methodologies: Algorithms, Applications, and Systems, 63–87. https://doi.org/10.1016/B978-0-12-822226-3.00003-9

Guía Para La Identificación de Peligros, Valoración de Riesgos y Determinación de Controles, Pub. L. GTHG01 (2024). https://www.minsalud.gov.co/Ministerio/Institucional/Procesos%20y%20procedimientos/GTHG01.pdf

Hernández-Sampieri, R., & Mendoza, C. P. (2018). Metodología de la investigación. Las rutas cuantitativa, cualitativa y mixta Las rutas Cuantitativa Cualitativa y Mixta. In *McGRAW-HILL Interamericana Editores S.A. de C.V.* (Primera Edición). Mc Graw Hill Education. https://virtual.cuautitlan.unam.mx/rudics/?p=2612

Huang, S., Hou, Y., Liu, L., Yu, X., & Shen, X. (2026). *Real-Time Object Detection Meets DINOv3*. http://arxiv.org/abs/2509.20787

Huang, S., Lu, Z., Cun, X., Yu, Y., Zhou, X., & Shen, X. (2024). *DEIM: DETR with Improved Matching for Fast Convergence*. https://arxiv.org/pdf/2412.04234

Jegham, N., Koh, C. Y., Abdelatti, M., & Hendawi, A. (2024). *Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors*. https://arxiv.org/abs/2411.00201v1

Jiang, B., Chen, J., & Liu, Y. (2023). Single-shot pruning and quantization for hardware-friendly neural network acceleration. *Engineering Applications of Artificial Intelligence*, *126*, 106816. https://doi.org/10.1016/J.ENGAPPAI.2023.106816

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., & Bernstein, J. (2024). *Muon: an optimizer for hidden layers in neural networks.* Https://Kellerjordan.Github.Io/Posts/Muon/.

Kala, R. (2024). An introduction to machine learning and deep learning. *Autonomous Mobile Robots*, 569–625. https://doi.org/10.1016/B978-0-443-18908-1.00022-4

Kaur, R., & Singh, S. (2023). A comprehensive review of object detection with deep learning. *Digital Signal Processing*, *132*, 103812. https://doi.org/10.1016/J.DSP.2022.103812

Khanam, R., & Hussain, M. (2024a). *What is YOLOv5: A deep look into the internal features of the popular object detector*. https://arxiv.org/pdf/2407.20892

Khanam, R., & Hussain, M. (2024b). *YOLOv11: An Overview of the Key Architectural Enhancements*. https://arxiv.org/pdf/2410.17725

Khanday, N. Y., & Sofi, S. A. (2021). Taxonomy, state-of-the-art, challenges and applications of visual understanding: A review. *Computer Science Review*, *40*. https://doi.org/10.1016/j.cosrev.2021.100374

Khoshakhlagh, A. H., Malakoutikhah, M., Park, J. W., Kodnoueieh, M. D., Boroujeni, Z. R., Bahrami, M., & Ramezani, F. (2024). Assessing personal protective equipment usage and its correlation with knowledge, attitudes, performance, and safety culture among workers in small and medium-sized enterprises. *BMC Public Health*, *24*(1), 1–9. https://doi.org/10.1186/S12889-024-19517-3/FIGURES/2

Ley 9 de 1979 (1979). https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=1177

Ley 1562 de 2012 (2012). https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=48365

Ley 1581 de 2012, El Congreso de Colombia (2012). http://www.secretariasenado.gov.co/senado/basedoc/ley_1581_2012.html

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. https://arxiv.org/pdf/2209.02976

Li, S., Li, M., Li, R., He, C., & Zhang, L. (2023). *One-to-Few Label Assignment for End-to-End Dense Detection*. 7350–7359. https://doi.org/10.1109/cvpr52729.2023.00710

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8693 LNCS*(PART 5), 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. *Array*, *16*, 100258. https://doi.org/10.1016/J.ARRAY.2022.100258

Observatorio de Seguridad y Salud en el Trabajo. (2024). *Colombia registra la tasa de mortalidad laboral más baja en los últimos siete años, pero la accidentalidad no cede*. https://ccs.org.co/portfolio/informe-de-siniestralidad-laboral-del-primer-semestre-de-2024/

Peng, Y., Li, H., Wu, P., Zhang, Y., Sun, X., & Wu, F. (2024). *D-FINE: Redefine Regression Task in DETRs as Fine-grained Distribution Refinement*. https://arxiv.org/pdf/2410.13842

Powers, D. M. W., & Ailab. (2020). *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. https://arxiv.org/pdf/2010.16061

Raiaan, M. A. K., Sakib, S., Fahad, N. M., Mamun, A. Al, Rahman, M. A., Shatabda, S., & Mukta, M. S. H. (2024). A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks. *Decision Analytics Journal*, *11*, 100470. https://doi.org/10.1016/J.DAJOUR.2024.100470

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Computer Society Conference*

on Computer Vision and Pattern Recognition, *2016-December*, 779–788. https://doi.org/10.1109/CVPR.2016.91

REGLAMENTO (UE) 2024/1689 (2024). https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX:32024R1689

Resolución 312 de 2019 (2019). https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=82666

Resolución 2400 de 1979 (1979). https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=53565

Russell, S., & Norvig, P. (2021). *Artificial Intelligence, Global Edition A Modern Approach*. 1168. https://elibrary.pearson.de/book/99.150005/9781292401171

Sapkota, R., Cheppally, R. H., Sharda, A., & Karkee, M. (2026). *YOLO26: Key Architectural Enhancements and Performance Benchmarking for Real-Time Object Detection*. http://arxiv.org/abs/2509.25164

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. In *SN Computer Science* (Vol. 2, Number 3). https://doi.org/10.1007/s42979-021-00592-x

Schulz, J., Hu, S., Speidel, S., Seeling, P., & Fitzek, F. H. P. (2024). Negative Latency in Computer Vision: A Key to Efficient Edge Offloading. *Proceedings - IEEE Global Communications Conference, GLOBECOM*, 3110–3115. https://doi.org/10.1109/GLOBECOM52923.2024.10901398

Sehsah, R., El-Gilany, A. H., & Ibrahim, A. M. (2020). Personal protective equipment (PPE) use and its relation to accidents among construction workers. *La Medicina Del Lavoro*, *111*(4), 285. https://doi.org/10.23749/MDL.V111I4.9398

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L.,

Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., … Bojanowski, P. (2025). *DINOv3*. http://arxiv.org/abs/2508.10104

Spencer, B. F., Hoskere, V., & Narazaki, Y. (2019). Advances in Computer Vision-Based Civil Infrastructure Inspection and Monitoring. *Engineering*, *5*(2), 199–222. https://doi.org/10.1016/J.ENG.2018.11.030

Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (Second Edition). Springer International Publishing. https://doi.org/10.1007/978-3-030-34372-9

Tamayo y Tamayo, M. (2003). *El proceso de la investigación científica: incluye evaluación y administración de proyectos de investigación* (4 ed.). Limusa - Noriega.

Tian, Y., Ye, Q., & Doermann, D. (2025). *YOLOv12: Attention-Centric Real-Time Object Detectors*. http://arxiv.org/abs/2502.12524

Torralba, Antonio., Isola, Phillip., & Freeman William T., . (2024). *Foundations of computer vision*. The MIT Press.

Velasquez, R. A., Lara, J. V. M., & Velasquez, R. A. (2023). Yolo algorithms to improve the detection of personal protective equipment thought industrial computer vision. *Proceedings of the 2023 IEEE 30th International Conference on Electronics, Electrical Engineering and Computing, INTERCON 2023*. https://doi.org/10.1109/INTERCON59652.2023.10326066

Wan, Z. (2023). *Text Classification: A Perspective of Deep Learning Methods*. https://arxiv.org/pdf/2309.13761

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). *YOLOv10: Real-Time End-to-End Object Detection*. https://arxiv.org/pdf/2405.14458

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 7464–7475. https://doi.org/10.1109/cvpr52729.2023.00721

Wang, C.-Y., & Liao, H.-Y. M. (2024). *YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems*. https://doi.org/10.1561/116.20240058

Wang, S., Xia, C., Lv, F., & Shi, Y. (2024). *RT-DETRv3: Real-time End-to-End Object Detection with Hierarchical Dense Positive Supervision*. 11–13. https://arxiv.org/pdf/2409.08475

Xie, Z., Liu, H., Li, Z., & He, Y. (2018). A convolutional neural network based approach towards real-time hard hat detection. *Proceedings of the 2018 IEEE International Conference on Progress in Informatics and Computing, PIC 2018*, 430–434. https://doi.org/10.1109/PIC.2018.8706269

Yaseen, M. (2024). *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. https://arxiv.org/pdf/2408.15857

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into Deep Learning. *Journal of the American College of Radiology*, *17*(5), 637–638. https://doi.org/10.1016/j.jacr.2020.02.005

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2023). *DETRs Beat YOLOs on Real-time Object Detection*. https://doi.org/10.1109/CVPR52733.2024.01605

Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., & Feng, J. (2021). *DeepViT: Towards Deeper Vision Transformer*. https://arxiv.org/pdf/2103.11886

Anexos

Anexo A

Lista de los principales hiperparámetros configurados para el entrenamiento de los modelos YOLO

- **Epocas:** 120
- **Batch:** 16
- **Workers:** 8
- **Early stopping:** Sí,
- **Patience:** 35
- **Optimizer:** MuSGD (Muon + SGD)
- **Learning rate:** 0,01
- **Momentum:** 0,937
- **Exponential Moving Average (EMA):** Sí
- **Seed:** aleatorio

Anexo B

Lista de parámetros de aumento de datos y sus valores

Parámetro	Valor	Descripción
Geométricas		
degrees	0,0	Rango de rotación
translate	0,1	Traslación de la imagen
scale	0,5	Variación de escala
shear	0,0	Corte o *Shearing*
perspective	0,0	Transformación de perspectiva
Giros (flips)		
fliplr	0,5	Probabilidad de giro horizontal

Parámetro	Valor	Descripción
flipud	0,0	Probabilidad de giro vertical
Color		
hsv_h	0,015	Aumento de contraste
hsv_s	0,7	Aumento de contraste
hsv_v	0,4	Aumento de contraste
Otros		
mosaic	1,0	Mosaico de imágenes
mixup	0,0–0,1	Fusión de imágenes
copy_paste	0,0	Copia de instancia
erasing	0,4	Borrado aleatorio

Nota. Lista de parámetros de aumento de datos utilizados y sus valores correspondientes establecidos para el entrenamiento. Elaboración propia.

Anexo C

Resultados de los modelos entrenados de tamaño nano y small (YOLO)

Modelo	Ejecución 1	Ejecución 2	Ejecución 3	Ejecución 4	Ejecución 5	Promedio
YOLOv8n	0,58506	0,58009	0,58023	0,58817	0,57524	0,581758
YOLOv11n	0,57961	0,57820	0,58362	0,58087	0,58443	0,581346
YOLOv12n	0,58425	0,58613	0,58022	0,58284	0,58509	0,583706
YOLOv26n	0,56936	0,56580	0,56423	0,57229	0,56198	0,566732
YOLOv8s	0,59206	0,59574	0,59223	0,59407	0,59269	0,593358
YOLOv11s	0,59746	0,59502	0,59507	0,59677	0,59795	0,596454
YOLOv12s	0,59946	0,60024	0,59649	0,59762	0,59878	0,598518
YOLOv26s	0,58795	0,58676	0,58748	0,58274	0,58512	0,586010

Nota. Tabla de resultados de mAP@0.5:0.95 productos del entrenamiento de los diferentes modelos YOLO nano y small. Elaboración Propia.

Anexo D

Métricas estadísticas de los modelos entrenados de tamaño nano (YOLO)

Modelo	Métrica	Promedio	STD	IC bajo	IC alto	CV (%)
yolov8n	mAP50	0,861888	0,005041	0,855628	0,868148	0,58%
yolov8n	mAP50-95	0,581758	0,004990	0,575562	0,587954	0,86%
yolov8n	Precision	0,744864	0,012678	0,729123	0,760605	1,7%
yolov8n	Recall	0,880638	0,008275	0,870363	0,890913	0,94%
yolov11n	mAP50	0,862456	0,003518	0,858088	0,866824	0,40%
yolov11n	mAP50-95	0,581346	0,002637	0,578071	0,584621	0,45%
yolov11n	Precision	0,745302	0,007833	0,735576	0,755028	1,05%
yolov11n	Recall	0,873154	0,008541	0,862549	0,883759	0,98%
yolov12n	mAP50	0,864352	0,003487	0,860023	0,868681	0,40%
yolov12n	mAP50-95	0,583706	0,002291	0,580862	0,586550	0,39%
yolov12n	Precision	0,753666	0,007749	0,744044	0,763288	1,03%
yolov12n	Recall	0,880302	0,005849	0,873040	0,887564	0,66%
yolov26n	mAP50	0,836998	0,005576	0,830075	0,843921	0,67%
yolov26n	mAP50-95	0,566732	0,004108	0,561631	0,571833	0,72%
yolov26n	Precision	0,722530	0,006306	0,714700	0,730360	0,87%
yolov26n	Recall	0,852726	0,008267	0,842461	0,862991	0,97%

Nota. Tabla de métricas estadísticas del entrenamiento de los modelos YOLO de tamaño nano. Elaboración propia.

Anexo E

Métricas estadísticas de los modelos entrenados de tamaño small (YOLO)

Modelo	Métrica	Promedio	STD	IC inf.	IC sup.	CV (%)
yolov8s	mAP50	0,866096	0,002691	0,862755	0,869437	0,31%
yolov8s	mAP50-95	0,593358	0,001548	0,591436	0,595280	0,26%

Modelo	Métrica	Promedio	STD	IC inf.	IC sup.	CV (%)
yolov8s	Precision	0,758924	0,008778	0,748025	0,769823	1,16%
yolov8s	Recall	0,875650	0,009807	0,863473	0,887827	1,12%
yolov11s	mAP50	0,863452	0,003501	0,859105	0,867799	0,40%
yolov11s	mAP50-95	0,596454	0,001353	0,594774	0,598134	0,23%
yolov11s	Precision	0,748018	0,006563	0,739869	0,756167	0,88%
yolov11s	Recall	0,884342	0,002781	0,880889	0,887795	0,31%
yolov12s	mAP50	0,870772	0,003498	0,866429	0,875115	0,40%
yolov12s	mAP50-95	0,598518	0,001487	0,596672	0,600364	0,25%
yolov12s	Precision	0,757790	0,005762	0,750635	0,764945	0,76%
yolov12s	Recall	0,875732	0,002445	0,872696	0,878768	0,28%
yolov26s	mAP50	0,842972	0,006967	0,834322	0,851622	0,83%
yolov26s	mAP50-95	0,586010	0,002120	0,583378	0,588642	0,36%
yolov26s	Precision	0,722874	0,009202	0,711448	0,734300	1,27%
yolov26s	Recall	0,861346	0,006582	0,853174	0,869518	0,76%

Nota. Tabla de métricas estadísticas del entrenamiento de los modelos YOLO de tamaño small. Elaboración propia.

Anexo F

Lista de los principales hiperparámetros utilizados para el entrenamiento de los modelos RF-DETR.

- **Epocas:** 30
- **Batch:** 16 (8 gradient accumulation y 2 batches)
- **Early stopping:** Sí
- **Patience:** 8
- **Optimizer:** AdamW (Adam with Weight Decay)
- **Learning rate:** 0,001

- **Weight Decay:** 0,001
- **Exponential Moving Average (EMA):** Sí
- **Seed:** aleatoria (pesos no determinísticos)

Anexo G

Lista de los principales hiperparámetros utilizados para el entrenamiento de los modelos

DEIMv2

- **Epocas:** 60 (small), 65 (nano)
- **Batch size:** 4 (small), 6 (nano)
- **Workers:** 4
- **Early stopping:** No (No soportado)
- **Optimizer:** AdamW (Adam with Weight Decay)
- **Learning rate:** 0,000025
- **Weight Decay:** 0,0001
- **Exponential Moving Average (EMA):** Sí
- **Seed:** aleatorio

Anexo H

Lista de parámetros usados para aplicar umento de datos en DEIMv2

- **Mosaic:** 1,0
- **Random Photometric Distort:** 0,5
- **Random Zoom Out:** Sí
- **Random IoU Crop:** 0,8
- **Sanitize Bounding Boxes:** 1
- **Random Horizontal Flip:** Sí

- **Normalize:** media (0,486; 0,456; 0,46), std (0,229; 0,224; 0,225)
- **Convert Boxes:** cxcywh

Anexo I

Lista de características de hardware del dispositivo Raspberri Pi 5

- **Dispositivo:** Raspberry Pi 5
- **CPU:** Broadcom BCM2712, Cortex-A76 64Bit SoC
- **Frecuencia máxima de CPU:** 2.4GHz
- **GPU:** VideoCore VII
- **Frecuencia máxima de GPU:** 800Mhz
- **Memoria RAM:** 8GB LPDDR4X-4267 SDRAM
- **Consumo energético:** 5A@5V
- **Refrigeración:** Enfriamiento activo + disipador
- **Sistema operativo:** Linux
- **Plataforma de ejecución:** Python 3.11.15, torch-2.10.0, Ultralytics 8.4.19
- **Precio aprox.**: $124 USD (unidad)

Anexo J

Tabla de formatos exportados según su arquitectura y precisión

Arquitectura	Formato	FP32	FP16	INT8
YOLO	ONNX	Sí	Sí	No
	OPENVINO	Sí	Sí	Sí
	NCC	Sí	Sí	No
	MNN	Sí	Sí	Sí
	TORCHSCRIPT	Sí	No	No

Arquitectura	Formato	FP32	FP16	INT8
RF-DETR	ONNX	Sí	Sí	No
	OPENVINO	Sí	Sí	Sí
	MNN	Sí	Sí	Sí
DEIMv2	ONNX	Sí	Sí	No
	OPENVINO	Sí	Sí	Sí

Nota. Se muestran los formatos exportados, junto con su arquitectura y su precisión, para su ejecución en la CPU. Elaboración propia.

Anexo K

Tabla de métricas evaluadas para los diferentes modelos YOLO

Modelo	Precisión	mAP50-95	Latencia (ms)	FPS	Tamaño (MB)
yolov11n	FP32	0,5532	66,348 ±0,22	~15,1	10,0
yolov11n	FP16	0,5541	66,608 ±0,35	~15,0	5,1
yolov11n	INT8	0,5513	93,640 ±0,08	~10,7	2.7
yolov11s	FP32	0,5739	163,836 ±0,27	~6,1	36,1
yolov11s	FP16	0,5742	165,238 ±1,50	~6,0	18,2
yolov11s	INT8	0,5739	237,034 ±1,08	~4,2	9,3
yolov12n	FP32	0,5586	142.712 ±2.40	~7,0	10,0
yolov12n	FP16	0,5587	145,490 ±3,40	~6,9	5,1
yolov12n	INT8	0,5573	136,396 ±1,15	~7,3	2.8
yolov12s	FP32	0,5771	345,756 ±1,07	~2.9	35,4
yolov12s	FP16	0,5778	342.270 ±5,60	~2.9	17,9
yolov12s	INT8	0,5752	323,818 ±1,17	~3,1	9,3
yolov8n	FP32	0,5629	67,414 ±0,12	~14,8	11,6
yolov8n	FP16	0,5631	67,546 ±0,21	~14,8	5,9
yolov8n	INT8	0,5600	91,662 ±0,55	~10,9	3,1

Nota. Se muestran las distintas métricas evaluadas de cada modelo. Elaboración propia.

Anexo L

Lista de características de hardware del dispositivo NPU utilizado.

- **Dispositivo:** Hailo 8 HAT+
- **Tipo:** Neural Processing Unit – Raspberry HAT+
- **Fabricante:** HAILO
- **Velocidad:** 26 Tera-Operaciones por Segundo (TOPS)
- **Precisión:** INT8
- **Consumo energético:** 2.4W
- **Frameworks:** TensorFlow, TensorFlow Lite, Keras, PyTorch & ONNX
- **Precio aprox.:** $110 USD