INTEGRATION, the VLSI journal 95 (2024) 102127

Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

Quantitative comparison and performance evaluation of deep learning-based object detection models on edge computing devices

Darío G. Lema *, Rubén Usamentiaga, Daniel F. García

Department of Computer Science and Engineering, University of Oviedo, Campus de Viesques 33204 Gijón, Asturias, Spain

ARTICLE INFO

Keywords:
Edge computing devices
The VLSI journal
Deep learning
Object detection

ABSTRACT

Low latency in the detection of objects in images is a fundamental aspect to provide an immediate response in different application scenarios, and to carry out the relevant actions. Deep learning-based models offer better results than traditional object detection techniques. As a result of this increased accuracy, the computational cost of these models is often very high. For this reason, several edge computing devices have emerged that perform inference quickly and efficiently due to the incorporation of hardware accelerators. In this work, different devices are evaluated using object detection algorithms based on deep learning. For this purpose, YOLOv3, YOLOv5 and YOLOX, with all their variants, have been used on an NVIDIA Jetson Nano, an NVIDIA Jetson AGX Xavier and a Google Coral Dev Board. For the evaluation of the models and devices, one of the most widespread datasets, MS COCO, is used. In addition, they have been evaluated using twenty different input sizes and three frameworks (Pytorch, TensorRT and Tensorflow Lite). From the data obtained, data can be extrapolated to other models such as YOLOv8. Additionally, the FPS/Power Consumption and FPS/Cost ratios are analyzed, as well as their feasibility in a real use scenario. As a result of this work, valuable recommendations are provided for projects where this technology is to be applied.

1. Introduction

In the last decade the field of deep learning and CNNs (Convolutional Neuronal Networks) has undergone a revolution [1]. Although the concept of CNNs dates back to the last century [2], it was not until the popularization of GPUs (Graphics Processing Unit) that their full potential has been realized [3]. This is, primarily, because CPUs (Central Processing Unit) do not provide the computational capacity necessary to train with large amounts of data in reasonable periods of time. Thanks to the widespread use of GPUs, major advances have been made in recent years in sectors as diverse as autonomous driving [4–6], defect detection in industrial parts [7,8] or healthcare [9,10].

One of the fields where most progress has been made is in object detection. The increase in the accuracy of deep learning-based object detection models is due, in part, to the creation of deep networks. Increasing the number of layers in a CNN to improve the accuracy of models is not always positive, since at a certain point there is stagnation. This is because by using so many layers, the loss function decreases very slowly. One of the significant advancements aimed at addressing the issue known as the Vanishing Gradient problem [11] is the utilization of residual blocks [12]. In a traditional neural network, each layer feeds the next. In one with residual blocks, it feeds the next one and the layers that are a certain number of steps away. This way, the gradient descent is faster.

These advances have increased the accuracy of object detection models considerably, but a high computational cost. Due to this high computational cost, training is lengthy, and the speed of inference can limit real-time applications.

Currently, there are two alternatives for the deployment of these models in real environments: cloud computing and edge computing. Cloud computing is a type of computing carried out by several computers connected by IP networks, where data is stored and processed [13]. Applied to computer vision, this definition means that the images taken must be transmitted through an IP network to a computing center, processed with a certain model, and the result sent back to the device responsible for performing the appropriate actions.

The alternative to cloud computing is edge computing. Edge computing is the technique of processing data as close as possible to the source [14]. This type of computing has a number of advantages [15]:

1. Latency reduction. Latency is the delay between sending information and receiving a response. Low latency is the key to implementing real-time services. In the case of object recognition in images, latency is composed of the image transmission

* Corresponding author.
E-mail address: gonzalezdario@uniovi.es (D.G. Lema).

https://doi.org/10.1016/j.vlsi.2023.102127
Received 24 May 2023; Received in revised form 14 November 2023; Accepted 18 December 2023
Available online 27 December 2023
0167-9260/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

time, the inference time, and the time it takes to send the information back. With cloud computing, it is necessary to send the image or video through an IP network, therefore the latency is high. However, with edge computing, it is not necessary to send the information outside the local network (LAN), so the latency is significantly lower. By processing data on edge computing devices, network transfer requirements are reduced. In addition, larger network input sizes can be used. In cloud computing, it is necessary to transmit the image to be processed. If the image is too large, the latency will increase in excess. Using larger input sizes results in better accuracy.

2. No need for connectivity. When the devices which perform the expected actions are in remote locations, lack of connectivity plays a key role. If these devices do not have connectivity to the network, a solution based on edge computing is not be affected, whereas a solution based on cloud computing would not be functional.
3. Privacy and security. With edge computing it is not necessary to send data outside the LAN, which means an increase in privacy. If, for example, face images are collected, sending them over the network may imply a violation of privacy. In addition, sensitive data is more exposed to possible cyberattacks.

GPUs can perform object detection using deep learning models very efficiently. Unfortunately, they have two drawbacks: high power consumption and high cost. For mobility reasons, in many scenarios it is necessary to have a battery to power the device that will carry out the object detection, hence low power consumption is key. Also, due to their cost, acquiring many GPUs can be extremely expensive. For this reason, embedded devices, known as edge computing devices, are increasingly accepted to provide image processing solutions at a low cost, while maintaining performance features and low power consumption.

Several papers have evaluated some of these devices. In [16], Raspberry PI, NVIDIA Jetson Nano and NVIDIA Jetson TX2 are evaluated with a custom classification model. The conclusion is that NVIDIA Jetson TX2 performs inference faster than NVIDIA Jetson Nano and Raspberry PI. In [17], the objective is to detect vine trunks in a vineyard field. Two edge computing devices are evaluated: Google Coral USB Accelerator and NVIDIA Jetson Nano. After several experiments, it is concluded that Google Coral USB Accelerator is faster than NVIDIA Jetson Nano. However, the comparison is complex, since Google Coral USB Accelerator speeds up the inference of a certain CPU. This means that the inference time will depend on the CPU used. For this reason, it cannot be directly compared with other edge computing devices. In [18], the following edge computing devices were evaluated to detect construction vehicles: NVIDIA Jetson TX2, NVIDIA Jetson Nano and Raspberry PI 3B+ with Intel NCS (Neural Compute Stick). After testing their inference performance using a SSD-Mobilenet model [19,20], the conclusion is that it is feasible to process images in real-time with the NVIDIA Jetson TX2, but not with NVIDIA Jetson Nano and Raspberry PI 3B+ with Intel NCS (Neural Compute Stick).

Although all these papers analyze different edge computing devices, there are still aspects that have not been compared and evaluated. For example, there are few works that evaluate the NVIDIA Jetson AGX Xavier or the Google Coral Dev Board thoroughly. The models used for the evaluation of these devices are sometimes not state-of-the-art, as they are either old models that do not offer the best performance in terms of accuracy, or they are customized models. Another issue that has not been addressed is the input image size. It is important to analyze the input size of the network, as this directly affects the inference speed and the accuracy of predictions. Another factor is the dataset used to evaluate an object detector. Most of papers use their own dataset, therefore analyzing the results can be complex. The results, both in terms of inference speed and accuracy, vary depending on the framework used to create the network. This issue has not been fully addressed. Finally, other works are limited to showing the consumption indicated by the manufacturers. In this paper, the consumption of the different devices while performing the inference is measured and evaluated.

This work aims to make a more comprehensive evaluation of deep learning-based object detection models taking all relevant criteria into account. This includes the acquisition of devices, and the section on models and frameworks.

2. Materials and methods

Throughout this work, seven models (YOLOv5, YOLOv3, YOLOv3-Tiny, YOLOX, SSD-Mobilenet v2 and EfficientDet-Lite 1) will be analyzed on five edge computing devices (NVIDIA RTX 2080 Ti, NVIDIA Jetson Nano, NVIDIA Jetson Xavier AGX, i7-9700K and Google Coral Dev Board), using three frameworks (Pytorch, TensorRT and TFLite). The models are evaluated by varying the input sizes (from 320 pixels to 1216). Average Precision (AP) is used as it has been proven to be the most robust metric for comparing object detection models [21]. All models have been evaluated on the most widely used dataset in the field of object detection: MS COCO. Finally, the real power consumption of each of these devices is analyzed using a wattmeter.

2.1. Edge computing devices

This section discusses the technical specifications of the edge computing devices used. Table 1 summarizes their main characteristics.

2.1.1. NVIDIA Jetson Nano Developer Kit

NVIDIA Jetson Nano Developer Kit is a small and powerful computer designed to develop AI (Artificial Intelligence) solutions [25]. It is composed of a Quad-Core ARM Cortex-A57 CPU and a NVIDIA Maxwell 128-Cores GPU. More specifications details are shown in Table 1. One of its advantages is that as a small Linux-based computer, it can run almost any deep learning model. This device is available in a 2 GB and a 4 GB version, with prices varying from €52.37 to €93.32.

2.1.2. NVIDIA Jetson AGX Xavier Developer Kit

NVIDIA Jetson AGX Xavier Developer Kit is a robust embedded device that offers 32 TOPs (Tera Operations per Second). It is equipped with an Eight-Core ARM v8.2CPU, NVIDIA Volta 512-Cores GPU and two NVDLA Engines. As the Jetson Nano, it is a Linux-based computer. More details are shown in Table 1. Thanks to its hardware, it achieves very high inference speeds, albeit at a high cost.

2.1.3. Google Coral Dev Board

Google Coral Dev Board is a board computer to perform fast Machine Learning Models. It is equipped with a Quad-Core ARM Cortex-A53 CPU, but its most outstanding feature is the TPU (Tensor Processor Unit) [26]. A TPU is an ASIC (AI accelerator application-specific integrated circuit), developed by Google, optimized to perform neuronal networks with TensorFlow. It is designed to drive AI in embedded devices. For this reason, it is referred to as an edge TPU.

2.2. Analysis of object detection algorithms

In the last decade, several object detectors have been developed that outperform the results of traditional object detection methods. These algorithms can be divided in two types: one-stage and two-stage detectors. Two-stage detectors first propose a set of ROIs (Regions of Interest) and then classify each of the ROIs. R-CNN [27], Fast-RCNN [28] and Faster-RCNN [29] are some examples of this kind of detectors. One-stage detectors perform the detection by treating the entire image. YOLO (You Only Look Once) is an example of a one-stage detector. Typically, one-stage detectors are faster than two-stage, but less accurate. In this section, the object detection algorithms used on edge computing devices are analyzed.

Table 1
Technical specifications of NVIDIA Jetson Nano, NVIDIA Jetson AGX Xavier and Google Coral Dev Board.

	Nvidia Jetson Nano [22]	Nvidia Jetson AGX Xavier [23]	Google Coral Dev Board [24]
CPU	Quad-Core ARM Cortex-A57	Eight-Core ARM v8.2	Quad-Core ARM Cortex-A53
GPU	NVIDIA Maxwell 128-Cores	NVIDIA Volta 512-Cores	Integrated GC7000 Lite Graphics
TPU	–	–	Google Edge TPU Coprocessor
Memory	2 GB–4 GB LPDDR4	32 GB LPDDR4	1 GB LPDDR4
Storage	16 GB eMMC, MicroSD slot	32 GB eMMC	8 GB eMMC, MicroSD slot
Networking	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet
USB	4× USB 3.0, USB 2.0 Micro-B	2× USB-C 3.1	Type-C, Type-A 3.0, Micro-B serial
Power consumption	5–10 W	10–30 W	2 TOPS per watt
OS	Linux (Ubuntu 18.04)	Linux (Ubuntu 18.04)	Linux (Mendel)
Size (mm)	100 × 80 × 29	105 × 105 × 65	88.10 × 59.90 × 22.38
Price	€52.37–€93.32	€664.78	€157.28

2.2.1. You Only Look Once

YOLO is one of the most famous object detection algorithms due to is good accuracy-inference speed trade off. Several versions have been developed:

YOLOv1. The first version of the YOLO family was a major breakthrough since at the time, it was the fastest detector available and offering good Average Precision (AP) [30]. YOLO divides the image in an $S \times S$ grid. Each of the cells of the grid is responsible for the detection of the objects that fall in its center. After this, NMS (Non-Maximal Suppression) is applied to eliminate redundant predictions.

YOLOv2. This is the natural extension of YOLO [31], solving some of the drawbacks of YOLOv1. It introduces anchor boxes to the YOLO family. In YOLOv1, the predictions were made from scratch. Now, thanks to the anchor boxes, there are some initial guesses. The particularity of YOLOv2 is how it calculates these anchor boxes. In previous works, they were selected by hand. In YOLOv2, K-Means is used. Thanks to this innovation, better initial guesses and therefore, better predictions are made.

YOLOv3. YOLOv2's weakness is in the detection of small objects. YOLOv3 [32] solves this problem by performing predictions on three scales. The tradeoff for improving small-object detection is a lower inference speed. For this reason, a smaller version called YOLOv3-Tiny was created. This version is faster than YOLOv3, but its AP (Average Precision) is lower.

YOLOv4 and YOLOv5. YOLOv4 [33] and YOLOv5 [34] are the natural sequels of the YOLO family, created by different authors from the initial versions. As modern object detectors, both are composed of a backbone, a neck, and a head. CSPDarknet-53 is the chosen backbone as feature extractor. PA-Net, as neck, helps to transmit low-level features to the head. Finally, a YOLOv3-based head makes predictions on three scales. Both include additional improvements, such as mosaic augmentation. This technique consists of creating new images from parts of others. This way, the model can be trained with diverse data and therefore, better predictions will be made. YOLOv4 and YOLOv5 have several variants in which the depth and width is modified.

YOLOX. YOLOX [35] is an anchor-free version of YOLOv3. The authors decided not to use anchor boxes for two main reasons: they require a clustering algorithm to determine the optimal ones, and they increase the number of predictions per image. Another contribution is introducing OTA (Optimal Transport Assignment) [36]. With this advanced label assignment, the AP improves significantly. Like YOLOv4 or YOLOv5, it has several variants in which the depth and width of the model are modified.

YOLOv8. Following the introduction of YOLOv5, subsequent iterations have been released to enhance and build upon previous achievements. Among these versions, YOLOv8 [37] emerges as a noteworthy advancement, incorporating transformer-based operations. While retaining the fundamental architecture of YOLOv5, YOLOv8 distinguishes itself through its user-friendly design and specific development tailored for industrial applications.

2.2.2. Single shot multibox detector

SSD [38] is a one-stage detector, close to R-CNN in terms of accuracy, but faster. SSD introduces multi-scale detection, default boxes and aspect ratios. As there are usually more negative predictions than positive, it introduces hard negative mining. This technique consists of maintaining a 3:1 ratio between negatives and positives.

Originally, the feature extractor used was VGG-16, but in subsequent works Mobilenet was introduced [39]. This feature extractor changes speeds up the inference process, especially in embedded devices.

2.2.3. EfficientDet

Many detectors sacrifice inference speed to improve accuracy. EfficientDet [40], however, improves the accuracy of other state-of-the-art detectors without sacrificing their high efficiency. To this end, EfficientDet introduces several improvements. Feature fusion combines representations of a given image at different resolutions, using a bidirectional feature pyramid network (BiFPN). Model scaling is another important breakthrough of EfficientDet. Rather than using larger backbones or input sizes to improve accuracy, the feature extractor network and the class and box predictor are scaled up, which also improves efficiency.

2.3. Evaluation metrics

The most common metric in object detection is Average Precision (AP) [21]. Before explaining it, it is necessary to mention other basic metrics.

- TP (True positives): objects correctly predicted.
- FP (False positives): objects incorrectly predicted.
- FN (False negatives): objects incorrectly unpredicted.
- TN (True negatives): objects correctly unpredicted. In object detection, this metric is not used because in each image these would be infinite TNs.

In object detection, it is not easy to decide if a prediction is correct since it rarely matches its corresponding ground truth completely. Thus, the IOU (Intersection Over the Union) is used. The IOU, shown in Eq. (1), measures the degree of overlap between two regions. If it is over a threshold, typically 0.5, the prediction is considered as correct.

After collecting the TPs, FPs and FNs for all the test images with a certain IOU threshold, precision and recall can be calculated. Precision, shown in Eq. (2), measures how accurate the predictions made are. Recall, shown in Eq. (3), measures the percentage of correct predictions from the total of objects to be predicted.

$$IOU = \frac{|D \cap G|}{|D \cup G|} \tag{1}$$

$$Precision = \frac{TPs}{TPs + FPs} \tag{2}$$

$$Recall = \frac{TPs}{TPs + FNs} \tag{3}$$

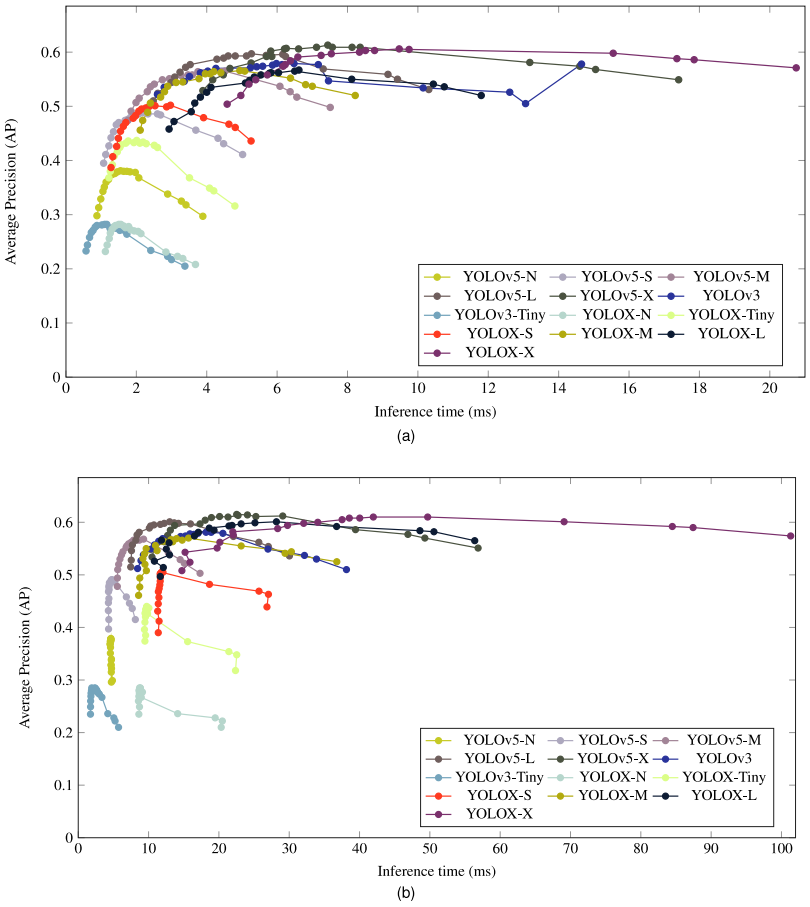

Fig. 1. Metrics varying input size on a NVIDIA RTX 2080Ti. Each mark represents an input size from 320 to 800 (in steps of 32), 1024, 1088, 1120 and 1216. (a) Using TensorRT. (b) Using Pytorch.

The predictions made are given a confidence score, which requires a minimum threshold. If the confidence score of a prediction surpasses this threshold, it is included in the metrics calculation. Otherwise, it is discarded. If a high confidence threshold is set, precision will be high and recall low. If a low confidence threshold is set, recall will be high, and precision will be low. Since it is complicated to compare models with all this casuistry, AP is used.

To calculate AP, simply requires computing precision and recall for several confidence thresholds. The AP is the area under the precision–recall curve. As mentioned above, these metrics depend on the chosen IOU threshold, so it is possible to calculate the AP in several ways:

- AP: computing the AP individually with the IOUs in the interval [0.5, 0.95] with a step of 0.05, and then averaging all the APs.
- AP_{50}: computing the AP only with an IOU of 0.5.

In this work, both APs are used to evaluate the models on edge computing devices. A modified version of this toolkit [41] has been used.

3. Results

Many works where edge computing devices are evaluated use their own datasets. It is impossible to compare devices in terms of precision with custom datasets, so this work uses MS COCO [42] to compare the results. The idea of using a general-purpose dataset is that the data can be reproduced in other studies. In addition, when using a large amount of data, the results are very reliable.

Inference times and APs are calculated using floating point arithmetic of 16 bits (FP16) and a batch size of one. This is because the objective is to process images in real-time, which is the maximum image acquisition rate offered by the camera.

To measure the times, all models were run 10 times on the validation set.

After making a hyperparametric adjustment and using an input size of 640 × 640 pixels, the models were optimized for the MS COCO dataset following the official guidelines [34,35,43].

3.1. NVIDIA RTX 2080Ti

When analyzing the performance of an object detection model, there are three aspects to consider: the accuracy of the model, its speed on the running device and power consumption. The accuracy of the model is independent of the device but may change when transformed for use with another framework. The frameworks influence the inference time since some of them are optimized for specific hardware. In the experiments analyzed, the models were evaluated using different input sizes. It was observed that Pytorch gives slightly better results than TensorRT in terms of AP, however, in terms of inference speed TensorRT seems to be a better choice. Table 2 shows the results of evaluating YOLOv3, YOLOv5 and YOLOX, with all their variants, on a NVIDIA RTX 2080Ti. In Fig. 1 the AP is plotted against the inference time. The optimal point is in the upper left corner since the AP is maximum and the inference time minimum.

Of the models evaluated, YOLOv5-X offers the best results in terms of AP and AP_{50}. The best result is achieved using an input size of 704 × 704 pixels. With higher or lower input sizes the AP and AP_{50} decrease. YOLOX-X gives similar (although slightly lower) results. In this case, the best result is obtained with an input size of 800 × 800 pixels.

There are significant differences in inference times between using TensorRT and Pytorch with a batch size of 1. With TensorRT, almost all models meet the requirement of processing images in real-time. To do this, models must run at 30 FPS (Frames Per Second), which is an arbitrary value used as a reference in this study because it is the most common in the low-cost cameras frequently used in embedded systems with edge devices. With TensorRT it does not exceed 21 ms (see Fig. 1(a) and Table 2). This means that the inference time is less than 33 ms, and therefore real-time. The same is not true for Pytorch. With Pytorch the inference times exceed 100 ms for YOLOX-X (see Fig. 1(b) and Table 2), YOLOv3-Tiny, as well as with YOLOv5-N, offer the best inference times. However, because the inference times of YOLOv5-X or YOLOX-X with TensorRT do not exceed the 33 ms limit for real-time processing and offer the best results in terms of AP, the choice of this model, as well as YOLOv3, YOLOv5-N, S, M, L and YOLOX-N, Tiny, S, M, L, is pointless in this device.

3.2. NVIDIA Jetson AGX Xavier Developer Kit

The conclusions obtained with the NVIDIA 2080 Ti cannot be extrapolated to the NVIDIA Jetson AGX Xavier, as this is an edge computing device with a GPU with limited performance.

Fig. 2 and Table 3 show the results obtained. In terms of AP there are no differences with the NVIDIA RTX 2080 Ti results. However, the same is not true in terms of inference speed. YOLOv5-N, YOLOX-N and YOLOv3-Tiny are the only models that do not exceed 33 ms for all input sizes (with TensorRT).

Other models such as YOLOv5-S, YOLOv5-M, YOLOX-Tiny, YOLOX-S, YOLOX-M or YOLOv3-Tiny exceed 33 ms for certain input sizes. Even if the input size is reduced, they still offer better results than simpler models. Intermediate models such as YOLOv5-L, YOLOX-L or YOLOv3 with reduced input sizes offer results below 33 ms. However, other simpler models offer similar results in terms of speed, but better results in terms of AP using larger input sizes. More complex models such as YOLOv5-X or YOLOX-X exceed the 33 ms barrier for practically all input sizes.

In this device, the difference between using Pytorch and TensorRT is more noticeable than for the NVIDIA RTX 2080 Ti. This difference is more significant as the input size increases and with complex models. For example, with YOLOX-X and an input size of 1216×1216 pixels, the difference between TensorRT and Pytorch is about 960 ms.

3.3. NVIDIA Jetson Nano Developer Kit

The NVIDIA Jetson Nano is a lower category device than the NVIDIA Jetson AGX Xavier, which is noted in Fig. 3 and Table 4.

The goal of having an inference speed of less than 33 ms is only met by the simplest models, with reduced input sizes and using TensorRT. This means that to use this device while processing images in real-time (30 FPS), the accuracy is reduced. If real-time processing is not necessary, this device can be used with models of medium complexity. It was not possible to evaluate the more complex models, such as YOLOv5-X and YOLOX-X, due to lack of memory.

3.4. Google Coral Dev Board

The Google Coral Dev Board, unlike the devices described above, includes a TPU. For this reason, the Tensorflow Lite (TFLite) framework is used. TensorRT cannot be used since this device does not have a NVIDIA GPU. The results can be seen in Table 5.

After analyzing the results obtained with YOLOv5-N, it was decided not to continue with more complex models since the inference times are too high. In addition, YOLOv5-N has a lower AP than TensorRT or Pytorch. This is because in the process of transforming the model from Pytorch to TFLite, the model is quantized to 8 bits for the representation of floating point (FP) numbers.

Models specifically designed for TFLite have also been tested. These models are SSD-Mobilenet v1 with TF1 and TF2, SSD-Mobilenet v1 with TF2 and EfficientDet-Lite 1 with TF2. These models, except for EfficientDet-Lite 1, offer reduced inference times but with a significantly lower AP than YOLOv5-N. EfficientDet-Lite 1 performs better than YOLOv5-N in this device, as it is faster and outperforms it in AP.

4. Discussion

Throughout this section, the devices analyzed are compared taking into account power consumption, real-time image processing, the use of devices without hardware accelerators and the execution of these models in the cloud.

4.1. Influence of model size on results

Model size has a big influence on results. The bigger the size, the more coefficients the model will have to work with, which will allow it to solve more complex problems. However, if the model is too big it can be overfit to the training data, which can lead to worse results. Therefore, it is important to find a balance between model size and performance. In addition, the bigger the model, the longer it will take to process an image. Fig. 4 shows a linear growth in inference times on all devices. However, it is important to note that in the case of the NVIDIA Jetson Nano, the obtained slope is much higher than that of the NVIDIA Jetson AGX Xavier and the NVIDIA RTX 2080 Ti. This means that using larger models will affect it more. Currently, there is a trend to create models with more layers to solve complex problems. Although inference times also depend on future architectural improvements of networks, this analysis serves to understand the possible scalability of these devices. For example, if using a model with 200 million parameters, the inference time on a NVIDIA Jetson Nano would be approximately 1266 ms, while on the NVIDIA Jetson AGX Xavier it would be 161 ms and on the NVIDIA RTX 2080 Ti it would be 14 ms.

4.2. Inference with new models

In recent years, there has been a notable surge in the development of models for object detection in images. This trend suggests that new versions of detectors will continue to emerge in the coming years. Given the proliferation of models, it is impractical to evaluate all existing ones. Currently, there is a leaning towards incorporating object detection models based on transformers due to their exceptional results. YOLOv8 stands out as a prominent example of such detectors.

The results obtained with YOLOv8 on the NVIDIA RTX 2080Ti are presented in Table 6. It is observed that, in terms of inference time, the results are very similar to those of YOLOv5. The key to understanding these results lies primarily in the number of parameters in the model, as this significantly affects the inference time. Considering this aspect and the information provided in Fig. 4, inference times can be extrapolated for future models.

4.3. Power consumption

The power consumption of these devices is mainly caused by the performance of the CPU and GPU. In the measurements carried out, a real scenario has been considered in which inference is performed constantly. Therefore, CPU and GPU usage is almost continuous, except for input/output (I/O) operations. The GPU usage percentage varies depending on the device and the model employed. Overall, the NVIDIA 2080Ti has a utilization of 27%, the NVIDIA Jetson AGX Xavier is at 44%, and the NVIDIA Jetson Nano is at 55%. The percentage does not reach 100% because there is a preprocessing and postprocessing component carried out on the CPU. Considering this context, Table 7 shows the power consumption of each device measured with the Brennenstuhl EM 230 W m. The NVIDIA RTX 2080Ti has the highest consumption

Table 2
Metrics of all models by varying the input size over the NVIDIA RTX 2080Ti.

Model	Size	AP_{50}	AP	Inference time (ms)		Model	Size	AP_{50}	AP	Inference time (ms)		Model	Size	AP_{50}	AP	Inference time (ms)	
				TensorRT	Pytorch					TensorRT	Pytorch					TensorRT	Pytorch
YOLOv5-N	1216	0.368	0.297	3.89	4.86	YOLOX-N	1216	0.271	0.208	3.68	20.33	YOLOX-Tiny	1216	0.388	0.316	4.80	22.36
	1120	0.394	0.318	3.41	4.73		1120	0.292	0.219	3.32	20.49		1120	0.425	0.344	4.20	22.54
	1088	0.401	0.325	3.28	4.68		1088	0.299	0.223	3.17	19.47		1088	0.430	0.349	4.08	21.42
	1024	0.415	0.338	2.89	4.70		1024	0.309	0.231	2.84	14.14		1024	0.450	0.368	3.51	15.55
	800	0.445	0.368	2.07	4.57		800	0.348	0.265	2.13	8.96		800	0.504	0.424	2.60	9.86
	768	0.456	0.378	1.97	4.67		768	0.352	0.269	2.05	8.88		768	0.508	0.428	2.51	9.77
	736	0.457	0.379	1.82	4.65		736	0.358	0.270	1.94	8.89		736	0.507	0.431	2.28	9.78
	704	0.457	0.380	1.73	4.64		704	0.360	0.272	1.84	9.13		704	0.511	0.434	2.19	10.04
	672	0.459	0.380	1.64	4.61		672	0.364	0.278	1.78	8.82		672	0.509	0.432	2.10	9.70
	640	0.458	0.381	1.55	4.60		640	0.360	0.276	1.69	8.85		640	0.512	0.437	2.01	9.74
	608	0.455	0.379	1.46	4.63		608	0.366	0.279	1.64	8.84		608	0.511	0.433	1.93	9.72
	576	0.452	0.376	1.39	4.57		576	0.365	0.282	1.56	8.83		576	0.509	0.436	1.77	9.71
	544	0.448	0.374	1.31	4.56		544	0.362	0.282	1.50	8.79		544	0.508	0.432	1.70	9.67
	512	0.439	0.365	1.21	4.51		512	0.363	0.280	1.41	8.72		512	0.504	0.429	1.57	9.59
	480	0.433	0.360	1.14	4.59		480	0.355	0.278	1.37	8.63		480	0.495	0.424	1.53	9.49
	448	0.423	0.351	1.09	4.57		448	0.350	0.273	1.31	8.69		448	0.487	0.416	1.46	9.56
	416	0.413	0.343	1.05	4.72		416	0.339	0.266	1.26	8.62		416	0.477	0.408	1.37	9.48
	384	0.397	0.329	0.99	4.66		384	0.329	0.256	1.23	8.56		384	0.460	0.392	1.32	9.42
	352	0.379	0.313	0.93	4.70		352	0.315	0.244	1.17	8.71		352	0.446	0.380	1.26	9.58
	320	0.363	0.298	0.88	4.71		320	0.299	0.232	1.12	8.62		320	0.419	0.368	1.22	9.48
YOLOv5-S	1216	0.491	0.411	5.02	8.12	YOLOX-S	1216	0.520	0.436	5.26	26.84	YOLOv3-Tiny	1216	0.273	0.205	3.38	5.73
	1120	0.515	0.431	4.48	7.68		1120	0.543	0.461	4.81	27.05		1120	0.289	0.217	3.00	5.24
	1088	0.528	0.441	4.32	7.28		1088	0.550	0.467	4.63	25.70		1088	0.298	0.223	2.89	5.06
	1024	0.541	0.456	3.69	6.84		1024	0.558	0.479	3.91	18.66		1024	0.309	0.234	2.41	4.20
	800	0.566	0.484	2.69	5.20		800	0.578	0.502	2.97	11.83		800	0.350	0.264	1.73	3.37
	768	0.568	0.486	2.60	4.87		768	0.576	0.499	2.86	11.72		768	0.355	0.271	1.66	3.01
	736	0.565	0.486	2.33	4.72		736	0.576	0.501	2.53	11.73		736	0.355	0.271	1.53	2.86
	704	0.566	0.488	2.17	4.78		704	0.577	0.500	2.37	12.05		704	0.359	0.275	1.41	2.75
	672	0.565	0.486	2.06	4.68		672	0.572	0.497	2.27	11.64		672	0.365	0.276	1.33	2.63
	640	0.560	0.482	1.96	4.50		640	0.567	0.495	2.15	11.68		640	0.359	0.275	1.25	2.52
	608	0.559	0.480	1.84	4.43		608	0.564	0.491	2.07	11.67		608	0.365	0.282	1.16	2.45
	576	0.556	0.476	1.76	4.39		576	0.559	0.483	1.98	11.66		576	0.366	0.282	1.10	2.37
	544	0.550	0.474	1.68	4.35		544	0.549	0.478	1.91	11.60		544	0.363	0.281	1.02	2.31
	512	0.546	0.470	1.50	4.39		512	0.541	0.470	1.70	11.51		512	0.362	0.280	0.88	1.92
	480	0.542	0.466	1.43	4.30		480	0.534	0.463	1.63	11.39		480	0.357	0.277	0.83	1.88
	448	0.526	0.453	1.35	4.36		448	0.523	0.454	1.55	11.47		448	0.350	0.271	0.77	1.82
	416	0.519	0.442	1.28	4.29		416	0.510	0.441	1.49	11.38		416	0.341	0.267	0.72	1.80
	384	0.501	0.427	1.22	4.28		384	0.496	0.426	1.44	11.30		384	0.327	0.258	0.67	1.76
	352	0.485	0.411	1.13	4.35		352	0.473	0.407	1.33	11.50		352	0.318	0.244	0.61	1.75
	320	0.466	0.395	1.07	4.31		320	0.449	0.387	1.28	11.38		320	0.296	0.233	0.57	1.74
YOLOv5-M	1216	0.582	0.498	7.51	17.32	YOLOX-M	1216	0.604	0.520	8.22	36.77	YOLOv3	1216	0.590	0.505	14.65	38.14
	1120	0.603	0.517	6.56	15.08		1120	0.617	0.537	7.00	29.41		1120	0.610	0.527	13.06	33.88
	1088	0.609	0.527	6.37	14.49		1088	0.620	0.540	6.81	30.31		1088	0.616	0.535	12.61	32.17
	1024	0.620	0.537	6.08	12.81		1024	0.631	0.552	6.38	23.17		1024	0.628	0.547	10.15	26.98
	800	0.637	0.565	4.52	9.29		800	0.638	0.565	5.08	15.68		800	0.652	0.575	7.46	20.57
	768	0.638	0.564	4.42	8.69		768	0.640	0.566	4.94	13.99		768	0.653	0.578	7.17	18.92
	736	0.638	0.562	3.93	8.28		736	0.636	0.562	4.40	13.20		736	0.655	0.577	6.48	18.75
	704	0.634	0.564	3.75	7.86		704	0.636	0.564	4.20	14.78		704	0.654	0.579	6.14	18.17
	672	0.635	0.558	3.59	7.54		672	0.632	0.560	4.10	12.15		672	0.647	0.575	5.98	17.03
	640	0.633	0.558	3.47	7.47		640	0.631	0.560	3.96	13.44		640	0.648	0.575	5.87	16.44
	608	0.628	0.557	3.27	7.16		608	0.622	0.551	3.78	10.98		608	0.647	0.575	5.59	15.88
	576	0.625	0.551	2.94	6.90		576	0.618	0.545	3.32	9.66		576	0.644	0.569	5.42	15.25
	544	0.620	0.549	2.74	6.79		544	0.613	0.544	3.13	11.21		544	0.643	0.568	5.24	14.55
	512	0.613	0.541	2.51	6.35		512	0.607	0.537	2.89	9.43		512	0.639	0.567	4.26	12.01
	480	0.608	0.536	2.40	6.15		480	0.598	0.527	2.80	9.20		480	0.635	0.562	4.03	11.44
	448	0.601	0.527	2.31	5.93		448	0.585	0.517	2.69	9.49		448	0.629	0.554	3.82	10.84
	416	0.588	0.515	2.08	5.75		416	0.572	0.506	2.40	9.69		416	0.620	0.547	3.51	10.38
	384	0.581	0.507	1.99	5.61		384	0.557	0.489	2.32	8.75		384	0.608	0.536	3.36	9.26
	352	0.562	0.491	1.85	5.57		352	0.542	0.474	2.18	8.75		352	0.597	0.523	2.79	8.92
	320	0.544	0.474	1.76	5.56		320	0.521	0.456	2.11	8.58		320	0.584	0.507	2.60	8.44
YOLOv5-L	1216	0.610	0.531	10.31	30.03	YOLOX-L	1216	0.604	0.520	11.80	56.38						
	1120	0.632	0.550	9.42	27.03		1120	0.616	0.536	10.75	50.58						
	1088	0.641	0.559	9.15	25.68		1088	0.619	0.541	10.44	48.59						
	1024	0.648	0.569	7.31	22.06		1024	0.630	0.550	8.12	36.74						
	800	0.664	0.592	6.24	15.96		800	0.639	0.565	6.50	28.18						
	768	0.668	0.596	6.16	14.25		768	0.640	0.567	6.62	25.13						
	736	0.666	0.594	5.69	13.65		736	0.636	0.562	6.04	23.16						
	704	0.667	0.597	5.27	13.00		704	0.635	0.562	5.83	21.87						
	672	0.666	0.593	5.13	12.23		672	0.632	0.561	5.78	21.43						
	640	0.661	0.593	4.81	11.67		640	0.631	0.558	5.54	18.63						
	608	0.661	0.593	4.61	10.76		608	0.624	0.554	5.35	19.31						
	576	0.660	0.589	4.48	10.29		576	0.620	0.548	5.22	17.08						
	544	0.657	0.587	4.29	10.11		544	0.614	0.543	5.10	16.53						
	512	0.646	0.577	3.53	8.67		512	0.608	0.535	4.12	11.84						
	480	0.643	0.572	3.40	8.36		480	0.599	0.526	4.00	12.92						
	448	0.633	0.563	3.24	8.19		448	0.585	0.517	3.82	12.52						
	416	0.626	0.554	3.09	7.70		416	0.574	0.506	3.66	12.98						
	384	0.614	0.544	2.99	7.57		384	0.558	0.490	3.56	10.83						
	352	0.596	0.524	2.61	7.54		352	0.539	0.472	3.07	12.10						
	320	0.581	0.510	2.48	7.46		320	0.522	0.458	2.93	11.67						
YOLOv5-X	1216	0.623	0.549	17.41	56.86	YOLOX-X	1216	0.651	0.571	20.75	101.31						
	1120	0.642	0.568	15.05	49.29		1120	0.661	0.586	17.85	87.43						
	1088	0.650	0.574	14.60	46.88		1088	0.665	0.588	17.36	84.48						
	1024	0.658	0.581	13.18	39.43		1024	0.674	0.598	15.55	69.09						
	800	0.681	0.609	8.36	29.08		800	0.676	0.605	9.75	49.69						
	768	0.679	0.609	8.13	25.26		768	0.677	0.606	9.47	41.96						
	736	0.682	0.609	7.57	24.06		736	0.676	0.603	8.77	40.03						
	704	0.683	0.613	7.44	22.53		704	0.675	0.603	8.51	38.55						
	672	0.681	0.609	7.12	22.73		672	0.671	0.600	8.34	37.52						
	640	0.678	0.606	6.61	21.30		640	0.666	0.597	7.54	34.06						
	608	0.677	0.607	6.26	20.10		608	0.665	0.594	7.25	32.06						
	576	0.675	0.606	6.21	18.94		576	0.660	0.591	6.59	29.76						
	544	0.670	0.602	5.82	18.05		544	0.655	0.584	6.38	28.36						
	512	0.664	0.592	5.85	17.32		512	0.647	0.577	6.23	21.98						
	480	0.662	0.592	5.70	13.71		480	0.642	0.574	6.17	22.04						
	448	0.652	0.580	5.26	13.10		448	0.627	0.558	5.72	20.14						
	416	0.645	0.570	4.66	12.71		416	0.618	0.549	5.39	19.75						
	384	0.633	0.558	4.47	12.19		384	0.608	0.541	5.20	15.19						
	352	0.619	0.549	4.17	10.84		352	0.589	0.520	4.97	15.89						
	320	0.603	0.529	3.89	10.49		320	0.570	0.504	4.58	14.74						

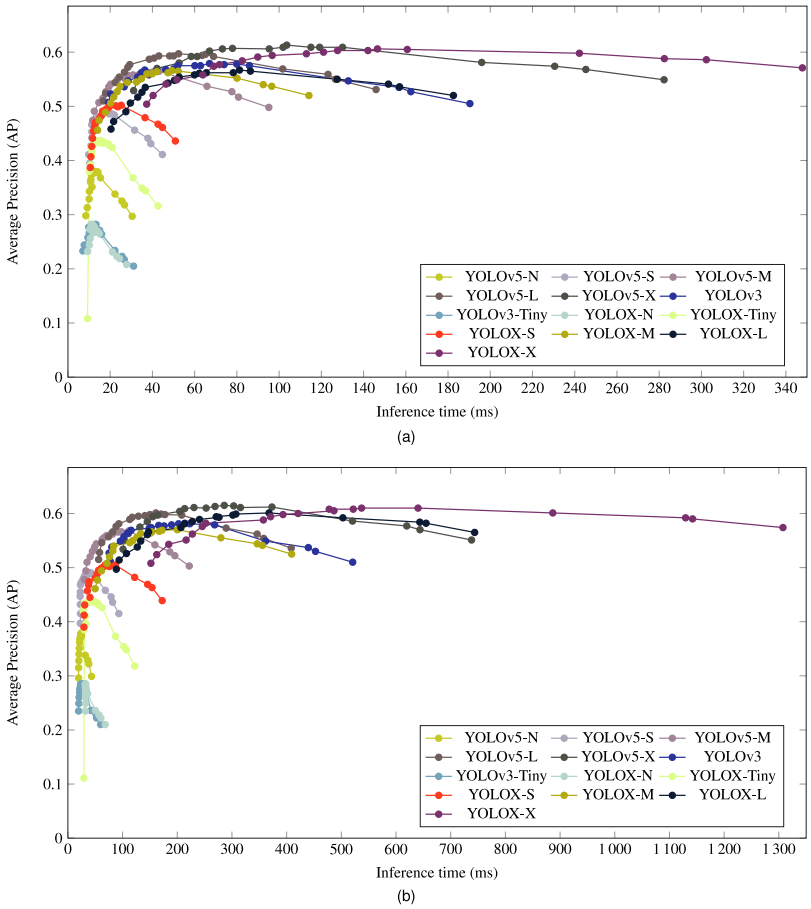

Fig. 2. Metrics varying input size on a NVIDIA Jetson AGX Xavier. Each mark represents an input size from 320 to 800 (in steps of 32), 1024, 1088, 1120 and 1216. (a) Using TensorRT. (b) Using Pytorch.

but is the fastest device. For edge computing devices the Google Coral Dev Board has the lowest consumption, with 4.5 W. This low power consumption could compensate for the low AP obtained with models capable of real-time image processing on this device in scenarios where there are power supply restrictions. The NVIDIA Jetson AGX Xavier and NVIDIA Jetson Nano have intermediate power consumptions, so when, there are no extreme power constraints, they offer better results (both inference speed and AP) than the Google Coral Dev Board.

4.4. Real-time image processing

As mentioned above, to process images in real-time (30 FPS) the inference time cannot exceed 33 ms. Table 8 lists the models with the highest APs that do not exceed this time for each device. It seems clear that, except for the Google Coral Dev Board, which has special characteristics, YOLOv5 offers the best results. The difference is given by the AP difference between the different versions of YOLOv5 and the input size. YOLOv5-X with an input size of 704 × 704 pixels offers a 4% higher AP than YOLOv5-L with an input size of 512 × 512, and 21% higher than YOLOv5-S at 320 × 320.

4.5. Real-time people detection

Within the wide range of possibilities offered by this technology, one of the most common is the detection of people. This involves the creation of systems oriented to the autonomous detection or security control of a given space. Since the person class is one of the classes in the MS COCO dataset, this section evaluates the use of these devices to detect people using the models with the highest AP for each device. Table 9 shows the APs of each model for the person class. The results, in terms of AP, are better than when averaging the results of the 80 classes (see Table 8). Between YOLOv5-X and YOLOv5-L there are no significant differences, but with YOLOv5-S and SSD-Mobilenet v2 the differences are immense. Visually, this difference can be seen in Fig. 5, where three examples of person detection are shown for each model. YOLOv5-X and YOLOv5-L correctly detect most people, including those in the background. YOLOv5-S detects well the people in the foreground but not those in the background (see Fig. 5j). SSD-Mobilenet v2 also fails to detect people in the background as well as generating a false positive (see Fig. 5n).

4.6. FPS/Watt and FPS/€ comparison

The selection of a device may be determined by inference speed, power consumption or cost. For this reason, it is interesting to know the FPS/Watt and FPS/€ ratio of each of the selected devices. Table 10 shows these ratios, having selected the fastest models with a minimum AP_{50} of 0.4. Under these conditions, the device that processes images the fastest in terms of power consumption is the NVIDIA Jetson AGX Xavier followed by the NVIDIA Jetson Nano. In terms of cost, the NVIDIA RTX 2080Ti (€1169.01) is the most recommended device (note that, in this case, the GPU would have to be integrated into a PC, with the additional costs that this entails), followed by the NVIDIA Jetson Nano. These results have a minimum AP_{50} of 0.4. For more demanding requirements, setting a minimum AP_{50} of 0.65, the results vary. These differences can be seen in Table 11. The Google Coral Dev Board is not an option, since none of the models evaluated in Section 3.4 reach the minimum AP_{50}. Since the stated AP_{50} is higher, it is necessary to use more complex models and larger input sizes to obtain the desired results. This results in lower FPS. In this scenario, the NVIDIA Jetson

Table 3
Metrics of all models by varying the input size over the NVIDIA Jetson AGX Xavier.

Model	Size	AP_{50}	AP	Inference time (ms)		Model	Size	AP_{50}	AP	Inference time (ms)		Model	Size	AP_{50}	AP	Inference time (ms)	
				TensorRT	Pytorch					TensorRT	Pytorch					TensorRT	Pytorch
	1216	0.368	0.297	30.40	43.21		1216	0.271	0.208	27.80	68.17		1216	0.388	0.316	42.65	122.24
	1120	0.394	0.318	26.78	38.33		1120	0.292	0.219	24.41	60.39		1120	0.425	0.344	36.76	106.79
	1088	0.401	0.325	25.56	36.49		1088	0.299	0.223	23.17	56.65		1088	0.430	0.349	35.04	102.14
	1024	0.415	0.338	22.27	31.88		1024	0.309	0.231	21.09	50.23		1024	0.450	0.368	30.80	87.09
	800	0.445	0.368	15.39	24.25		800	0.348	0.265	14.88	36.09		800	0.504	0.424	20.92	62.51
	768	0.456	0.378	14.24	23.61		768	0.352	0.269	14.14	33.32		768	0.508	0.428	19.61	55.90
	736	0.457	0.379	13.80	24.52		736	0.358	0.270	13.30	32.92		736	0.507	0.431	19.05	54.63
	704	0.457	0.380	12.11	23.98		704	0.360	0.272	13.00	32.46		704	0.511	0.434	16.92	49.36
	672	0.459	0.380	12.86	24.29		672	0.364	0.278	12.44	32.40		672	0.509	0.432	16.62	47.79
YOLOv5-N	640	0.458	0.381	11.33	24.64	YOLOX-N	640	0.360	0.276	12.09	32.44	YOLOX-Tiny	640	0.512	0.437	15.47	43.16
	608	0.455	0.379	13.72	23.81		608	0.366	0.279	11.94	32.12		608	0.511	0.433	14.53	40.52
	576	0.452	0.376	11.97	23.14		576	0.365	0.282	11.57	32.19		576	0.509	0.436	13.71	39.56
	544	0.448	0.374	11.12	22.81		544	0.362	0.282	10.98	32.10		544	0.508	0.432	12.46	36.82
	512	0.439	0.365	10.87	21.91		512	0.363	0.280	11.89	32.17		512	0.504	0.429	11.79	30.15
	480	0.433	0.360	10.71	21.43		480	0.355	0.278	11.19	31.88		480	0.495	0.424	10.81	29.46
	448	0.423	0.351	11.41	20.92		448	0.350	0.273	10.27	32.38		448	0.487	0.416	11.42	27.91
	416	0.413	0.343	10.15	20.19		416	0.339	0.266	11.89	32.03		416	0.477	0.408	11.15	32.58
	384	0.397	0.329	10.05	20.02		384	0.329	0.256	10.57	32.16		384	0.460	0.392	10.70	32.77
	352	0.379	0.313	9.14	19.66		352	0.315	0.244	10.10	32.16		352	0.446	0.380	10.05	30.18
	320	0.363	0.298	8.44	19.58		320	0.299	0.232	9.33	32.07		320	0.419	0.368	9.27	29.22
	1216	0.491	0.411	44.72	93.23		1216	0.520	0.436	50.87	172.46		1216	0.273	0.205	31.07	59.81
	1120	0.515	0.431	39.25	81.70		1120	0.543	0.461	44.75	153.87		1120	0.289	0.217	26.70	52.55
	1088	0.528	0.441	37.87	78.92		1088	0.550	0.467	42.66	145.86		1088	0.298	0.223	25.62	51.50
	1024	0.541	0.456	31.56	68.47		1024	0.558	0.479	36.50	122.15		1024	0.309	0.234	22.23	42.32
	800	0.566	0.484	21.98	47.97		800	0.578	0.502	25.23	86.88		800	0.350	0.264	15.93	34.40
	768	0.568	0.486	20.87	42.72		768	0.576	0.499	23.67	76.01		768	0.355	0.271	15.01	29.14
	736	0.565	0.486	19.56	41.36		736	0.576	0.501	22.72	74.58		736	0.355	0.271	13.83	27.86
	704	0.566	0.488	17.10	38.67		704	0.577	0.500	19.72	68.24		704	0.359	0.275	12.03	27.24
	672	0.565	0.486	16.40	35.95		672	0.572	0.497	19.11	66.11		672	0.365	0.276	12.07	26.73
YOLOv5-S	640	0.560	0.482	15.88	33.73	YOLOX-S	640	0.567	0.495	18.13	61.25	YOLOv3-Tiny	640	0.359	0.275	12.28	26.13
	608	0.559	0.480	14.32	30.55		608	0.564	0.491	16.39	56.07		608	0.365	0.282	12.68	26.11
	576	0.556	0.476	13.45	28.43		576	0.559	0.483	15.77	53.41		576	0.366	0.282	13.31	27.32
	544	0.550	0.474	13.70	27.35		544	0.549	0.478	15.02	50.06		544	0.363	0.281	12.64	26.19
	512	0.546	0.470	11.74	24.07		512	0.541	0.470	12.94	38.82		512	0.362	0.280	11.25	23.16
	480	0.542	0.466	11.20	22.92		480	0.534	0.463	12.53	37.80		480	0.357	0.277	9.81	22.80
	448	0.526	0.453	11.31	22.63		448	0.523	0.454	12.14	35.69		448	0.350	0.271	10.93	21.53
	416	0.519	0.442	11.30	22.42		416	0.510	0.441	11.65	40.33		416	0.341	0.267	10.07	21.50
	384	0.501	0.427	10.82	22.68		384	0.496	0.426	11.29	30.69		384	0.327	0.258	9.43	20.41
	352	0.485	0.411	9.80	22.97		352	0.473	0.407	10.85	29.71		352	0.318	0.244	7.69	19.90
	320	0.466	0.395	10.36	22.55		320	0.449	0.387	10.56	29.29		320	0.296	0.233	7.06	19.49
	1216	0.582	0.498	95.13	221.95		1216	0.604	0.520	114.12	409.10		1216	0.590	0.505	190.35	520.83
	1120	0.603	0.517	80.77	195.55		1120	0.617	0.537	96.51	355.76		1120	0.610	0.527	162.39	452.74
	1088	0.609	0.527	77.64	186.50		1088	0.620	0.540	92.52	346.24		1088	0.616	0.535	156.79	439.70
	1024	0.620	0.537	65.85	159.13		1024	0.631	0.552	80.05	279.78		1024	0.628	0.547	132.77	361.47
	800	0.637	0.565	43.23	111.77		800	0.638	0.565	52.24	200.09		800	0.652	0.575	85.94	268.28
	768	0.638	0.564	40.56	98.19		768	0.640	0.566	49.24	172.17		768	0.653	0.578	79.94	222.95
	736	0.638	0.562	38.92	93.51		736	0.636	0.562	47.38	167.06		736	0.655	0.577	73.94	213.79
	704	0.634	0.564	35.84	89.38		704	0.636	0.564	42.12	153.56		704	0.654	0.579	67.04	202.36
	672	0.635	0.558	33.36	81.95		672	0.632	0.560	39.96	151.43		672	0.647	0.575	62.76	189.52
YOLOv5-M	640	0.633	0.558	30.89	77.12	YOLOX-M	640	0.631	0.560	37.39	132.98	YOLOv3	640	0.648	0.575	60.09	175.90
	608	0.628	0.557	28.55	70.45		608	0.622	0.551	34.81	126.33		608	0.647	0.575	51.97	166.00
	576	0.625	0.551	26.92	65.46		576	0.618	0.545	32.46	118.61		576	0.644	0.569	50.04	152.34
	544	0.620	0.549	23.29	62.67		544	0.613	0.544	28.10	113.13		544	0.643	0.568	46.29	148.81
	512	0.613	0.541	20.27	52.01		512	0.607	0.537	24.70	83.77		512	0.639	0.567	36.39	115.59
	480	0.608	0.536	19.33	48.69		480	0.598	0.527	23.29	82.23		480	0.635	0.562	34.73	109.14
	448	0.601	0.527	17.95	44.87		448	0.585	0.517	21.70	75.99		448	0.629	0.554	33.00	102.46
	416	0.588	0.515	16.52	40.70		416	0.572	0.506	20.18	71.56		416	0.620	0.547	28.46	95.96
	384	0.581	0.507	14.63	34.99		384	0.557	0.489	17.72	61.44		384	0.608	0.536	26.62	80.21
	352	0.562	0.491	12.48	32.96		352	0.542	0.474	14.75	53.96		352	0.597	0.523	20.24	75.39
	320	0.544	0.474	11.72	30.13		320	0.521	0.456	14.04	49.77		320	0.584	0.507	19.19	77.35
	1216	0.610	0.531	145.92	408.39		1216	0.604	0.520	182.55	743.80						
	1120	0.632	0.550	126.65	358.01		1120	0.616	0.536	157.05	654.94						
	1088	0.641	0.559	123.17	346.03		1088	0.619	0.541	151.77	643.51						
	1024	0.648	0.569	101.72	289.33		1024	0.630	0.550	127.52	502.88						
	800	0.664	0.592	68.94	208.11		800	0.639	0.565	86.40	367.82						
	768	0.668	0.596	65.12	177.36		768	0.640	0.567	81.26	307.29						
	736	0.666	0.594	63.61	170.49		736	0.636	0.562	78.29	301.54						
	704	0.667	0.597	52.40	160.85		704	0.635	0.562	65.52	271.20						
	672	0.666	0.593	50.03	150.13		672	0.632	0.561	62.21	276.35						
YOLOv5-L	640	0.661	0.593	48.15	140.78	YOLOX-L	640	0.631	0.558	59.94	240.66						
	608	0.661	0.593	43.05	128.93		608	0.624	0.554	52.66	227.40						
	576	0.660	0.589	40.77	117.63		576	0.620	0.548	50.44	213.53						
	544	0.657	0.587	38.51	113.12		544	0.614	0.543	47.41	206.73						
	512	0.646	0.577	29.28	93.16		512	0.608	0.535	36.62	147.37						
	480	0.643	0.572	28.35	88.00		480	0.599	0.526	35.06	145.33						
	448	0.633	0.563	26.75	81.79		448	0.585	0.517	33.08	133.20						
	416	0.626	0.554	24.28	74.16		416	0.574	0.506	29.56	126.93						
	384	0.614	0.544	22.36	61.87		384	0.558	0.490	27.44	107.09						
	352	0.596	0.524	17.84	57.89		352	0.539	0.472	21.63	93.68						
	320	0.581	0.510	16.73	56.23		320	0.522	0.458	20.39	88.54						
	1216	0.623	0.549	282.31	738.13		1216	0.651	0.571	347.82	1307.59						
	1120	0.642	0.568	245.27	643.86		1120	0.661	0.586	302.34	1142.29						
	1088	0.650	0.574	230.51	619.64		1088	0.665	0.588	282.46	1129.55						
	1024	0.658	0.581	195.95	520.48		1024	0.674	0.598	242.22	886.70						
	800	0.681	0.609	130.05	373.17		800	0.676	0.605	160.70	640.32						
	768	0.679	0.609	119.19	315.62		768	0.677	0.606	146.58	537.05						
	736	0.682	0.609	115.06	303.22		736	0.676	0.603	142.08	521.85						
	704	0.683	0.613	103.62	285.91		704	0.675	0.603	127.68	477.86						
	672	0.681	0.609	101.89	269.03		672	0.671	0.600	121.04	486.61						
YOLOv5-X	640	0.678	0.606	95.33	252.76	YOLOX-X	640	0.666	0.597	112.93	421.01						
	608	0.677	0.607	77.92	230.60		608	0.665	0.594	96.73	393.33						
	576	0.675	0.606	73.16	213.55		576	0.660	0.591	89.91	370.59						
	544	0.670	0.602	67.05	204.56		544	0.655	0.584	82.52	357.46						
	512	0.664	0.592	61.19	164.08		512	0.647	0.577	71.75	252.12						
	480	0.662	0.592	58.37	155.17		480	0.642	0.574	69.04	245.71						

(*continued on next page*)

Table 3 (*continued*).

448	0.652	0.580	52.63	145.22	448	0.627	0.558	63.87	227.60
416	0.645	0.570	42.17	131.66	416	0.618	0.549	51.58	216.16
384	0.633	0.558	38.24	110.61	384	0.608	0.541	46.40	184.39
352	0.619	0.549	33.18	103.04	352	0.589	0.520	40.30	162.49
320	0.603	0.529	31.10	100.81	320	0.570	0.504	37.29	151.78

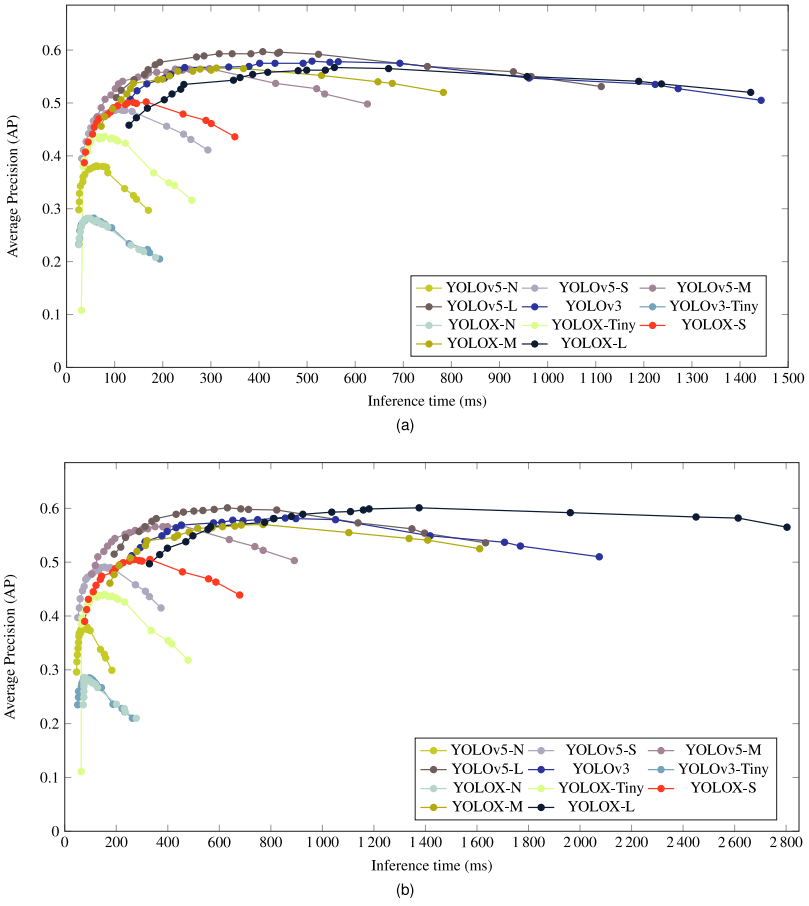

Fig. 3. Metrics varying input size on a NVIDIA Jetson Nano. Each mark represents an input size from 320 to 800 (in steps of 32), 1024, 1088, 1120 and 1216. (a) Using TensorRT. (b) Using Pytorch.

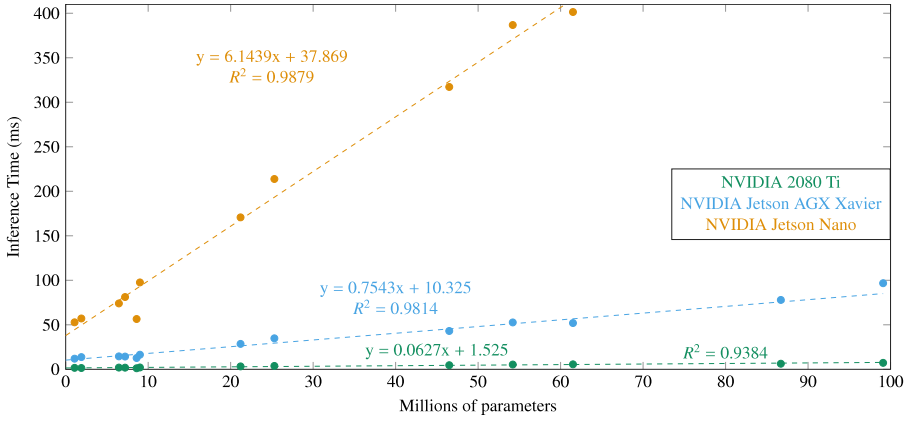

Fig. 4. Evolution of inference times as a function of the size of the models.

Table 4
Metrics of all models by varying the input size over the NVIDIA Jetson Nano.

Model	Size	AP_{50}	AP	Inference time (ms)		Model	Size	AP_{50}	AP	Inference time (ms)		Model	Size	AP_{50}	AP	Inference time (ms)	
				TensorRT	Pytorch					TensorRT	Pytorch					TensorRT	Pytorch
YOLOv5-N	1216	0.368	0.297	170.12	183.10	YOLOX-N	1216	0.271	0.208	184.90	277.56	YOLOX-Tiny	1216	0.388	0.316	260.24	478.75
	1120	0.394	0.318	145.35	158.90		1120	0.292	0.219	160.15	234.52		1120	0.425	0.344	224.40	415.49
	1088	0.401	0.325	138.98	153.61		1088	0.299	0.223	150.37	231.03		1088	0.430	0.349	212.78	401.23
	1024	0.415	0.338	120.77	138.53		1024	0.309	0.231	133.81	200.90		1024	0.450	0.368	180.97	335.34
	800	0.445	0.368	85.77	98.59		800	0.348	0.265	85.86	127.22		800	0.504	0.424	122.10	232.30
	768	0.456	0.378	83.32	87.46		768	0.352	0.269	82.02	119.13		768	0.508	0.428	106.07	206.63
	736	0.457	0.379	80.26	87.49		736	0.358	0.270	74.73	111.10		736	0.507	0.431	102.18	197.06
	704	0.457	0.380	65.29	74.40		704	0.360	0.272	70.59	103.41		704	0.511	0.434	94.08	182.38
	672	0.459	0.380	75.15	74.47		672	0.364	0.278	63.34	95.26		672	0.509	0.432	91.26	170.93
	640	0.458	0.381	61.81	72.32		640	0.360	0.276	58.35	86.84		640	0.512	0.437	78.59	155.06
	608	0.455	0.379	57.13	66.83		608	0.366	0.279	52.79	82.41		608	0.511	0.433	73.98	142.67
	576	0.452	0.376	50.13	62.48		576	0.365	0.282	48.02	77.82		576	0.509	0.436	67.89	130.39
	544	0.448	0.374	45.16	61.95		544	0.362	0.282	44.26	75.69		544	0.508	0.432	68.03	125.68
	512	0.439	0.365	38.12	56.16		512	0.363	0.280	38.72	74.97		512	0.504	0.429	50.62	100.76
	480	0.433	0.360	34.18	54.88		480	0.355	0.278	36.27	74.01		480	0.495	0.424	48.94	97.91
	448	0.423	0.351	33.80	53.71		448	0.350	0.273	33.29	74.61		448	0.487	0.416	44.73	87.21
	416	0.413	0.343	28.74	51.42		416	0.339	0.266	30.51	74.28		416	0.477	0.408	43.70	79.79
	384	0.397	0.329	26.77	49.15		384	0.329	0.256	28.55	74.19		384	0.460	0.392	37.11	66.70
	352	0.379	0.313	26.46	47.19		352	0.315	0.244	26.26	73.99		352	0.446	0.380	34.24	64.32
	320	0.363	0.298	25.39	45.94		320	0.299	0.232	25.41	71.97		320	0.419	0.368	30.76	63.78
YOLOv5-S	1216	0.491	0.411	293.79	373.64	YOLOX-S	1216	0.520	0.436	349.80	679.01	YOLOv3-Tiny	1216	0.273	0.205	193.42	262.34
	1120	0.515	0.431	258.34	328.49		1120	0.543	0.461	300.67	586.71		1120	0.289	0.217	172.42	232.29
	1088	0.528	0.441	243.24	313.30		1088	0.550	0.467	289.67	558.20		1088	0.298	0.223	168.31	222.54
	1024	0.541	0.456	207.94	274.44		1024	0.558	0.479	242.08	456.63		1024	0.309	0.234	130.00	187.15
	800	0.566	0.484	136.28	194.07		800	0.578	0.502	165.44	330.88		800	0.350	0.264	93.66	142.29
	768	0.568	0.486	121.69	176.79		768	0.576	0.499	145.20	299.75		768	0.355	0.271	79.42	122.44
	736	0.565	0.486	115.71	166.19		736	0.576	0.501	138.73	289.04		736	0.355	0.271	76.76	119.29
	704	0.566	0.488	108.19	154.27		704	0.577	0.500	126.49	269.61		704	0.359	0.275	70.76	115.56
	672	0.565	0.486	101.20	140.95		672	0.572	0.497	122.07	251.10		672	0.365	0.276	69.95	108.68
	640	0.560	0.482	89.22	130.78		640	0.567	0.495	106.39	226.60		640	0.359	0.275	62.27	106.10
	608	0.559	0.480	81.15	119.65		608	0.564	0.491	97.60	213.40		608	0.365	0.282	56.48	100.13
	576	0.556	0.476	79.73	109.58		576	0.559	0.483	91.51	196.16		576	0.366	0.282	56.85	95.83
	544	0.550	0.474	72.64	101.82		544	0.549	0.478	84.88	185.73		544	0.363	0.281	51.30	91.13
	512	0.546	0.470	60.94	88.01		512	0.541	0.470	67.26	143.07		512	0.362	0.280	39.74	74.04
	480	0.542	0.466	55.58	81.56		480	0.534	0.463	62.78	137.89		480	0.357	0.277	37.10	71.34
	448	0.526	0.453	50.21	75.25		448	0.523	0.454	57.40	121.30		448	0.350	0.271	31.48	66.74
	416	0.519	0.442	45.77	68.81		416	0.510	0.441	53.95	110.52		416	0.341	0.267	29.70	64.50
	384	0.501	0.427	40.32	59.76		384	0.496	0.426	44.78	91.82		384	0.327	0.258	27.40	53.61
	352	0.485	0.411	35.00	56.77		352	0.473	0.407	39.41	85.04		352	0.318	0.244	27.21	53.20
	320	0.466	0.395	31.43	50.59		320	0.449	0.387	36.82	77.26		320	0.296	0.233	25.37	50.20
YOLOv5-M	1216	0.582	0.498	625.36	891.16	YOLOX-M	1216	0.604	0.520	783.30	1610.57	YOLOv3	1216	0.590	0.505	1443.86	2074.70
	1120	0.603	0.517	536.53	770.11		1120	0.617	0.537	677.11	1408.73		1120	0.610	0.527	1271.37	1768.54
	1088	0.609	0.527	519.83	737.88		1088	0.620	0.540	647.27	1336.64		1088	0.616	0.535	1223.61	1706.92
	1024	0.620	0.537	434.52	638.76		1024	0.631	0.552	529.96	1102.64		1024	0.628	0.547	961.28	1418.06
	800	0.637	0.565	296.93	454.14		800	0.638	0.565	367.77	769.53		800	0.652	0.575	692.87	1050.94
	768	0.638	0.564	254.89	399.01		768	0.640	0.566	311.45	685.07		768	0.653	0.578	564.81	897.09
	736	0.638	0.562	241.04	380.75		736	0.636	0.562	300.42	660.21		736	0.655	0.577	547.37	855.90
	704	0.634	0.564	226.07	350.43		704	0.636	0.564	277.56	612.36		704	0.654	0.579	510.27	809.51
	672	0.635	0.558	208.09	322.33		672	0.632	0.560	262.36	572.06		672	0.647	0.575	491.70	748.74
	640	0.633	0.558	187.36	299.98		640	0.631	0.560	231.61	515.86		640	0.648	0.575	433.27	691.91
	608	0.628	0.557	170.71	273.37		608	0.622	0.551	213.75	482.77		608	0.647	0.575	401.40	652.22
	576	0.625	0.551	163.97	251.48		576	0.618	0.545	199.27	438.36		576	0.644	0.569	379.21	609.58
	544	0.620	0.549	148.55	234.36		544	0.613	0.544	189.61	425.92		544	0.643	0.568	337.21	577.54
	512	0.613	0.541	116.56	194.47		512	0.607	0.537	138.69	319.07		512	0.639	0.567	245.62	452.50
	480	0.608	0.536	107.72	182.14		480	0.598	0.527	132.91	313.11		480	0.635	0.562	234.27	432.00
	448	0.601	0.527	101.30	166.05		448	0.585	0.517	125.29	280.31		448	0.629	0.554	216.46	397.36
	416	0.588	0.515	92.31	151.65		416	0.572	0.506	113.29	254.34		416	0.620	0.547	203.61	377.29
	384	0.581	0.507	80.01	128.25		384	0.557	0.489	95.59	210.45		384	0.608	0.536	166.89	311.50
	352	0.562	0.491	71.97	118.65		352	0.542	0.474	79.08	191.62		352	0.597	0.523	146.36	293.26
	320	0.544	0.474	63.56	104.77		320	0.521	0.456	71.98	175.17		320	0.584	0.507	132.46	260.28
YOLOv5-L	1216	0.610	0.531	1111.46	1633.32	YOLOX-L	1216	0.604	0.520	1422.16	2803.41						
	1120	0.632	0.550	965.79	1396.58		1120	0.616	0.536	1236.56	2613.85						
	1088	0.641	0.559	928.83	1347.54		1088	0.619	0.541	1189.40	2450.35						
	1024	0.648	0.569	750.04	1138.25		1024	0.630	0.550	957.30	1962.33						
	800	0.664	0.592	523.69	822.86		800	0.639	0.565	669.48	1375.42						
	768	0.668	0.596	442.63	713.26		768	0.640	0.567	556.73	1181.25						
	736	0.666	0.594	438.78	683.32		736	0.636	0.562	538.00	1159.05						
	704	0.667	0.597	407.85	631.36		704	0.635	0.562	499.17	1109.07						
	672	0.666	0.593	383.16	579.93		672	0.632	0.561	481.34	1036.40						
	640	0.661	0.593	344.15	536.17		640	0.631	0.558	418.18	924.12						
	608	0.661	0.593	317.15	501.15		608	0.624	0.554	386.83	879.69						
	576	0.660	0.589	285.75	460.62		576	0.620	0.548	360.60	812.27						
	544	0.657	0.587	270.14	432.25		544	0.614	0.543	346.94	775.81						
	512	0.646	0.577	193.81	355.21		512	0.608	0.535	243.87	566.45						
	480	0.643	0.572	184.23	338.36		480	0.599	0.526	237.37	556.09						
	448	0.633	0.563	169.39	311.20		448	0.585	0.517	219.17	497.40						
	416	0.626	0.554	162.03	288.41		416	0.574	0.506	203.10	469.88						
	384	0.614	0.544	140.32	235.94		384	0.558	0.490	168.02	398.59						
	352	0.596	0.524	113.53	217.30		352	0.539	0.472	145.11	368.69						
	320	0.581	0.510	103.47	191.71		320	0.522	0.458	129.75	328.61						

AGX Xavier offers the best FPS/Watt ratio, with the NVIDIA Jetson Nano being disadvantaged by the change in criteria. Based on cost, a device with an NVIDIA RTX 2080Ti is the most cost-effective.

4.7. Use of edge computing devices without hardware accelerator

One of the most widely used devices in the field of IoT (Internet of Things) is the Raspberry Pi v4. This device is priced at approximately 64 euros for the 4 GB memory version, and 85 euros for the 8 GB version. In terms of cost, it competes directly with the NVIDIA Jetson Nano. However, this device does not include GPU, nor any other hardware accelerator. Table 12 shows the inference times of the YOLOv5-N model on the 8 GB Raspberry Pi v4 using Pytorch (since it does not have an NVIDIA GPU, TensorRT cannot be used). As can be seen, for all input image sizes the inference time is over 1000 ms. The NVIDIA Jetson Nano, which is similarly priced, does not exceed 175 ms for any input size. This analysis demonstrates the importance of having a hardware accelerator, such as a GPU, for the application of object detection models based on deep learning. These inference times over 1000 ms may be feasible in scenarios where there are no major time constraints, but for cases where real-time (30 FPS) is needed, it may be excessive.

4.8. Use of cloud alternative

The use of cloud computing is a possible alternative to the use of edge computing devices. However, the main problem of this solution is the transfer time of the images to the computing center, and the transfer

Table 5
Metrics of all models by varying the input size over the Google Coral Dev Board.

Model	Size	AP50	AP	Inference time (ms)
YOLOv5-N	1216	0.345	0.270	3402.55
	1120	0.367	0.288	2888.34
	1088	0.379	0.298	2726.18
	1024	0.394	0.311	2421.79
	800	0.427	0.344	1469.61
	768	0.434	0.350	1358.18
	736	0.433	0.350	1246.06
	704	0.435	0.352	1134.66
	672	0.444	0.359	1034.63
	640	0.438	0.355	940.79
	608	0.438	0.354	847.69
	576	0.432	0.351	761.07
	544	0.430	0.350	679.96
	512	0.416	0.338	602.46
	480	0.415	0.337	526.83
	448	0.407	0.329	457.76
	416	0.392	0.319	395.29
	384	0.381	0.309	336.79
	352	0.367	0.297	281.95
	320	0.347	0.280	232.36
SSD-Mobilenet v1_TF1	300	0.274	0.218	10.14
SSD-Mobilenet v2_TF1	300	0.309	0.257	13.99
SSD-Mobilenet v2_TF2	300	0.283	0.229	19.66
EfficientDet-Lite 1_TF2	384	0.467	0.399	202.63

Table 6
Comparison in YOLOv5-M and YOLOv8-M with Pytorch.

Model	Millions of parameters	Size	AP_{50}	AP	Inference time (ms)
YOLOv5-M	21.2	768	0.638	0.564	8.69
		608	0.628	0.557	7.16
		448	0.601	0.527	5.93
		320	0.544	0.474	5.56
YOLOv8-M	25.9	768	0.790	0.611	7.55
		608	0.780	0.607	6.20
		448	0.752	0.582	5.19
		320	0.695	0.542	5.36

Table 7
Actual consumption of tested devices.

Device	Watts
NVIDIA RTX 2080Ti	258.0
NVIDIA Jetson AGX Xavier	19.0
NVIDIA Jetson Nano	7.0
Google Coral Dev Board	4.5

time of the results to the system in charge of performing the appropriate actions. In [44], it is shown how network times considerably influence the final processing time. In addition, another key factor is cost. If the NC6 instance of Azure is selected (cheapest option with GPU), whose cost per hour is €0.98, and assuming constant processing services are needed, the monthly cost of the cloud service will be €705.60. Given that the cost of an NVIDIA Jetson Nano is between €50 and €100, and that of an NVIDIA Jetson AGX Xavier is approximately €650, the use of cloud alternatives may be too expensive. Another cheaper alternative is the use of a CaaS (Container as a Service) with a CPU. This option costs €0.11 per hour, so the monthly cost would be €79.20. Even though it is a cheaper choice, its monthly cost may not compensate its use.

The choice of a cloud alternative may be interesting if constant processing is not required, in which case it would be convenient to turn off the services when they are not needed, or when the processing needs are not stable over time.

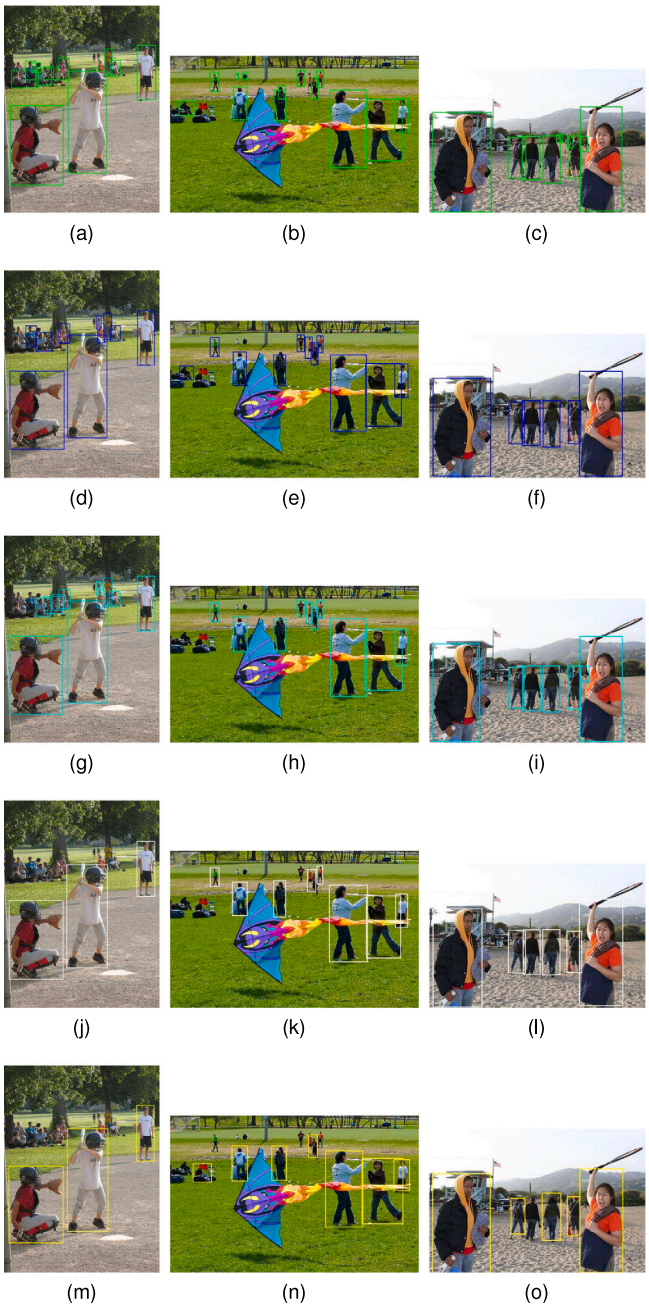

Fig. 5. Detections of persons carried out with various models. (a–c) Ground truth. (d–f) YOLOv5-X at 704. (g–i) YOLOv5-L at 512. (j–l) YOLOv5-S at 320. (m–o) SSD-Mobilenet v2_TF1 at 300.

5. Conclusion

In this paper the performance of object detection models based on deep learning on edge computing devices are quantitatively compared.

Throughout the paper, the results of evaluating YOLOv3, YOLOv5 and YOLOX, with all their variants, on a server with a NVIDIA RTX 2080 Ti, an NVIDIA Jetson AGX Xavier, an NVIDIA Jetson Nano and a Google Coral Dev Board are shown. The results are measured both in terms of inference speed and Average Precision (AP) using the Pytorch, TensorRT and Tensorflow Lite frameworks.

After analyzing the results, the following conclusions are reached:

- Inference is faster with TensorRT than with Pytorch (batch size of 1). The more complex the model and/or the more limited the

Table 8
Higher AP models processing images in real-time (30 FPS).

Device	Model	Size	Inference time (ms)	AP_{50}	AP
NVIDIA RTX 2080 Ti	YOLOv5-X	704	7.44	0.683	0.613
NVIDIA Jetson AGX Xavier	YOLOv5-L	512	29.28	0.646	0.577
NVIDIA Jetson Nano	YOLOv5-S	320	31.43	0.466	0.395
Google Coral Dev Board	SSD-Mobilenet v2_TF1	300	13.99	0.309	0.257

Table 9
Higher AP (Person class) models processing images in real-time (30 FPS).

Device	Model	Size	Inference time (ms)	AP_{50}	AP
NVIDIA RTX 2080 Ti	YOLOv5-X	704	7.44	0.844	0.610
NVIDIA Jetson AGX Xavier	YOLOv5-L	512	29.28	0.814	0.571
NVIDIA Jetson Nano	YOLOv5-S	320	31.43	0.661	0.399
Google Coral Dev Board	SSD-Mobilenet v2_TF1	300	13.99	0.464	0.247

Table 10
Analysis of the FPS/Watt and FPS/€ ratios of the fastest models with a minimum AP_{50} of 0.4.

Device	Model	Size	FPS	AP_{50}	FPS/Watio	FPS/€
NVIDIA RTX 2080 Ti	YOLOv5-N	416	952.4	0.413	3.69	0.81
NVIDIA Jetson AGX Xavier	YOLOX-Tiny	320	107.9	0.419	5.68	0.16
NVIDIA Jetson Nano	YOLOv5-N	416	34.8	0.413	4.97	0.37
Google Coral Dev Board	EfficientDet-Lite 1_TF2	384	4.9	0.467	1.10	0.03

Table 11
Analysis of the FPS/Watt and FPS/€ ratios of the fastest models with a minimum AP_{50} of 0.65.

Device	Model	Size	FPS	AP_{50}	FPS/Watio	FPS/€
NVIDIA RTX 2080 Ti	YOLOv5-L	544	233.1	0.657	0.90	0.20
NVIDIA Jetson AGX Xavier	YOLOv5-L	544	26.0	0.657	1.37	0.04
NVIDIA Jetson Nano	YOLOv5-L	544	3.7	0.657	0.53	0.04

Table 12
YOLOv5-N metrics varying the input size over the Raspberry Pi v4 8 GB.

Model	Size	AP_{50}	AP	Inference time (ms)
YOLOv5-N	1216	0.368	0.297	19 995.60
	1120	0.394	0.318	16 142.07
	1088	0.401	0.325	14 973.07
	1024	0.415	0.338	13 335.00
	800	0.445	0.368	7 344.73
	768	0.456	0.378	6 674.43
	736	0.457	0.379	6 174.60
	704	0.457	0.380	5 584.57
	672	0.459	0.380	5 004.10
	640	0.458	0.381	4 566.80
	608	0.455	0.379	4 051.70
	576	0.452	0.376	3 520.87
	544	0.448	0.374	3 149.07
	512	0.439	0.365	2 772.37
	480	0.433	0.360	2 418.90
	448	0.423	0.351	2 112.93
	416	0.413	0.343	1 841.77
	384	0.397	0.329	1 526.20
	352	0.379	0.313	1 381.90
	320	0.363	0.298	1 150.20

device where the model is evaluated, the more this difference is accentuated.

- There are no differences in terms of AP between TensorRT and Pytorch. The same is not true for Tensorflow Lite. This is because when converting the model to TFLite, it is necessary to use 8FP instead of 16FP.
- The NVIDIA RTX 2080 Ti can run all models for all input sizes in real-time using TensorRT.
- The NVIDIA Jetson AGX Xavier is a high-end edge computing device that can run a variety of models in real-time under different input sizes.
- The NVIDIA Jetson Nano is a lower-end edge computing device that can run a limited number of models in real-time.
- The Google Coral Dev Board is a device with more software limitations than NVIDIA's devices. Since it has TPU, the best option is to use TFLite as a framework. Model conversion between frameworks is not quite optimal, as many of the operations performed by the models cannot be performed on the TPU and therefore must be performed on the CPU, slowing down the inference process.
- The computational power of the devices is closely related to their power consumption. The NVIDIA RTX 2080 Ti has the highest power consumption, while the Google Coral Dev Board has the lowest.

A server with a GPU such as the NVIDIA RTX 2080 Ti is not feasible in scenarios where mobility is required or where many devices are needed, since its cost is very high. The use of edge computing devices may be the solution to many of these requirements. The NVIDIA Jetson AGX Xavier performs well, making it possible to process images with high precision in real time (30 FPS). In the case of power consumption restrictions, the NVIDIA Jetson Nano or the Google Coral Dev Board are good options as they have very low power consumption.

This paper compares different edge computing devices for real-time image processing using deep learning-based object detection models. The choice of the appropriate device depends on the usage scenario, since for certain applications temporal or accuracy constraints can be decisive. In a scenario where there are no major restrictions, low-cost devices can be chosen. However, in scenarios where there are strong restrictions it may be necessary to use higher performance devices, and therefore more expensive.

CRediT authorship contribution statement

Darío G. Lema: Conceptualization, Data curation, Formal analysis, Methodology, Investigation, Visualization, Writing – original draft, Writing – review & editing. **Rubén Usamentiaga:** Funding acquisition, Investigation, Project administration, Resources, Software, Supervision, Validation. **Daniel F. García:** Project administration, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Dario G. Lema reports was provided by University of Oviedo.

Data availability

Data is in the manuscript.

Acknowledgments

This work has been partially funded by the project RTI2018-094849-B-I00 of the Spanish National Plan for Research, Development and Innovation.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Handwritten digit recognition with a back-propagation network, in: D. Touretzky (Ed.), Advances in Neural Information Processing Systems, Vol. 2, Morgan-Kaufmann, 1989, URL https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

[3] G. Gilman, R.J. Walls, Characterizing concurrency mechanisms for NVIDIA GPUs under deep learning workloads, Perform. Eval. 151 (2021) 102234, http://dx.doi.org/10.1016/j.peva.2021.102234, URL https://www.sciencedirect.com/science/article/pii/S0166531621000511.

[4] D.-S. Hong, H.-H. Chen, P.-Y. Hsiao, L.-C. Fu, S.-M. Siao, CrossFusion net: Deep 3D object detection based on RGB images and point clouds in autonomous driving, Image Vis. Comput. 100 (2020) 103955, http://dx.doi.org/10.1016/j.imavis.2020.103955, URL https://www.sciencedirect.com/science/article/pii/S0262885620300871.

[5] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-view 3D object detection network for autonomous driving, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1907–1915.

[6] A. Baba, T. Bonny, FPGA-based parallel implementation to classify hyperspectral images by using a convolutional neural network, Integration 92 (2023) 15–23, http://dx.doi.org/10.1016/j.vlsi.2023.04.003, URL https://www.sciencedirect.com/science/article/pii/S0167926023000597.

[7] D.G. Lema, O.D. Pedrayes, R. Usamentiaga, P. Venegas, D.F. García, Automated detection of subsurface defects using active thermography and deep learning object detectors, IEEE Trans. Instrum. Meas. 71 (2022) 1–13, http://dx.doi.org/10.1109/TIM.2022.3169484.

[8] R. Usamentiaga, D.G. Lema, O.D. Pedrayes, G. Daniel, Automated surface defect detection in metals: a comparative review of object detection and semantic segmentation using deep learning, IEEE Trans. Ind. Appl. (2022) 1, http://dx.doi.org/10.1109/TIA.2022.3151560.

[9] R. Elakkiya, V. Subramaniyaswamy, V. Vijayakumar, A. Mahanti, Cervical cancer diagnostics healthcare system using hybrid object detection adversarial networks, IEEE J. Biomed. Health Inf. 26 (4) (2022) 1464–1471, http://dx.doi.org/10.1109/JBHI.2021.3094311.

[10] L. Lecrosnier, R. Khemmar, N. Ragot, B. Decoux, R. Rossi, N. Kefi, J.-Y. Ertaud, Deep learning-based object detection, localisation and tracking for smart wheelchair healthcare mobility, Int. J. Environ. Res. Public Health 18 (1) (2021) URL https://www.mdpi.com/1660-4601/18/1/91.

[11] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 6 (02) (1998) 107–116.

[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[13] L. Qian, Z. Luo, Y. Du, L. Guo, Cloud computing: An overview, in: M.G. Jaatun, G. Zhao, C. Rong (Eds.), Cloud Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 626–631.

[14] K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research, IEEE Access 8 (2020) 85714–85728, http://dx.doi.org/10.1109/ACCESS.2020.2991734.

[15] J. Chen, X. Ran, Deep learning with edge computing: A review, Proc. IEEE 107 (8) (2019) 1655–1674, http://dx.doi.org/10.1109/JPROC.2019.2921977.

[16] A.A. Süzen, B. Duman, B. Şen, Benchmark analysis of jetson TX2, jetson nano and raspberry PI using deep-CNN, in: 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2020, pp. 1–5, http://dx.doi.org/10.1109/HORA49412.2020.9152915.

[17] A.S. Pinto de Aguiar, F.B. Neves dos Santos, L.C. Feliz dos Santos, V.M. de Jesus Filipe, A.J. Miranda de Sousa, Vineyard trunk detection using deep learning – an experimental device benchmark, Comput. Electron. Agric. 175 (2020) 105535, http://dx.doi.org/10.1016/j.compag.2020.105535, URL https://www.sciencedirect.com/science/article/pii/S0168169920304555.

[18] S. Arabi, A. Haghighat, A. Sharma, A deep-learning-based computer vision solution for construction vehicle detection, Comput.-Aided Civ. Infrastruct. Eng. 35 (7) (2020) 753–767, http://dx.doi.org/10.1111/mice.12530, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12530. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12530.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: Single shot MultiBox detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 21–37.

[20] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.

[21] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338, http://dx.doi.org/10.1007/s11263-009-0275-4.

[22] NVIDIA, Jetson nano, 2020, URL https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Online] Accessed on 18 May 2022.

[23] NVIDIA, Jetson AGX xavier, 2020, URL https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit. [Online] Accessed on 18 May 2022.

[24] Google, Coral dev board, 2018, URL https://coral.ai/products/dev-board/#tech-specs. [Online] Accessed on 18 May 2022.

[25] A. Gagliardi, F. de Gioia, S. Saponara, A real-time video smoke detection algorithm based on Kalman filter and CNN, J. Real-Time Image Process. 18 (6) (2021) 2085–2095, http://dx.doi.org/10.1007/s11554-021-01094-y.

[26] S. Cass, Taking AI to the edge: Google's TPU now comes in a maker-friendly package, IEEE Spectr. 56 (5) (2019) 16–17, http://dx.doi.org/10.1109/MSPEC.2019.8701189.

[27] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[28] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015.

[29] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 28, Curran Associates, Inc., 2015, URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[30] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[31] J. Redmon, A. Farhadi, YOLO9000: Better, faster, stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[32] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.

[33] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, 2020, arXiv preprint arXiv:2004.10934.

[34] G. Jocher, Ultralytics/yolov5: v6.1 - TensorRT, TensorFlow edge TPU and openvino export and inference, 2022, http://dx.doi.org/10.5281/zenodo.6222936.

[35] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, YOLOX: Exceeding YOLO series in 2021, 2021, arXiv preprint arXiv:2107.08430.

[36] Z. Ge, S. Liu, Z. Li, O. Yoshie, J. Sun, OTA: Optimal transport assignment for object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 303–312.

[37] G. Jocher, A. Chaurasia, J. Qiu, YOLO by Ultralytics, 2023, URL https://github.com/ultralytics/ultralytics.

[38] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: Single shot MultiBox detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 21–37.

[39] Y.-C. Chiu, C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, T.-T. Lee, Mobilenet-SSDv2: An improved object detection model for embedded systems, in: 2020 International Conference on System Science and Engineering (ICSSE), 2020, pp. 1–5, http://dx.doi.org/10.1109/ICSSE50014.2020.9219319.

[40] M. Tan, R. Pang, Q.V. Le, EfficientDet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[41] R. Padilla, W.L. Passos, T.L.B. Dias, S.L. Netto, E.A.B. da Silva, A comparative analysis of object detection metrics with a companion open-source toolkit, Electronics 10 (3) (2021) http://dx.doi.org/10.3390/electronics10030279, URL https://www.mdpi.com/2079-9292/10/3/279.

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.

[43] G. Jocher, Ultralytics/yolov3, 2022, URL https://github.com/ultralytics/yolov3.

[44] D.G. Lema, O.D. Pedrayes, R. Usamentiaga, D.F. García, Á. Alonso, Cost-performance evaluation of a recognition service of livestock activity using aerial images, Remote Sens. 13 (12) (2021) http://dx.doi.org/10.3390/rs13122318, URL https://www.mdpi.com/2072-4292/13/12/2318.