# Caracterización de rendimiento computacional en plataformas embebidas para aplicaciones de Edge AI en modelos de detección de personas

## Performance characterization on embedded systems for Edge AI person-detection models

Laura Cabrera-Quirós[1], Kimberly Orozco-Retana[2]

*Fecha de recepción: 24 de enero, 2025*
*Fecha de aprobación: 8 de mayo, 2025*

Cabrera-Quirós, L; Orozco-Retana, K. Caracterización de rendimiento computacional en plataformas embebidas para aplicaciones de Edge AI en modelos de detección de personas. *Tecnología en Marcha*. Vol. 38, Nº 4. Octubre-Diciembre, 2025. Pág. 191-201.

https://doi.org/10.18845/tm.v38i4.7754



1 Instituto Tecnológico de Costa Rica. Costa Rica.
lcabrera@itcr.ac.cr
2 Instituto Tecnológico de Costa Rica. Costa Rica.
asdkimberlyasd@gmail.com
https://orcid.org/0009-0004-8241-4991

## Palabras clave

Edge AI; NVIDIA Jetson Nano; Raspberry Pi 4; MLPerf Inference Benchmark; SSD-MobileNet.

## Resumen

Este documento presenta una caracterización del rendimiento del hardware para dos plataformas de Edge AI: Raspberry Pi 4 y NVIDIA Jetson Nano, para la tarea de detección automática de personas utilizando un modelo de aprendizaje profundo. Con fines comparativos, utilizamos el sistema de evaluación MLPerf Inference Benchmark. La caracterización considera los resultados de un modelo de detección de objetos SSD-Mobilenet utilizando dos conjuntos de datos diferentes, uno con 80 clases de objetos distintas y otro solo con personas. Las métricas de comparación consideran la precisión del modelo, la latencia, las consultas procesadas por segundo y las muestras procesadas por segundo bajo la evaluación de diferentes escenarios de ejecución.

## Keywords

Edge AI; NVIDIA Jetson Nano; Raspberry Pi 4; MLPerf Inference Benchmark; SSD-MobileNet.

## Abstract

This paper presents a hardware performance characterization for two Edge AI platforms: Raspberry Pi 4 and NVIDIA Jetson Nano, for the task of automatic people detection using a deep learning model. For comparison purposes, we use the MLPerf Inference Benchmark evaluation system. The characterization considers the results from an SSD-Mobilenet object-detection model using two different datasets, one with 80 different object classes and another with only people. Comparison metrics consider model accuracy, latency, queries processed per second, and samples processed per second under the evaluation of different execution scenarios.

## Introducción

Actualmente existe una tendencia de crecimiento vertiginoso en la cantidad de datos disponibles y la necesidad de su procesamiento autónomo y continuo para diferentes tareas, donde también la baja latencia en las aplicaciones finales se ha convertido en una demanda común. La inteligencia artificial (IA) en el borde o Edge AI surge como una solución para procesar los datos mediante modelos de IA cerca de la fuente de la información mediante dispositivos de bajo consumo [1].

Para esto, el uso de sistemas embebidos o empotrados como plataformas de soporte se ha vuelto una práctica normal. Sin embargo, la implementación de IA en sistemas embebidos, en específico algoritmos de alta demanda computacional como las redes neuronales, requiere de una reconciliación entre elementos de hardware y software de la plataforma. Alternativas incluyen el uso de redes neuronales artificiales más pequeñas, en donde los modelos a utilizar estén cuantificados, comprimidos y/o podados. Por otro lado, los sistemas embebidos modernos proveen características adicionales como aceleradores, GPUs y NPUs para soportar dichos modelos.

Por su parte, la detección de objetos consiste en detectar automáticamente instancias de objetos de una o varias clases conocidas en imágenes o video; y es ampliamente utilizada por distintas industrias. Los modelos de detección de personas se enfocan específicamente en esta clase en particular, usando por lo general modelos basados en redes neuronales [2].

Aún cuando el uso de modelos de detección de personas en sistemas embebidos para aplicaciones Edge AI es un tema de gran impacto y aplicabilidad, existen pocas caracterizaciones de este tipo de modelos para su uso eficiente en plataformas embebidas de recurso computacional reducido. En su mayoría, estas comparaciones y caracterizaciones se enfocan en servidores de alta escala [1]. No obstante, conocer qué tipo de modelo o algoritmo mapea mejor a una plataforma embebida en específico le puede permitir a investigadores e industria utilizar de forma óptima sus recursos. En los últimos años se han desarrollado caracterizaciones de rendimiento utilizando diversos estándares como AIoTbench, IoTbench o pCAMP [1] para distintos sistemas embebidos. Sin embargo, muchas de las caracterizaciones realizadas se han enfocado en tareas como traducción, otras tareas de visión como clasificación y reconocimiento de voz. Más no consideran la tarea de detección automática de objetos en la diversidad de su dominio.

Este artículo presenta una caracterización de rendimiento para un modelo de detección ampliamente utilizado en sistemas embebidos, como lo es el MobileNet, en dos plataformas embebidas aptas para uso en Edge AI: la Raspberry Pi 4 Modelo B y la NVIDIA Jetson Nano. Para garantizar una comparación correcta y escalable, se utilizó el estándar MLPerf Inference Benchmark [3].

## MLPerf Inference Benchmark

El estándar MLPerf Inference es un conjunto de programas de evaluación para inferencia de modelos de inteligencia artificial que mide el rendimiento de inferencia del hardware, el software y los servicios de aprendizaje automático con métricas adecuadas y un método de evaluación comparativa de manera justa [3]. MLPerf Inference Benchmark Suite incluye programas de evaluación que cubren tres tareas de inferencia: clasificación de imágenes, detección de objetos, y reconocimiento de voz y traducción. Estas tareas pueden ser evaluadas bajo cuatro distintos escenarios: Flujo único, Flujo múltiple, Servidor y Fuera de línea. Dichos escenarios permiten emular el comportamiento de carga de trabajo de aprendizaje automático que se tiene en el mundo real para dispositivos móviles, vehículos autónomos, y configuraciones basadas en la nube. Los cuatros escenarios de ejecución serían:

- Flujo único: representa un flujo de consulta de inferencia con un tamaño de muestra de 1, lo que refleja aplicaciones cliente donde la capacidad de respuesta es crítica. Su métrica recomendada es la latencia del percentil 90 del flujo de consultas.
- Flujo múltiple: representa aplicaciones con un solo flujo de consultas, pero cada consulta comprende múltiples inferencias, lo que refleja tareas de automatización industrial y detección remota. Su métrica de rendimiento es el número entero de flujos que admite el sistema mientras cumple con el requisito de calidad de servicio (QoS, por sus siglas en inglés).
- Servidor: representa aplicaciones en línea donde la llegada de consultas es aleatoria y la latencia es importante. La métrica de rendimiento es el parámetro de Poisson que indica las consultas por segundo (QPS, por sus siglas en inglés) que se pueden lograr mientras se cumple el requisito de QoS.
- Fuera de línea: representa aplicaciones de procesamiento por lotes donde todos los datos están disponibles de inmediato y la latencia no está restringida. La métrica para el escenario fuera de línea es el rendimiento medido en muestras por segundo.

## Plataformas embebidas a caracterizar

Adicional a estas plataformas, se utiliza una computadora Dell Inc. Inspiron 5567, como la base de comparación en el rendimiento de los sistemas embebidos.

### Raspberry Pi 4 Modelo B

Raspberry Pi es desarrollada por *Raspberry Pi Foundation* [4]. Las Raspberry Pi tienen suficiente capacidad para realizar automatización doméstica, gracias a su microprocesador Quad core Cortex-A72 (ARM v8) de 64-bit SoC @ 1.8GHz. Esto permite implementar aplicaciones industriales e implementar computación en el borde para una amplia variedad de aplicaciones, a bajo costo.

### NVIDIA Jetson Nano

Como parte de la primera serie de sistemas NVIDIA Jetson Nano, el kit de desarrollador NVIDIA Jetson Nano 2GB revolucionó la informática integrada al brindar el poder de la inteligencia artificial a los dispositivos computacionales de última generación. La tarjeta proporciona una interfaz GPIO para conectarse a módulos de dispositivos externos. Cuenta con CPU ARM A57 y tiene una GPU Maxwell de 128 núcleos [5].

## Modelo de detección y conjunto de datos

### Modelo SSD-MobileNet

El enfoque del detector de disparo único o *Single Shot Detector* (SSD) se basa en una red convolucional de avance que produce una colección de cuadros delimitadores de tamaño fijo con puntuaciones para la presencia de instancias de clase en esos cuadros, seguido de un paso de supresión no máxima para producir las detecciones finales. Las primeras capas de la red se basan en una arquitectura estándar utilizada para la clasificación de imágenes de alta calidad, pero truncada antes de las capas de clasificación, llamada red base [6]. En el caso de este trabajo la red base que utiliza es MobileNet, de modo que la arquitectura final se distingue con el nombre SSD-MobileNet. Los modelos MobileNet son modelos de clasificación eficientes para aplicaciones de visión móviles e integradas. Se enfocan principalmente en optimizar la latencia, generando redes pequeñas construidas principalmente a partir de convoluciones separables en profundidad. Este tipo de convolución es una forma de convolución factorizada, que como su nombre lo indica factoriza una convolución estándar en una convolución en profundidad y una convolución 1x1 denominada convolución puntual [7].

### MS-COCO

Microsoft COCO (*Common Objects in Context*) es un conjunto de datos de detección de objetos a gran escala, de acceso libre y altamente utilizado mundialmente. El conjunto de datos comprende imágenes de escenas cotidianas complejas que contienen objetos comunes en su contexto natural. Los objetos se etiquetan utilizando segmentaciones por instancia para ayudar a localizarlos de forma precisa [8]. Además, presenta separaciones consistentes entre conjuntos de entrenamiento, validación y prueba; para fomentar la reproducibilidad y comparación de resultados.

En 2017 la división de entrenamiento/validación se cambió. El conjunto de prueba utilizado (COCOval2017) es un subconjunto de 41000 imágenes del conjunto de prueba de 2015 [9]. La nueva división usa las mismas imágenes y anotaciones. Este conjunto de validación contiene 5000 imágenes con 80 tipos de objetos en la nueva división.

Además, COCO tiene varios tipos de anotaciones: para detección de objetos, detección de puntos clave, segmentación de cosas, segmentación panóptica, densa y subtítulos de imágenes. Para este trabajo se utilizaron solamente las anotaciones de objetos usando cuadros delimitadores.

## Metodología y esquema de pruebas

### Implementación de MLPerf Inference Benchmark en plataformas embebidas

Para implementar el estándar de pruebas es necesario contar con un sistema operativo funcional en cada plataforma embebida a evaluar. Para esto en cada sistema se deben contar con los archivos necesarios para la instalación de MLPerf Inference Benchmark en almacenamiento. Estos se pueden descargar desde el repositorio dedicado oficial.

Posteriormente, se realiza la instalación del generador de carga o *LoadGen*, que es un generador de tráfico para MLPerf Inference que carga el sistema bajo prueba (SUT, por sus siglas en inglés) y mide su rendimiento. Además, recopila información de registro, depuración y post-procesamiento de datos [3]. El generador de carga registra las consultas y las respuestas del SUT y al final de la ejecución reporta las estadísticas, resúmenes de resultados y determina si la ejecución fue válida.

Una vez instalado el generador de carga, es posible cargar datos y modelos bajo escenarios específicos de ejecución mediante comandos en la terminal del sistema haciendo uso de archivos Unix shell.

### Preparación de datos

El modelo SSD-MobileNet utiliza el tamaño de imagen de 300x300, por lo que el conjunto de datos para las pruebas (incluyendo imágenes y anotaciones) debe escalarse a un tamaño de imagen 300x300. Posteriormente, se extrae un subconjunto de datos de 100 imágenes y se conserva el archivo dimensionado de anotaciones 300x300 de más de 80 clases de objetos etiquetados en el COCOval2017. A este subconjunto se le llamará *all_objects_300*. Además, se extrae un segundo subconjunto de datos de 100 imágenes que contienen solamente instancias de la clase ‘persona’, a este se le llamará *person_objects_300*. En ese caso se extraen de las anotaciones 300x300 solamente la información de imágenes que tengan instancias de personas.

Es importante enfatizar que el modelo utilizado está pre-entrenado utilizando el conjunto de entrenamiento de MSCOCO. Los subconjuntos propuestos serán utilizados solamente para pruebas.

### Escenarios de ejecución

Se propone ejecutar 4 pruebas con cada subconjunto de datos extraído de COCOval2017 en cada plataforma embebida, contemplando los cuatro escenarios de ejecución de MLPerf Inference Benchmark. Cada escenario representa el entorno de funcionamiento de distintas tareas de inferencia. Aunque algunos de ellos no son específicamente representativos para la detección de objetos, se busca brindar una caracterización que considere todos los escenarios y brinde a quien diseñe los subsecuentes sistemas un panorama más completo de las capacidades de las plataformas en diversos escenarios.

La Figura 1 muestra un esquema que integra los dos conjuntos de datos en el sistema de pruebas MLPerf Inference Benchmark y la metodología de prueba propuesta.

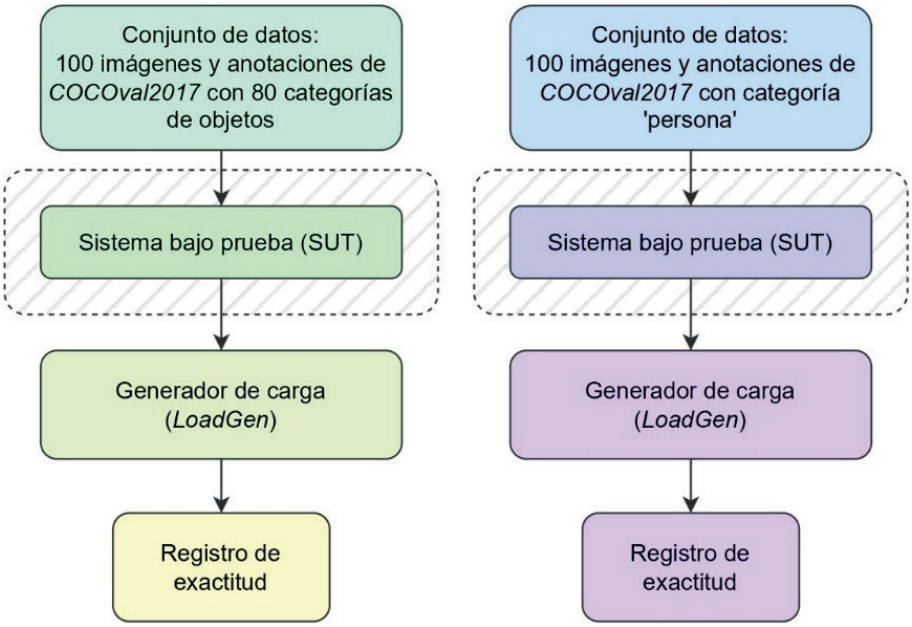


**Figura 1.** Esquema de integración en MLPerf Inference Benchmark para ejecución de pruebas.

## Resultados y discusión

Para las pruebas propuestas y cada una de las plataformas a caracterizar, se calculan las siguientes métricas de rendimiento: exactitud del modelo de detección, tiempos de inferencia, latencia del percentil 90, consultas por segundo y muestras por segundo. Los resultados para las mismas se detallan a continuación.

### Pruebas de exactitud

Las pruebas de exactitud permiten medir cuán efectivo es un modelo en la tarea de detectar objetos, en este caso personas. Para este efecto se utilizan las métricas de precisión promedio media (*mean average precision* o mAP) y exactitud (*accuracy*), las cuales son ampliamente utilizadas en investigación sobre detección de objetos [10]. Estas se calculan realizando las pruebas en cada plataforma y utilizando dos subconjuntos de datos mencionados.

El cuadro 1 desglosa los resultados de mAP y exactitud para el modelo SSD-Mobilenet con dos conjuntos de datos distintos según las plataformas bajo prueba.

**Cuadro 1.** Métricas de mAP y exactitud para modelo SSD-Mobilenet según plataformas bajo prueba y conjuntos de datos considerados.

| | Conjunto de datos: all_objetcts_300 | | | Conjunto de datos: person_objects_300 | | |
|---|---|---|---|---|---|---|
| Plataforma | Modelo | mAP (%) | Exactitud (%) | Modelo | mAP (%) | Exactitud (%) |
| Dell Inspiron | SSD-Mobilenet | 33.55 | 94.06 | SSD-Mobilenet | 27.48 | 62.10 |
| Raspberry Pi 4 | SSD-Mobilenet | 33.55 | 94.06 | SSD-Mobilenet | 27.48 | 62.10 |
| Jetson Nano CPU | SSD-Mobilenet | 33.55 | 94.06 | SSD-Mobilenet | 27.48 | 62.10 |
| Jetson Nano GPU | SSD-Mobilenet | 33.55 | 94.06 | SSD-Mobilenet | 27.48 | 62.10 |

Los resultados muestran que se mantiene la exactitud y mAP en todas las plataformas, corroborando que el desempeño de SSD-Mobilenet se mantiene consistente independientemente de la plataforma para un mismo conjunto de datos. De acuerdo con [11], los modelos MobileNet tienen un rango de mAP dentro del 18% al 25% cuando se realiza inferencia utilizando el conjunto de datos COCOval2017, lo que se logra replicar.

### Tiempo de inferencia

El tiempo de inferencia representa la cantidad de tiempo promedio que tarda un modelo en detectar un objeto para una entrada dada. Valores más bajos de tiempo de inferencia indican un mayor rendimiento de la plataforma.

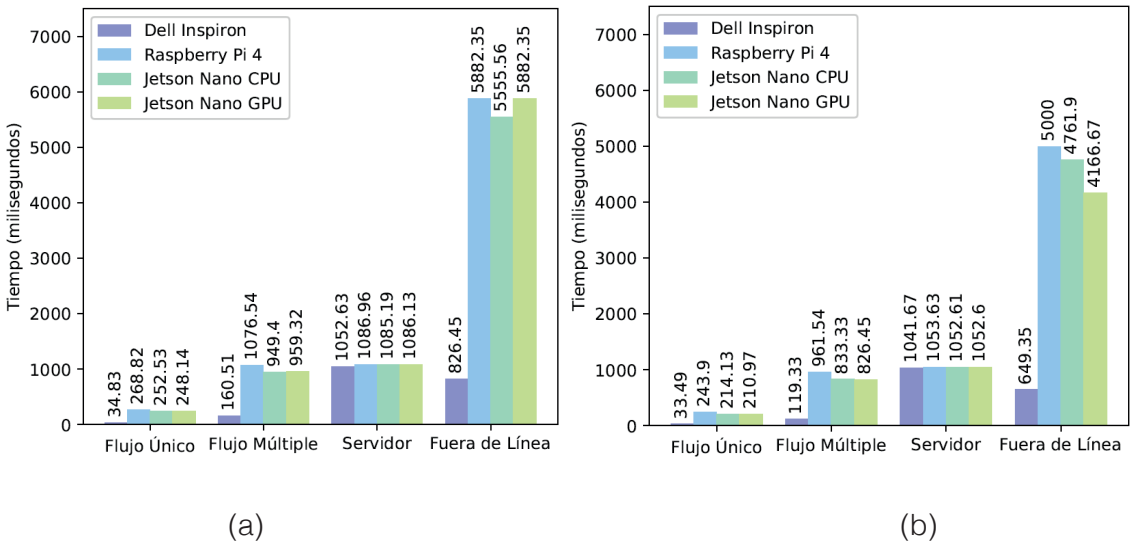


**Figura 2.** Tiempo de inferencia para todos los escenarios utilizando (a) Subconjunto *all_objects* y (b) Subconjunto *person_objects*.

La Figura 2 muestra los tiempos de inferencia para los cuatro escenarios disponibles considerando dos conjuntos de datos. La diferencia entre los tiempos de ejecución de acuerdo con cada plataforma muestra a las dos plataformas embebidas con tiempos de inferencia similares considerando ambos conjuntos de datos.

Considerando los escenarios de ejecución se observan los tiempos de inferencia más bajos (rondando los 200 - 300 milisegundos) para el escenario de Flujo Único. Por su parte, el escenario Fuera de Línea presenta tiempos de inferencia más altos de estas pruebas. En este escenario el tiempo de inferencia significativamente mayor se asocia el tamaño del lote a procesar, ya que en este escenario se procesan lotes de imágenes más grandes en comparación a escenarios de flujo múltiple o único. Un tamaño mayor de lote implica procesar varias imágenes en paralelo, lo que aumenta el tiempo de inferencia total.

### Latencia del percentil 90

Una métrica de rendimiento específica para el escenario de Flujo Único es la latencia del percentil 90 del flujo de consultas. Este escenario representa un solo flujo de entrada crítico, como podría ser el flujo proveniente de una cámara de video. La medición específica de la latencia en este percentil está asociada con el procesamiento del 90% de las consultas en un tiempo específico.

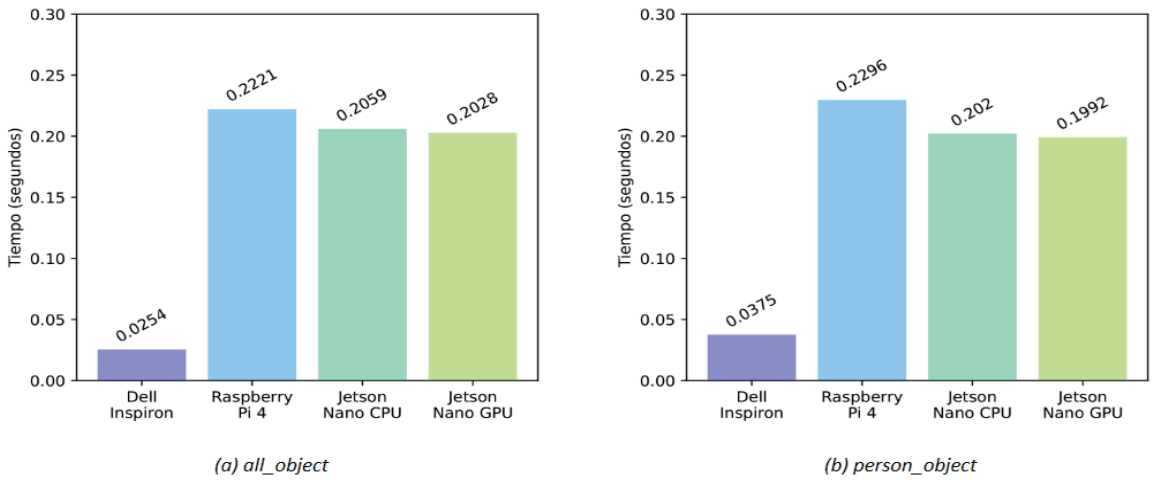


**Figura 3.** Latencia del percentil 90 para escenario Flujo Único utilizando (a) Subconjunto *all_objects* y (b) Subconjunto *person_objects*.

Los resultados de esta latencia al ejecutar las pruebas en cada plataforma se pueden visualizar en la Figura 3. Considerando estos resultados es posible notar una latencia levemente mayor para Raspberry Pi 4 respecto a los dos modos de ejecución de Jetson Nano. Además, en contraste, el uso de diferentes conjuntos de datos no muestra cambios en los resultados de latencia para SSD-Mobilenet.

En general, los resultados de latencia en Flujo Único para SSD-Mobilenet pueden considerarse consistentes en la detección para las dos plataformas embebidas, tanto para los objetos de 80 clases como para la categoría específica de 'persona'. Resultados como baja latencia hace a SSD-Mobilenet apto para implementaciones críticas donde se tenga un solo flujo de consultas, aún en plataformas de recursos limitado como la Raspberry Pi 4.

## Consultas por segundo

En casos donde se representen aplicaciones con un solo flujo de consultas, pero con múltiples inferencias como la detección remota con múltiples cámaras de video (Flujo Múltiple), o bien casos donde la llegada de consultas es aleatoria (Servidor), es importante conocer la métrica de consultas por segundo o QPS procesadas por el sistema.

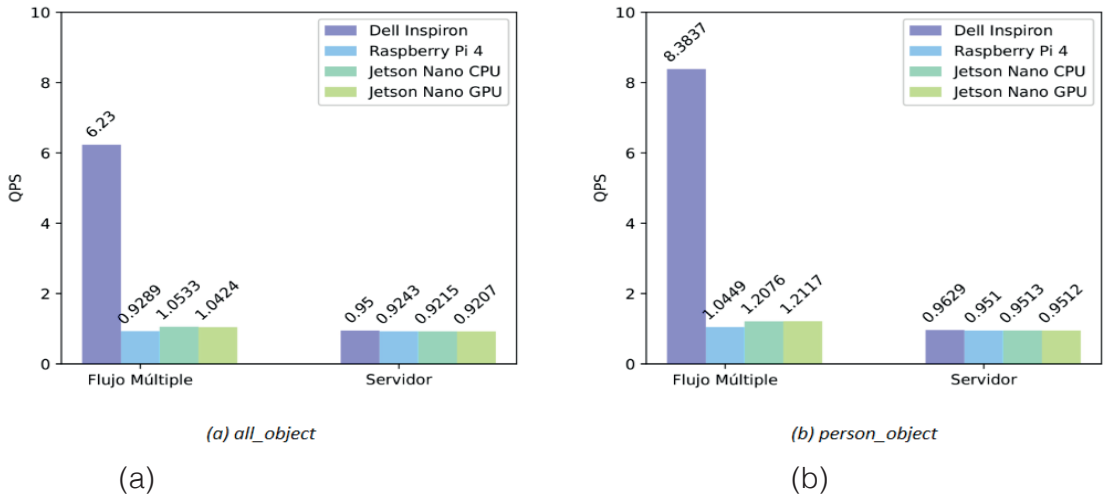


**Figura 4.** Consultas por segundo escenario Flujo Múltiple y servidor utilizando (a) Subconjunto *all_objects* y (b) Subconjunto *person_objects*.

Al realizar este conjunto de pruebas es importante conocer los parámetros fijos de ejecución en cada escenario, dado que en caso Servidor, la latencia restringe la cantidad de consultas por segundo que puedan ingresar, así como en el caso de Flujo Múltiple donde la diferencia entre las muestras exige límites de latencia para procesar una consulta entrante. En este caso se trabaja con una latencia fija de 50ms previamente establecido por el sistema de pruebas de inferencia. La Figura 4 muestra los resultados de la aplicación de pruebas en las plataformas de acuerdo con los dos escenarios relevantes.

De manera general, se nota un desempeño similar en las consultas por segundo en ambos escenarios para las plataformas embebidas. Estos resultados se complementan con los obtenidos en la sección de tiempos de inferencia, donde se puede apreciar la congruencia en los tiempos de inferencia y QPS para el escenario Servidor. El tiempo de inferencia y el QPS están inversamente relacionados. Un tiempo de inferencia más bajo implica mayor velocidad de procesamiento de las solicitudes, lo que a su vez conduce a un QPS más alto. Por otro lado, un tiempo de inferencia más alto disminuye la velocidad de procesamiento de las solicitudes, lo que resulta en un QPS más bajo. En este caso, los resultados de tiempo de inferencia son similares para ambos escenarios en ejecución, lo que se refleja en los resultados de la Figura 4. Las diferencias entre plataformas embebidas son mínimas tomando en consideración ambos escenarios. Además, las diferencias entre conjuntos de datos no marcan una diferencia significativa para las plataformas embebidas.

## Muestras por segundo

Finalmente, el escenario Fuera de Línea no representa ambientes de inferencia críticos sino aplicaciones de procesamiento por lotes donde todos los datos están disponibles de inmediato. Así, la latencia no está restringida y el tiempo de inferencia no es crítico, por lo que la métrica de rendimiento se reduce a las muestras procesadas por segundo. La Figura 5 muestra los resultados de las muestras por segundo procesadas por segundo bajo la ejecución del escenario Fuera de Línea.

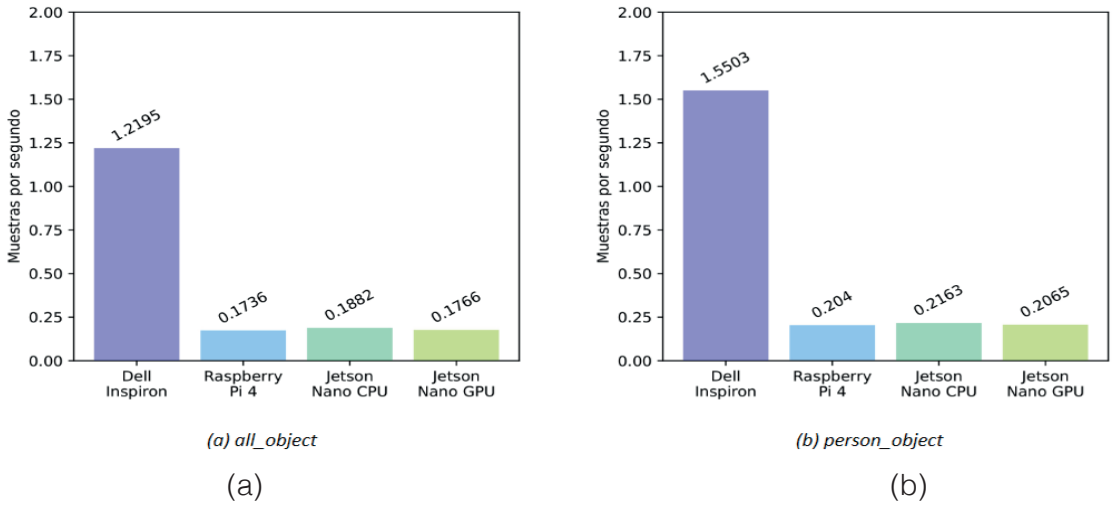


**Figura 5.** Muestras procesadas por segundo escenario Fuera de Línea utilizando (a) Subconjunto *all_objects* y (b) Subconjunto *person_objects*

En términos de las plataformas embebidas se aprecia un comportamiento similar en el número de muestras procesadas por segundo. Los resultados de la Figura 5 se complementan con los expuestos en la sección de tiempo de inferencia en cuanto al tiempo que le toma al modelo

SSD-Mobilenet realizar una inferencia bajo el escenario Fuera de Línea. En el caso de las plataformas embebidas, la tasa de procesamiento de muestras se encuentra muy por debajo de la unidad por segundo.

## Análisis de rendimiento según escenarios

Considerando las plataformas evaluadas solo en términos de caracterización, es posible considerar un mayor rendimiento de la Jetson Nano en sus dos modos de ejecución sobre el rendimiento de Raspberry Pi 4 de manera general. Esto es claro a partir de un rendimiento para la detección (mAP o exactitud) equivalente entre plataformas, pero con latencias más bajas para la Jetson Nano en ambas configuraciones. No obstante, esta diferencia es mínima. Resalta además la similitud de los resultados para la evaluación de la Jetson Nano trabajando con GPU en comparación a su funcionamiento únicamente como CPU. Intuitivamente, se podría esperar que al utilizar una GPU cuya naturaleza en paralelización le da mayor poder computacional se obtengan tiempos de ejecución menores. Sin embargo, aunque hay una reducción en tiempos y latencias esta es mínima. Este comportamiento puede ser explicado pues para todas las evaluaciones solo se realizaron pruebas a los modelos, más no así su entrenamiento. Así, es conocido que la fase de entrenamiento de un modelo de aprendizaje automático es aquella que requiere mayor poder computacional [12] y, por ende, se beneficia de la presencia de una GPU. Este no es necesariamente el caso para la fase de prueba, donde se realiza solamente inferencia.

Ahora bien, si se considera el costo de cada plataforma según el precio por unidad, NVIDIA Jetson Nano 2GB tiene un precio de mercado de 59$, mientras que Raspberry Pi 4 2GB tiene un precio de 49.5$. Sin embargo, actualmente NVIDIA no produce esta unidad [13] optando por un modelo más costoso.

Tomando en cuenta lo anterior, el desempeño de Raspberry Pi 4 la hace una buena contendiente de la Jetson Nano para la misma aplicación. Las pequeñas diferencias en las métricas de rendimiento permiten visualizar potenciales implementaciones utilizando modelos más livianos que realicen la misma tarea que SSD-Mobilenet. Además, considerando el costo por unidad y la vigencia de la producción, es posible considerar a Raspberry Pi 4 como una plataforma apta para realizar implementaciones.

## Conclusiones y trabajo futuro

Las plataformas Edge AI evaluadas demuestran resultados competentes para la tarea de detección de personas dentro del rango de mAP usual para modelos con redes base MobileNet 18% - 25% entrenado para la detección de 80 clases distintas.

Si bien se logran resultados muy prometedores para la detección automática de personas haciendo uso de SSD-MobileNet, este solo brinda un panorama reducido de las verdaderas capacidades de las plataformas evaluadas. Considerar más arquitecturas de modelos ampliaría este panorama.

El desempeño de las métricas obtenidas refleja la detección de objetos como una tarea de múltiples escenarios, dependiendo de la aplicación final. Es posible llevar a la implementación sistemas que comprendan uno o varios flujos de consultas para escenarios críticos, o bien una implementación que considere los beneficios de la nube para realizar inferencias cuando las plataformas de borde así lo requieran. Sin embargo, considerando la capacidad limitada y la afinidad de las plataformas Edge AI, la inferencia fuera de línea no representa un uso eficiente de estas.

Como trabajo futuro, se recomienda tomar en cuenta una variedad más amplia de modelos de detección, así como modelos afinados para la detección exclusiva de personas, con el fin de acotar la verdadera capacidad de las plataformas evaluadas en este trabajo para una tarea específica. Para lograr esto es posible considerar alternativas como MLHarness [14] que aborda las limitaciones de modelos entrenados por MLPerf Inference Benchmark mediante un sistema escalable de evaluación comparativa que se adapta a éste. Con esta herramienta sería posible utilizar modelos pre-entrenados de diferentes fuentes para ampliar el rango de caracterización de las plataformas evaluadas.

## Referencias

[1] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi y C. Stewart, «A Survey on Edge Performance Benchmarking,» *ACM Computing Surveys,* vol. 54, nº 3, pp. 1-33, Abril 2022.

[2] Z. Zhou, K. Chen, Z. Shi, Y. Guo y J. Ye, «Object Detection in 20 Years: A Survey,» *Proceedings of the IEEE,* vol. 111, nº 3, pp. 257-276, Marzo 2023.

[3] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou et al., «MLPerf Inference Benchmark,» *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA),* pp. 446-459, 2020.

[4] E. Upton y G. Halfacree, Raspberry Pi user guide, John Wiley & Sons, 2016.

[5] F. N. Uzun, M. Kayrici y B. Akkuzu, «Nvidia Jetson Nano Development Kit,» *Programmable Smart Microcontroller Cards,* p. 82, 2021.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu y A. C. Berg, «SSD: Single Shot MultiBox Detector,» de *European conference on computer vision*, 2016.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto y H. Adam, «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,» Publisher: arXiv Version Number: 1, 2017. [En línea]. Available: https://arxiv.org/abs/1704.04861.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár y C. L. Zitnick, «Microsoft COCO: Common Objects in Context,» *Computer Vision – ECCV 2014,* vol. 8693, pp. 740-755, 2014.

[9] «COCO dataset - Common Objects in Context,» [En línea]. Available: https://cocodataset.org/#home. [Último acceso: Julio 2025].

[10] R. Padilla, S. L. Netto y E. Da Silva, «A survey on performance metrics for object-detection algorithms,» de *International conference on systems, signals and image processing (IWSSIP*, 2020.

[11] W. Wang, W. Hong, F. Wang y J. Yu, «GAN-Knowledge Distillation for One-Stage Object Detection,» *IEEE Access,* vol. 8, pp. 60719--60727, 2020.

[12] Y. Hu, N. Chen, Y. Hou, X. Lin, B. Jing y P. Liu, «Lightweight deep learning for real-time road distress detection on mobile devices,» *Nature Communications,* vol. 16, nº 1, 2025.

[13] NVIDIA Developer, «Documentation Jetson Nano 2GB Developer Kit,» [En línea]. Available: https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide. [Último acceso: Julio 2025].

[14] Y.-H. Chang, J. Pu, W.-m. Hwu y J. Xiong, «MLHarness: A scalable benchmarking system for MLCommons,» *BenchCouncil Transactions on Benchmarks, Standards and Evaluations,* vol. 1, 2021.

## Declaración sobre uso de Inteligencia Artificial (IA)

Utilizamos la herramienta de inteligencia artificial disponible en https://www.deepl.com/en/translator para traducir partes de este artículo del inglés al español. La herramienta nos ayudó a agilizar el proceso de traducción, pero realizamos una revisión exhaustiva para asegurar la calidad y precisión de las traducciones.