Benchmarking Object Detection Deep Learning Models on Edge Devices

Daghash K. Alqahtani
daghash.alqahtani@student.unimelb.edu.au
University of Melbourne
Melbourne, Victoria, Australia

Aamir Cheema
Aamir.Cheema@monash.edu
Monash University
Melbourne, Victoria, Australia

Adel N. Toosi
adel.toosi@unimelb.edu
University of Melbourne
Melbourne, Victoria, Australia

ABSTRACT

Many modern applications, such as autonomous vehicles, require deploying deep learning network algorithms on resource-constrained edge computing devices to enable real-time image and video processing and decision-making. However, there is a lack of comprehensive understanding regarding the efficiency and performance of various object detection models on the wide range of edge devices available on the market. In this paper, we conduct a thorough evaluation of state-of-the-art object detection models, including YOLOv8 (Nano, Small, Medium variants), EfficientDet Lite (Lite0, Lite1, Lite2), and SSD (SSD MobileNet V1, SSDLite MobileDet). We deployed these models on the popular edge devices such as the Raspberry Pi 3, 4, and 5 with/without TPU accelerators, and the Jetson Orin Nano and collected key performance metrics, including energy consumption, inference time, and Mean Average Precision (mAP), to gain insights into the operational efficiency of these models across different hardware architectures. Our findings highlight that lower mAP models such as SSD MobileNet V1 are more energy-efficient and faster in inference, whereas higher mAP models like YOLOv8 Medium generally consume more energy and have slower inference, though with exceptions when accelerators like TPUs are used. Among the edge devices, the Jetson Orin Nano stands out as the fastest and most energy-efficient option. These results emphasize the need to balance accuracy, speed, and energy efficiency when deploying deep learning models on edge devices, offering valuable guidance for practitioners and researchers selecting models and devices for their applications.

CCS CONCEPTS

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

KEYWORDS

Deep Learning, Object Detection Models, Inference Time, Energy Efficiency, Precision, Edge

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Daghash K. Alqahtani, Aamir Cheema, and Adel N. Toosi. 2018. Benchmarking Object Detection Deep Learning Models on Edge Devices. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Object detection is a pivotal technology in the field of computer vision, empowering machines to identify and accurately locate objects within visual inputs such as images or videos. This capability renders the technology an invaluable asset for organizations seeking to integrate innovative solutions to optimize and automate their business processes. Advancements in machine learning and artificial intelligence have substantially expanded the potential applications of this technology. Software systems that leverage object detection technology serve as valuable and efficient tools for addressing a wide range of tasks. Object detection is utilized in numerous real-world systems, including autonomous vehicles, surveillance systems, retail, healthcare, agriculture, manufacturing, sports analytics, environmental monitoring, and smart cities.

For instance, object detection is a fundamental technology that underpins the concept of autonomous transportation. The precise recognition of objects, including pedestrians, obstacles, and other vehicles, is essential for ensuring the safe operation of self-driving vehicles on public roads. In autonomous vehicles, object detectors identify the vehicle's position within its environment, consider the surrounding context, and track other objects to facilitate route planning. If the vehicle required rapid stopping or turning to avert an accident, transmitting data between the vehicle and cloud for processing would be too slow. Edge computing brings cloud computing capabilities directly to the vehicle, enabling the onboard IoT sensors to locally process data in real-time and respond accordingly to avoid a collision. These capabilities highlight the significant value that object detection offers in enhancing the performance and safety of automated systems, which must accurately understand and respond to their surroundings in a timely manner [1, 19].

The field of object detection is currently characterized by promising opportunities, fueled by continuous technological progress. A prominent trend involves the ongoing refinement of detection algorithms, which aim to more effectively navigate complex environments while simultaneously enhancing both accuracy and speed. Moreover, the emergence of edge computing presents a viable pathway forward, enabling real-time object detection on a variety of edge devices, such as smartphones, drones, and Internet of Things (IoT) systems. This approach reduces latency and decreases the reliance on cloud-based infrastructure. Despite these advancements, significant challenges persist in developing and deploying robust

and reliable object detection systems in the edge computing domain. The constrained resources of edge devices present obstacles when deploying deep learning object detection models, making it a non-trivial task. Energy consumption is a critical factor when executing these models on edge devices, as they vary considerably in energy requirements and inference time. Researchers and industries have developed a diverse range of object detection models to improve accuracy and processing speed. Therefore, the challenges in this area include the selection of appropriate models and the identification of edge devices that can adequately meet the system's needs.

In this paper, we aim to evaluate the performance of most popular deep learning object detection models across prominent edge devices, collecting key metrics such as energy consumption, inference time, and accuracy. Additionally, we provide insights for deploying these models on the investigated edge devices. Our **key contributions** can be summarized as follows:

- We developed object detection applications for processing images as a web service using Flask-API.
- We utilized different frameworks, including PyTorch, TensorFlow Lite, and TensorRT, to deploy our deep learning web service models on the edge devices including Raspberry Pi series, Edge TPU, and Jetson Orin Nano.
- We employed the FiftyOne tool to evaluate the accuracy of the object detection models and collected the mean Average Precision using COCO datasets for each model on each device.
- We conducted automated comprehensive performance measurment tests using the Locust tool and reported the performance of YOLOv8, SSD, and EfficientDet Lite models with different versions on various edge devices.

The remainder of this paper is organized as follows. Section 2 provides an overview of the edge computing and object detection deep learning architectures. Section 3 outlines the performance evaluation, including the evaluation metrics, experimental setup, and the results of our experiments. Section 4 presents the related work. Finally, Section 5 concludes the paper and suggests future research directions.

2 EDGE DEVICES, FRAMEWORKS AND MODEL FORMATS

2.1 Edge Devices

2.1.1 Raspberry Pi. The Raspberry Pi is a line of single-board computers produced by the Raspberry Pi Foundation [8]. When connected to accessories like a keyboard, mouse, and monitor, the Raspberry Pi becomes a low-cost personal computer. It is widely used for robotics, Internet of Things applications, and real-time image and video processing. The latest Raspberry Pi models include the Raspberry Pi 3 Model B+ [15], Raspberry Pi 4, and Raspberry Pi 5. The Raspberry Pi 4 maintains compatibility with the previous Raspberry Pi 3 Model B+ while offering improvements in processor speed, multimedia capabilities, memory capacity, and connectivity [16]. The Raspberry Pi 5 is the newest model, featuring significant enhancements in CPU and GPU performance as well as increased memory capacity and I/O bandwidth compared to the Raspberry Pi 4. This latest iteration also introduces the integration of Raspberry Pi silicon into a flagship device [17].

2.1.2 TPU Accelerator. The Coral USB Accelerator is a USB device that functions as an Edge TPU co-processor for computational devices. It features a USB-C port, allowing it to connect to a host computer and accelerate machine learning inference tasks. The Edge TPU, an Application-Specific Integrated Circuit developed by Google, is designed to execute machine learning models on edge computing devices like the Raspberry Pi [6].

2.1.3 NVIDIA Jetson. The NVIDIA Jetson Orin Series represents the most recent developer board series unveiled by NVIDIA Jetson Official. It facilitates the realization of cutting-edge products through the deployment of the world's most potent AI computing systems tailored for energy-efficient autonomous machinery. With NVIDIA Jetson Orin modules boasting up to 275 trillion operations per second (TOPS) and exhibiting an 8X performance enhancement over the previous generation, the platform enables the execution of numerous concurrent AI inference pipelines. Moreover, its robust support for high-speed interfaces accommodating multiple sensors positions it as the quintessential solution for ushering in a new era of robotics. The Orin Nano stands as the introductory tier in the Jetson Orin series, tailored for scenarios emphasizing diminished power consumption and cost considerations while retaining the necessity for AI processing prowess. It finds applicability in edge AI apparatus, Internet of Things (IoT) devices, and various embedded systems wherein spatial constraints, power efficiency, and cost-efficiency hold paramount significance [18].

2.2 Object Detection Deep Learning Architectures

Object detection refers to a computer vision methodology aimed at identifying object instances within images or videos. These algorithms usually rely on machine learning or deep learning methodologies to yield significant outcomes. Much like humans swiftly identify and pinpoint objects in visual content, the objective of object detection is to emulate this cognitive ability through computational means.

Object detection utilizing deep learning distinguishes itself from other methodologies through the application of convolutional neural networks (CNNs). These networks replicate the intricate neural structures of the human brain and are composed of an input layer, multiple hidden layers, and an output layer. The training of these neural networks can be categorized as supervised, semi-supervised, or unsupervised, depending on the extent to which the data is labeled. CNN-based deep neural networks for object detection achieve the highest levels of speed and accuracy in detecting single or multiple objects. This is attributed to CNNs' ability to perform automated learning with minimal manual feature engineering. While deep learning and convolutional neural networks (CNNs) encompass a vast array of topics, this section will concentrate solely on the key aspects related to object detection models and frameworks.

Early deep learning-based object detection models are classified into two categories: one-stage and two-stage detectors. One-stage

Feature	Pi3 Model B+	Pi4	Pi5	Coral USB Accelerator	Jetson Orin Nano
CPU	Arm Cortex-A53	Arm Cortex-A72	Arm Cortex-A76	Edge TPU ML accelerator	Arm Cortex-A76
CPU speed	1.4GHz	1.8GHz	2.4GHz	N/A	2.4GHz
GPU	VideoCore IV	VideoCore IV	VideoCore VII	Edge TPU	NVIDIA Ampere with 1024 CUDA cores and 32 Tensor cores
RAM	1GB	1GB, 2GB, 4GB or 8GB	4GB and 8GB	N/A	4GB and 8GB
Connectivity	Wi-Fi, Bluetooth, Gigabit Ethernet	Wi-Fi, Bluetooth, Gigabit Ethernet	Wi-Fi, Bluetooth, Gigabit Ethernet	N/A	Wi-Fi, Bluetooth, Gigabit Ethernet
Power	5V/2.5A	5V/3A	5V/5A	powered via USB	5V/5A
Release Year	2018	2019	2023	2020	2023
Price	$60	$60-&120	$100-$130	160	$800-$1000

Table 1: Edge Devices Comparison

detectors constitute a class of object detection algorithms that simultaneously predict both the bounding box and the object's category in a single forward pass through the network. This concurrent processing enables one-stage detectors to identify the presence of objects and delineate their bounding boxes more efficiently and faster than two-stage detectors. Such capabilities make one-stage detectors particularly suitable for real-time detection applications. Notable examples of one-stage object detection models include YOLO, SSD, and EfficientDet, each of which will be introduced separately.

In the remaining part of this section we provide overview of the popular deep learning based object detection models and the frameworks that can be utilized to deploy them on our edge devices.

2.2.1 You Only Look Once (YOLO). The YOLO algorithm, introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, marks a significant advancement in real-time object detection [21]. Unlike conventional methods that apply a classifier to multiple regions within an image, YOLO models conceptualize detection as a single regression problem, predicting bounding boxes and class probabilities from entire images in a single evaluation. This holistic approach markedly improves detection speed, rendering YOLO particularly suitable for applications demanding real-time processing. The original YOLO algorithm has undergone several iterations, with YOLOv2, YOLOv3, YOLOv4 and the latest YOLOv8 each introducing enhancements in accuracy, speed, and the capability to detect a broader range of object sizes.

2.2.2 Single shot multibox detector (SSD). The SSD algorithm predicts multiple bounding boxes and their corresponding class scores in a single pass. It leverages multiple feature maps from different layers of the network to perform detections at various scales. The base network, commonly a VGG16 pre-trained on ImageNet, is employed for feature extraction and is truncated before the fully connected layers, enabling the model to handle inputs of any size. A key innovation of SSD is the use of feature maps from various network layers to predict detections at multiple scales, effectively addressing the challenge of detecting objects of different sizes. Specifically, smaller feature maps are utilized to detect larger objects, while larger feature maps are used to detect smaller objects [14].

2.2.3 EfficientDet. An advanced object detection model developed by Google's Brain Team, is renowned for its efficiency and scalability. Built upon the highly efficient EfficientNet architecture, EfficientDet scales the network's depth, width, and resolution through a compound scaling method. This approach enhances performance while reducing computational cost. Additionally, the model integrates a novel feature network and a weighted bi-directional feature pyramid network, which improve feature fusion and cross-scale connections. EfficientDet offers a robust solution for object detection tasks, achieving superior results with fewer parameters and FLOPs. The EfficientDet model includes seven variants, designated as EfficientDet0 through EfficientDet6, as detailed in the EfficientDet research paper [22].

The deep learning frameworks play a crucial role in the development and deployment of object detection models by providing a comprehensive suite of tools and functionalities. These frameworks streamline the process of building, training, and implementing object detection systems by offering pre-built models, data augmentation techniques, and utilities for various stages of model development. Examples of such frameworks include TensorFlow Lite [23], PyTorch [20], TensoRT [7], among others. They enable researchers and developers to efficiently manage the complexities of object detection tasks, from data preprocessing and model training to evaluation and deployment.

3 PERFORMANCE EVALUATION

3.1 Metrics

In this study, we evaluate the performance of three object detection models—*YOLOv8*, *EfficientDet Lite*, and *SSD*—on multiple edge devices, including Raspberry *Pi 3*, *4*, *5*, *Pi 3 with TPU*, *Pi 4 with TPU*, *Pi 5 with TPU*, and *Jetson Orin Nano*. The evaluation focuses on three key performance metrics: *inference time*, *energy consumption*, and *mean Average Precision (mAP)*.

3.1.1 Inference Time: This metric measures the time taken by each model from when it receives the input image until it produces the detection results, excluding any pre-processing or post-processing steps. This inference time is crucial for applications requiring real-time object detection. We report the inference time in milliseconds for each model on each device, with the average calculated over a series of test images to ensure consistent and reliable measurements.

3.1.2 Energy Consumption: This metric evaluates the energy efficiency of each model when deployed on different edge devices. First, we measure the base energy consumption (BE) for five minutes of each device without running any computations. This base energy consumption provides insights into the inherent energy efficiency of the edge devices. Next, we measure the total energy consumption (TE) for five minutes as the same duration of base energy consumption while running an object detection model on the device. To determine the energy consumption excluding the base energy usage (EexcR), we subtract the base energy consumption from the total energy consumption. This difference is then divided by the number of requests (NR) processed to obtain the

energy consumption per request excluding the base energy usage. The formula used is as follows:

$$\text{EexcR} = \frac{\text{TE} - \text{BE}}{\text{NR}}$$

This metric is reported in milliwatt-hours (mWh) and is critical for applications where power efficiency is a concern, such as battery-powered devices.

3.1.3 Model Evaluation Using COCO Dataset: To determine the capabilities and accuracy of the YOLOv8, EfficientDet Lite, and SSD models, which have been trained on the COCO dataset, we utilize the COCO validation dataset consisting of 5,000 images. The FiftyOne tool [25] , an open-source tool, facilitates the visualization and access to COCO data resources and serves as an evaluation tool for model analysis on COCO [13]. It calculates the accuracy of a deep learning model by comparing the detected objects from the Detect Objects Using Deep Learning tool to ground reference data. The accuracy of a model is evaluated using four metrics: *Precision*, *Recall*, *F1 score*, and COCO mean Average Precision (mAP). Here is a clarification of each metric:

- *Precision*: The ratio of the number of true positives to the total number of positive predictions.
- *Recall*: The ratio of the number of true positives to the total number of actual (relevant) objects.
- *F1 Score*: The F1 score is a weighted average of precision and recall. Values range from 0 to 1, where 1 indicates the highest accuracy.
- *Average Precision (AP)*: AP is the precision averaged across all recall values between 0 and 1.
- *Mean Average Precision (mAP)*: mAP is the average AP over multiple Intersection over Union (IoU) thresholds.

Each model is tested for a duration of five minutes per edge device, ensuring that the results are representative of sustained usage rather than short bursts of activity. This comprehensive evaluation allows us to understand the trade-offs between speed, accuracy, and energy efficiency for each model on different edge devices, providing valuable insights for selecting the optimal model and device combination for specific applications.

3.2 Experimental Setup

In this study, we evaluated the performance of object detection models on various edge devices. The evaluation focused on three primary metrics: inference time, energy consumption, and accuracy.

3.2.1 Hardware and Device Setup. The hardware used in our experimental setup consisted of various edge devices with different specifications as table 2 shows that to evaluate the performance of object detection models. The Raspberry Pi series included the Raspberry Pi 3 Model B+ with 1 GB of LPDDR2 RAM, the Raspberry Pi 4 Model B with 4 GB of LPDDR4-3200 SDRAM, and the Raspberry Pi 5 with 4 GB of LPDDR4x RAM. These devices were chosen due to their popularity and affordability, making them accessible options for many applications. To leverage the capabilities of Tensor Processing Units (TPUs), we utilized the Raspberry Pi 3 Model B+, Pi 4 Model B, and Pi 5 equipped with Google Coral USB Accelerators, enhancing their computational power for deep learning tasks. Lastly, the NVIDIA Jetson Orin Nano with 4 GB of RAM and an integrated GPU was included to provide a high-performance comparison. This device was selected because it differs from the Raspberry Pi models by offering a powerful GPU, allowing us to evaluate the performance differences between devices with CPUs, TPUs, and GPUs. Additionally, we used a USB power meter specifically the UM25C model with Bluetooth connectivity to measure the energy consumption of the edge devices. These edge devices offered a diverse range of processing capabilities and memory configurations, making them ideal for assessing the efficiency and effectiveness of various object detection models in resource-constrained environments.

Edge Device	RAM
Raspberry Pi 3 Model B+	1 GB
Raspberry Pi 4 Model B	4 GB
Raspberry Pi 5	4 GB
Pi 3 Model B+ with TPU	1 GB
Pi 4 Model B with TPU	4 GB
Pi 5 with TPU	4 GB
Orin Nano	4 GB

Table 2: Edge Devices Specifications

3.2.2 Software and Frameworks. In our experimental setup, we utilize various software frameworks and tools to deploy and run object detection models on different edge devices as Table 3 displays that. The choice of software and frameworks is influenced by the need to optimize performance for each specific device, including those with CPUs, TPUs, and GPUs. We use PyTorch to deploy and run YOLOv8 models on the Raspberry Pi 3, Raspberry Pi 4, and Raspberry Pi 5. To leverage the capabilities of TPUs, the YOLOv8 models are converted from PyTorch to TensorFlow Lite (TFLite) format and compiled to run on TPUs in the Raspberry Pi 3, Raspberry Pi 4, and Raspberry Pi 5. For deployment on the NVIDIA Jetson Orin Nano, the YOLOv8 models are converted to TensorRT format to optimize for the GPU. EfficientDet Lite and SSD models are initially in TFLite format and are deployed on the Raspberry Pi 3, Raspberry Pi 4, and Raspberry Pi 5. These models are compiled to run on TPUs in the respective Raspberry Pi devices. For the NVIDIA Jetson Orin Nano, EfficientDet Lite and SSD models are converted from TFLite to TensorRT format for optimized performance on the GPU. The operating systems used are Raspberry Pi OS (Bookwork - 64 bit) for the Raspberry Pi devices and Jetson Linux (Ubuntu-based) for the NVIDIA Jetson Orin Nano. Additional libraries and tools, such as OpenCV for image processing and FiftyOne for model evaluation, are utilized to facilitate the experiments. To detect objects in an image, we write Python code and utilize the Flask-RESTful library to run this code as a service with an API URL. This approach allow us to deploy the object detection functionality as a web service, enabling easy integration and testing across different edge devices.

3.2.3 Experimental Procedure. The detailed procedure for evaluating the object detection models involved several steps. We developed custom Python scripts to deploy the object detection models on the edge devices, with each script representing a different model (YOLOv8, EfficientDet Lite, SSD). These scripts were deployed on the respective edge devices, and the Flask-RESTful library was utilized to run the Python code as a service, enabling users or clients to post images to an API endpoint. The service processed the images,

Edge Device	YOLOv8 Framework	EfficientDet Framework	SSD Framework
Raspberry Pi 3 Model B+	PyTorch	TFLite	TFLite
Raspberry Pi 4 Model B	PyTorch	TFLite	TFLite
Raspberry Pi 5	PyTorch	TFLite	TFLite
Pi 3 Model B+ with TPU	TFLite	TFLite	TFLite
Pi 4 Model B with TPU	TFLite	TFLite	TFLite
Pi 5 with TPU	TFLite	TFLite	TFLite
Orin Nano	TensorRT	TensorRT	TensorRT

Table 3: Edge Devices, Models and Frameworks

invoked the model, and returned the output results in JSON format, including the detected objects and the inference time.

For measuring inference time, we recorded the time taken by the model to detect objects in an image for each edge device. The inference time was included in the response returned by the API service. To ensure accurate measurement, we employed the Locust tool to send multiple requests back-to-back for a duration of five minutes. The average inference time was calculated from the responses received during this period.

Energy consumption was measured using a USB power meter, which was set up between the power cable and the edge device. This meter, equipped with Bluetooth connectivity, enabled real-time energy reading. The base energy consumption (BE) was measured by running the energy reader Python code on another Raspberry Pi 4 device (referred to as the agent device) for five minutes without any computational load. During the inference time measurement with Locust, the Python code also connected to the USB power meter to read the total energy consumption (TE) in milliwatt-hours (mWh) for the duration of the experiment. The energy consumption per request (EexcR) was then calculated by subtracting the base energy consumption from the total energy consumption and dividing the result by the number of requests (NR).

To automate the testing process for each device as Figure 1 presents, we wrote a bash script on the agent device. This script started by running the object detection service (model) on the edge device, followed by executing the Locust file. Upon completion of the Locust tests, the script terminated the service and proceeded to the next model. This automation ensured that each model was tested consistently across all devices. To ensure the reliability of our results, we repeated the experiments three times for each model on each device. The average values from these repetitions were used for the final analysis.

For accuracy measurement, we wrote Python scripts to utilize the FiftyOne tool to download the COCO validation dataset, which contains 5,000 images. The object detection models were run on this dataset to calculate accuracy metrics, including Precision, Recall, F1 score, and mean Average Precision (mAP). These metrics were recorded in a CSV file. This comprehensive procedure ensured that the performance metrics—inference time, energy consumption, and accuracy—were accurately measured and reported for each model and edge device configuration.

3.3 Experimental Results

3.3.1 Energy Consumption. This section demonstrates the base energy consumption results for Raspberry Pi3, Pi4, and Pi5 devices with and without TPU, as well as the Jetson Orin Nano device.

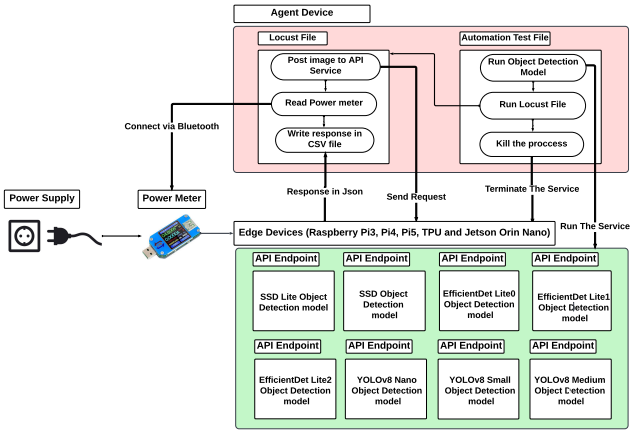


Figure 1: Expermintmal Procedure.

Additionally, it presents the total energy consumption per request and the energy consumption per request excluding the base energy.

To start with Figure 2(a) shows the baseline energy consumption of selected edge devices in milliwatt-hours (mWh). Comparing the different raspberry Pi models, the Pi3 consumes more energy (270 mWh) than both the Pi4 (200 mWh) and the Pi5 (220 mWh). This indicates an improvement in energy efficiency in newer models. However, when considering the TP variants, the energy consumption remains consistent at 300 mWh for both the Pi3 with TP and the Pi4 with TP, while the Pi5 with TP shows a reduced consumption of 250 mWh. This suggests that while base energy efficiency has improved in the Pi4 and Pi5 models, the addition of TPUs results in a higher energy consumption for the Pi3 and Pi4 models, but less so for the Pi5 with TP. Notably, the Orin Nano device demonstrates the highest baseline energy consumption at 350 mWh.

In addition, measuring the energy consumption per request, excluding the base energy, for the investigated object detection models on the evaluated edge devices yields interesting results. Firstly, as Figure 2(b) presents the outcomes on the Pi3, the Det_lite models exhibit energy consumption ranging from 0.41 mWh to 0.98 mWh, while SSD_v1 and SSD_lite models consume 0.31 mWh and 0.41 mWh, respectively. The Yolo8 models demonstrate higher energy demands, spanning from 1.22 mWh to 5.87 mWh. When the TPU is integrated with the Pi3, as Figure 2(c) displays, the Det_lite models consume between 0.32 mWh and 0.61 mWh, and the SSD_v1 and SSD_lite models show reduced consumption of 0.11 mWh and 0.13 mWh, respectively. The Yolo8 models also show decreased energy consumption, ranging from 0.23 mWh to 0.43 mWh. Secondly, the Pi4 introduces improved energy efficiency compared to Pi3, with Det_lite models consuming between 0.14 mWh and 0.33 mWh, and SSD_v1 and SSD_lite models consuming 0.11 mWh and 0.14 mWh, respectively, as Figure 2(d) shows. The Yolo8 models on the Pi4 range from 0.77 mWh to 2.92 mWh. With the addition of the TPU, the Pi4's energy consumption decreases further; Det_lite models consume between 0.13 mWh and 0.19 mWh, and SSD_v1 and SSD_lite models consume 0.10 mWh and 0.11 mWh, respectively. The Yolo8 models range from 0.26 mWh to 0.32 mWh on the Pi4 with TPU, as Figure 2(e) presents. Furthermore,

the Pi5 displays similar energy usage patterns with P4 as figure 2(f) shows. Det_lite models consuming between 0.24 mWh and 0.47 mWh, and SSD_v1 and SSD_lite models consuming 0.22 mWh and 0.24 mWh, respectively. The Yolo8 models on the Pi5 range from 1.02 mWh to 3.58 mWh. Figure 2(g) presents the integration of TPU with Pi5, with Det_lite models consuming between 0.18 mWh and 0.30 mWh, and SSD_v1 and SSD_lite models consuming 0.13 mWh and 0.14 mWh, respectively. The Yolo8 models range from 0.47 mWh to 0.62 mWh on the Pi5 with TPU. Finally, the Jetson Orin Nano as Figure 2(h) demonstrates the lowest energy consumption across all models, with Det_lite models consuming between 0.09 mWh and 0.14 mWh, and SSD_v1 and SSD_lite models consuming 0.01 mWh and 0.06 mWh, respectively. The Yolo8 models on the Jetson Orin Nano range from 0.13 mWh to 0.22 mWh.

Key insights: Pi3 devices generally exhibit higher energy consumption compared to Pi4 and Pi5 models, indicating an improvement in energy efficiency in the newer models. The addition of TPUs consistently reduces the energy consumption for object detection tasks across all Pi models, particularly in the Pi4 and Pi5. However, it is important to note that the addition of TPU has increased the base energy consumption of these devices by 11.11%, 50%, and 13.64% for Pi 3, 4, and 5, respectively. Among all the models tested, the Yolo8_m on the Pi3 consumes the highest energy at 5.87 mWh, while the SSD_v1 on the Jetson Orin Nano consumes the lowest energy per request at 0.01 mWh. The Orin Nano stands out with the highest base energy consumption, while demonstrating the superior overall energy efficiency per request.

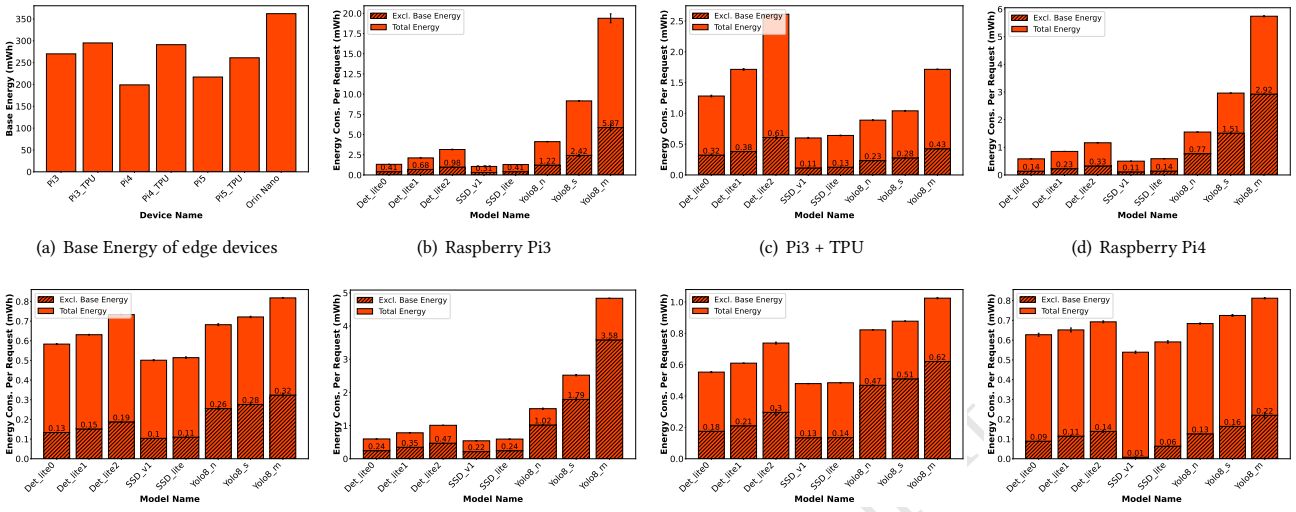
3.3.2 Inference Time. This section analyze the inference times of several object detection models evaluated on diverse edge computing devices. The measurements, reported in milliseconds, reveal distinct performance patterns across these platforms. Beginning with the Raspberry Pi 3, the SSD_v1 model exhibits the lowest inference time at 427 ms among all evaluated models. While the Det_lite models require longer inference times compared to SSD_v1 and SSD_lite, they still outperform all variants of the YOLO8 model. Notably, the YOLO8 models demonstrate the highest inference times, with the maximum recorded at 12,960 ms. When the Coral USB Accelerator is integrated with the Raspberry Pi 3, as shown in Figure 3, the inference times for SSD and YOLO8 models improve significantly, with SSD_v1 remaining the fastest at 61 ms. In contrast, the Det_lite2 model exhibits the highest inference time, taking 1,576 ms. The results reveal that on the Raspberry Pi 4, the SSD_v1 and SSD_lite models exhibited the fastest inference times at 209 ms and 292 ms, respectively. Conversely, the YOLOv8 models across all versions were slower than the Det_lite0 and Lite1 models, with the YOLO8_m representing the slowest at 3671 ms, as shown in Figure 3. The addition of the Edge TPU to the Raspberry Pi 4 significantly reduced the inference times of the SSD and YOLO8 models, with the SSD_v1 model achieving the lowest inference time of 12 ms, while the Det_lite2 model had the highest at 188 ms. Similarly, on the Raspberry Pi 5, the SSD_v1 and SSD_lite models were the fastest, with inference times of 93 ms and 127 ms, respectively. Although the Det_lite models were slower than the SSD models, they were still faster than the YOLO8 models, with the YOLO8_m exhibiting the highest inference time of 1348 ms. The integration of the Edge TPU to the Raspberry Pi 5 further improved the performance of the SSD and YOLO8 models, with the SSD_v1 achieving the lowest inference time of 10 ms and the Det_lite2 having the highest at 139 ms. Finally, on the NVIDIA Jetson Orin Nano, the YOLO8_n model demonstrated the minimum inference time of 16 ms, while the Det_lite and SSD models had similar inference times within the range of 20 ms. The YOLO8_m model remained the slowest at 50 ms, as presented in Figure 3.

Key insights: the SSD_v1 model exhibits the most rapid inference times when deployed across various edge devices. Additionally, the incorporation of Edge TPU, substantially enhances the performance of the evaluated models. Conversely, the YOLO8 models generally demonstrate the slowest inference times among the tested configurations.

3.3.3 Accuracy. This subsection presents the accuracy of various object detection models across different edge device architectures. The mean Average Precision (mAP) on Raspberry Pi devices varied across model sizes, as shown in Figure 4. While the SSD_v1 model had the lowest mAP at 19, the YOLO8_m model achieved the highest mAP of 44 among all evaluated models. The Det_lite0, lite1, and lite2 models exhibited medium mAPs ranging from 26 to 33. Both SSD_lite and YOLO8_n had similar mAPs around 30, while YOLO8_s had a higher mAP of approximately 40. When deploying these models on Raspberry Pi devices equipped with TPU accelerators, the Det_lite and SSD models, including all their versions, demonstrated comparable mAPs to those observed on the standalone Raspberry Pi devices, as illustrated in Figure 4. However, running the YOLO8 models on Raspberry Pis with TPU accelerators resulted in a reduction in accuracy, with YOLO8_n having the lowest mAP of 16. Furthermore, the mAP of YOLO8 object detection models on the Jetson Orin Nano followed a similar accuracy pattern as on the Raspberry Pi, ranging from 31 to 44, as presented in Figure 4. In contrast, the SSD_v1, SSD_lite, Det_lite0, Det_lite1, and Det_lite2 models exhibited a slight decrease in mAP compared to the Raspberry Pi and Edge TPU results. The SSD_v1 model had the lowest mAP at 16, while the SSD_lite model achieved 27. The Det_lite0 model had an accuracy of 23, while the Det_lite1 and Det_lite2 models performed better, with mAPs of 28 and 32, respectively.

Key insights: The YOLO8_m model demonstrates consistently superior accuracy compared to other evaluated models across various device platforms. Conversely, the SSD_v1 model often exhibits the lowest mean Average Precision (mAP) among the tested models. The use of TPU accelerators on Raspberry Pi devices yields similar accuracy levels for the Det_lite and SSD model families, but results in a reduction in accuracy for the YOLO8 models. Regarding the Jetson Orin Nano platform, it exhibits comparable accuracy patterns for the YOLO8 models to the other setups, but shows a slightly lower mAP for the remaining models in comparison to the Raspberry Pi and TPU-equipped configurations.

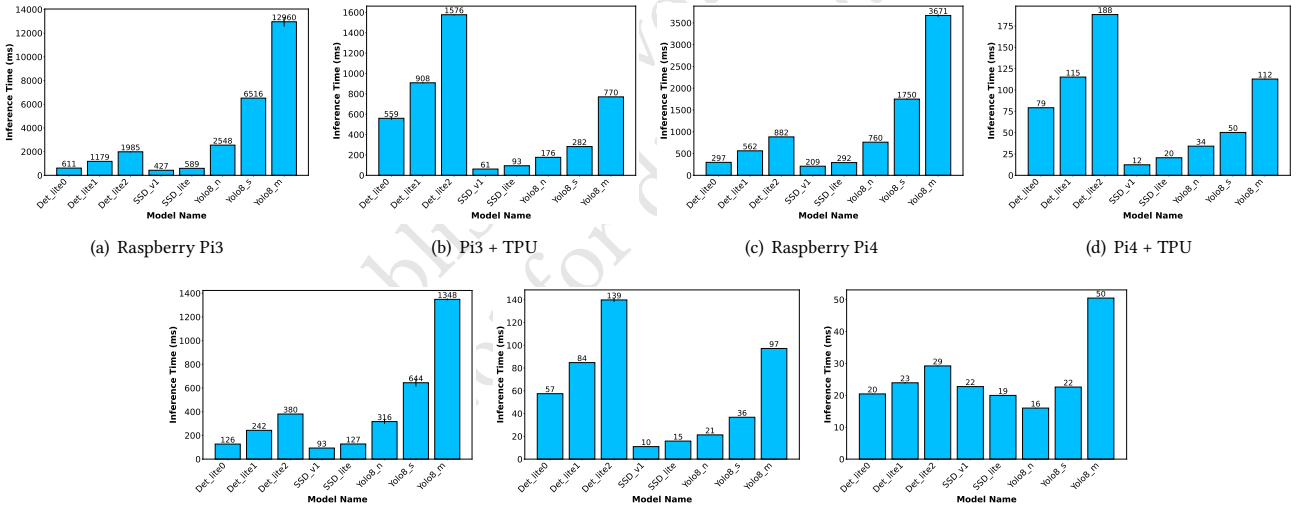
3.3.4 Energy Consumption vs Inference Time. This section highlights the results of the energy consumption versus inference time for object detection models on edge devices. This is essential for informed decisions across various aspects, including but not limited to optimizing performance, extending the battery life of edge devices, meeting real-time application needs, ensuring deployment



(a) Base Energy of edge devices (b) Raspberry Pi3 (c) Pi3 + TPU (d) Raspberry Pi4

(e) Pi4 + TPU (f) Raspberry Pi5 (g) Pi5 + TPU (h) Jetson Orin Nano

Figure 2: Base Energy of edge devices with (a), Energy Consumption per request excluding the base energy / Total Energy consumption per request for different edge devices, with (b) Raspberry Pi3, (c) Pi3 + TPU, (d) Raspberry Pi4, (e) Pi4 + TPU, (f) Raspberry Pi5, (g) Pi5 + TPU, and (h) Jetson Orin Nano



(a) Raspberry Pi3 (b) Pi3 + TPU (c) Raspberry Pi4 (d) Pi4 + TPU

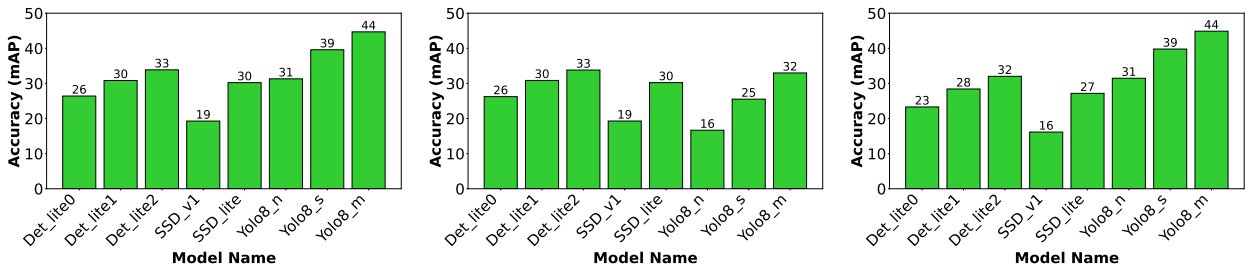
(e) Raspberry Pi5 (f) Pi5 + TPU (g) Jetson Orin Nano

Figure 3: Inference Time per request for different edge devices, with (a) Raspberry Pi3, (b) Pi3 + TPU, (c) Raspberry Pi4, (d) Pi4 + TPU, (e) Raspberry Pi5, (f) Pi5 + TPU, and (g) Jetson Orin Nano

feasibility, reducing costs, minimizing environmental impact, managing thermal output, and enhancing user experience. Henceforth, the energy consumption discussed in this paper excludes the base energy consumption and refers solely to the energy consumption per request, unless explicitly stated otherwise.

When we deploy the SSD_v1 model on the investigated edge devices, it demonstrates the best results regarding energy consumption as displayed in Figure5(a). The Jetson Orin Nano device is the most energy-efficient one and is faster than Pi3 (with or without TPU), Pi4, and Pi5. However, Pi4 and Pi5 with TPU achieve better inference times. Overall, Pi3 has the highest energy consumption and inference time among all the devices.

Testing the SSD_Lite model on the Jetson Orin Nano demonstrates the best results for both energy consumption and inference time compared to the other devices, except for the Pi5 with TPU,



(a) Raspberry Pi (b) Pi + TPU (c) Jetson Orin Nano

Figure 4: Accuracy (mAP) for different edge devices, with (a) Raspberry Pi, (b) Pi + TPU, (c) Jetson Orin Nano

which has a better inference time. In contrast, Pi3 shows the worst results for both energy consumption and inference time.

Moreover, Jetson Orin Nano shows the lowest energy consumption and inference time when deploying the Det_Lite0, Det_lite1 and Det_Lite2 object detection models. However, Pi3 performs the worst device for both energy and inference when runing the all version of Det_Lite models. When the Pi devices integrated with the TPU, they perfume better regarding the energy and infernce as Figure 5(c-e) presents. While the Pi5 with TPU is the fastest device compared to Pi3 and Pi4 with TPU, the Pi4 with TPU consume less energy.

Finally, deploying the YOLO8_n, YOLO8_s, and YOLO8_m models on the Jetson Orin Nano has demonstrated that it is the most energy-efficient device and the fastest at processing object detection tasks among the evaluated edge devices. In contrast, Pi3 has the highest energy consumption and inference time results, as shown in Figure 5(f-h). Additionally, attaching a TPU to the Pi3, Pi4, and Pi5 significantly improves their performance. While Pi3 and Pi4 with TPU consume less energy than the Pi5 with TPU, Pi5 with TPU is faster than both.

Key insights: The Raspberry Pi 3 demonstrates the highest energy consumption and the slowest performance among the devices tested. Integrating TPUs with Raspberry Pi devices significantly improves their performance. However, the Jetson Orin Nano, with its GPU capability, remains superior in both energy efficiency and inference time across the majority of the tested models, with minor exceptions for SSD models.

3.3.5 Energy Consumption vs Accuracy. This section discusses energy consumption versus accuracy. Jetson Orin Nano has the lowest energy consumption and highest accuracy when deploying the SSD_v1 and SSD_lite object detection models, as shown in Figure 6 (a-b). While Pi3, Pi4, and Pi5 have better accuracy results, they are among the most energy-consuming devices. When a TPU is integrated into the Pi devices, energy consumption is reduced without compromising accuracy.

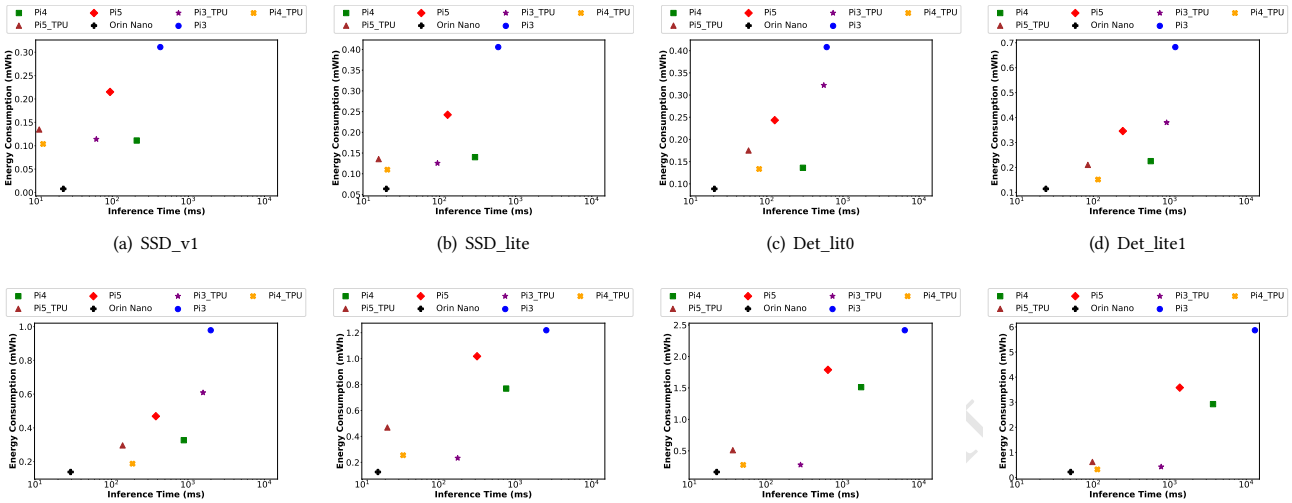
Implementing the Det_lite0, Det_lite1, and Det_lite2 models reveals varied results across the evaluated devices. The Jetson Orin Nano exhibits the lowest energy consumption, although it demonstrates a slight reduction in accuracy when running these object detection models. In contrast, the Raspberry Pi 3, 4, and 5 achieve the highest accuracy, but they also have the highest energy consumption among all devices. However, adding a TPU to the Raspberry Pi devices reduces their energy consumption without impacting the accuracy of the object detection models, as shown in Figure.6 (c-e).

Finally, deploying the YOLO8_n, YOLO8_s, and YOLO8_m models on the Jetson Orin Nano demonstrates that it is the device with the lowest energy consumption while showing similar accuracy to other devices. Although the Pi3, Pi4, and Pi5 also achieve high accuracy similar to the Jetson Orin Nano, they consume more energy. However, integrating a TPU with the Pi devices improves their energy efficiency but significantly affects the accuracy of these models, as shown in Figure 6 (f-h).

Key Insights: There is a trade-off between energy consumption and accuracy. Among the evaluated edge devices, the Jetson Orin Nano can be considered the most energy-efficient when deploying object detection models. However, the Jetson Orin Nano exhibits a slight decrease in accuracy compared to other devices when using SSD and EfficientDet_lite models. Conversely, the YOLO8 models maintain their accuracy when deployed on both the Jetson Orin Nano and Raspberry Pi devices. This could be attributed to the format of the engine file running on the Jetson Orin Nano, which retains the same accuracy as the PyTorch format used on the Raspberry Pi devices. When a TPU is added to Raspberry Pi devices, energy efficiency improves, but the accuracy significantly drops due to the optimization of these models for TPU deployment.

3.3.6 Inference Time vs Accuracy. This section highlights the inference time and accuracy results of the evaluated object detection models across the investigated edge devices. To start with, implementing SSD_v1 and SSD_lite on the Pi5 with TPU reveals that it is the fastest device, achieving the highest accuracy results, as shown in Figure 7(a-b). However, while Pi3 is the slowest device, it matches Pi5 with TPU in terms of accuracy. The Jetson Orin Nano has a shorter inference time than Pi devices without TPU, but this slightly impacts the accuracy of these models.

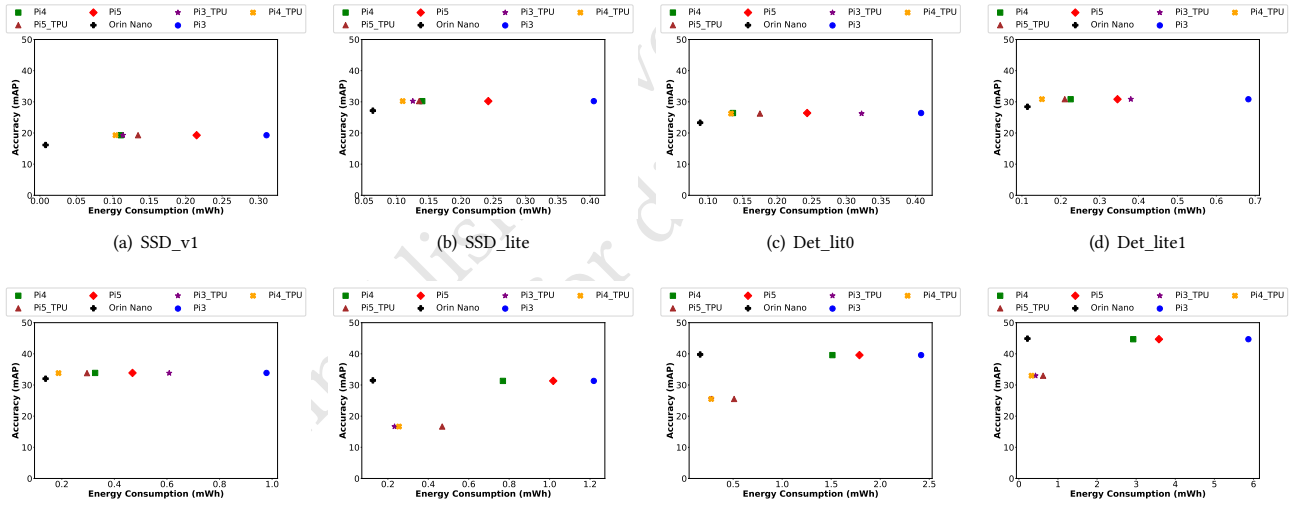
In addition, deploying the Det_lite0, Det_lite1, and Det_lite2 models on the Jetson Orin Nano demonstrates that this edge device is the fastest, although with a slight reduction in accuracy. Conversely, the Raspberry Pi 3, 4, and 5 models, while slower, achieve the highest accuracy results. Additionally, integrating a TPU with



(a) SSD_v1 (b) SSD_lite (c) Det_lit0 (d) Det_lite1

(e) Det_lite2 (f) Yolo8_n (g) Yolo8_s (h) Yolo8_m

Figure 5: Energy consumption per request excluding the base energy vs inference time for different object detection models, with (a) SSD_v1, (b) SSD_lite, (c) Det_lit0, (d) Det_lite1, (e) Det_lite2, (f) Yolo8_n, (g) Yolo8_s, and (h) Yolo8_m



(a) SSD_v1 (b) SSD_lite (c) Det_lit0 (d) Det_lite1

(e) Det_lite2 (f) Yolo8_n (g) Yolo8_s (h) Yolo8_m

Figure 6: Energy consumption per request excluding the base energy vs accuracy for different object detection models, with (a) SSD_v1, (b) SSD_lite, (c) Det_lit0, (d) Det_lite1, (e) Det_lite2, (f) Yolo8_n, (g) Yolo8_s, and (h) Yolo8_m

the Raspberry Pi devices enhances inference time without compromising the accuracy of these models, as shown in the provided Figure7(d-f).

Finally, the Jetson Orin Nano exhibits the fastest inference time and highest accuracy when executing all variants of the YOLO8 models, as demonstrated in Figure 7. Conversely, the Raspberry Pi 3, Pi 4, and Pi 5 achieve comparable accuracy results to the Jetson Orin Nano, but with slower inference times. While integrating a TPU with the Raspberry Pi 3, Pi 4, and Pi 5 improves their inference speed, this enhancement comes at the cost of significantly reduced model accuracy due to the optimizations required for TPU deployment.

Key Insights: There is a trade-off between inference time and accuracy when running object detection models on the Jetson Orin Nano. However, it stands out as the fastest device, maintaining high accuracy with the YOLO8 models due to the utilization of PyTorch format files converted to engine format. In contrast, while the Pi3, Pi4, and Pi5 devices are slower, they achieve the highest accuracy. Integrating a TPU with the Pi devices enhances the inference time

but significantly affects accuracy when running the YOLO8 models. Conversely, running SSD and Det_lite models does not affect the accuracy when using the TPU.

3.4 Energy Consumption vs Inference Time vs Accuracy:

This section presents the results through a three-way comparison. Comparing Energy Consumption, Inference Time, and Accuracy across different object detection models and edge devices helps researchers and practitioners optimize model selection, hardware utilization, deployment strategies, and overall sustainability while reducing operational costs and enhancing user experience. Figure **??** illustrates a 3D visualization of metrics for various object detection models. The analysis indicates that when deploying the SSD_v1 object detection model on edge computing platforms, the Raspberry Pi 4 and Raspberry Pi 5 equipped with TPU accelerators exhibit the fastest inference times without significantly compromising accuracy. However, these edge devices also tend to consume more energy compared to other options. Conversely, the Jetson Orin Nano emerges as the most energy-efficient choice among the evaluated devices, though it may involve a minor trade-off in terms of accuracy and inference speed.

Running SSD_lite on the Raspberry Pi 5 with the Edge TPU accelerator results in the lowest inference time and high accuracy, but it also consumes more energy. On the other hand, while the NVIDIA Jetson Orin Nano is the most energy-efficient device, it exhibits a slight reduction in accuracy and increased inference time compared to the Pi5 with TPU.

Deploying Det_lite0, Det_lite1, and Det_lite2 on the Jetson Orin Nano achieves the fastest inference time and lowest energy consumption, but it slightly impacts the accuracy. In contrast, the Raspberry Pi 3, Pi 4, and Pi 5 with TPU have the highest accuracy, but they consume more energy and take longer for inference compared to the Jetson Orin Nano. However, the TPU significantly improves the performance of the Raspberry Pi without affecting the accuracy.

The analysis reveals that the Jetson Orin Nano is the most efficient device among the tested options, offering favorable energy consumption and inference time without compromising accuracy when running YOLO8_n, YOLO8_s, and YOLO8_m. However, integrating the Raspberry Pis with TPU can improve inference time and energy consumption, but it significantly reduces the accuracy of these models.

Key Insights: When deploying SSD models, the Raspberry Pi 5 with TPU outperforms the evaluated devices in terms of accuracy and inference time, but with slightly higher energy consumption. However, the Jetson Orin Nano is the most energy-efficient device, although it slightly affects accuracy. Conversely, the Jetson Orin Nano outperforms the others when deploying Det_lite models, as it has the lowest energy consumption and fastest performance, but with a slight reduction in accuracy. Yet, if accuracy is crucial, the Pi4 and Pi5 with TPU can be suitable options. When running Yolo8 with all models, the Jetson Orin Nano can be the best choice, as it has the lowest inference time, energy consumption, and highest accuracy.

4 RELATED WORK

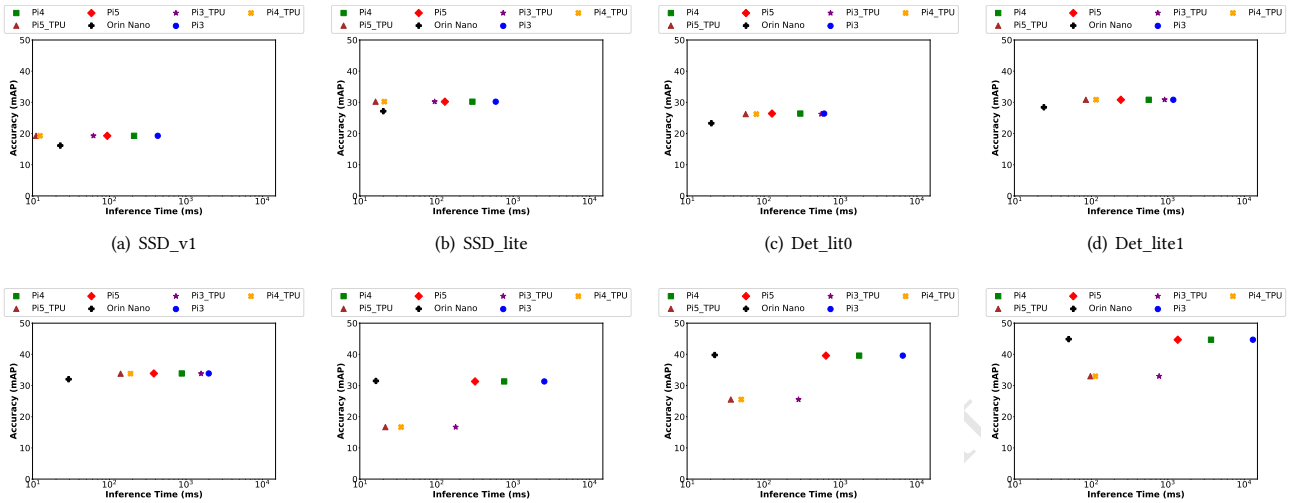
This section provides an overview of the most relevant research on object detection models for edge computing devices and compares our work with existing related work as shown in Table 4. To the best of our knowledge, our work is a unique study due to its comprehensive evaluation of various object detection models and edge devices.. Firstly, Cantero et al. [4] examines various quantization levels and model architectures to determine their efficiency and performance challenges. The research employs the NXP i-MX8M-PLUS application processor and the Google Coral Dev Board with EdgeTPU module, testing models such as SSD (Single Shot Multibox Detection), CenterNet, EfficientDet, and Faster R-CNN. The evaluation metrics include warm-up time, auxiliary processing time, model inference time, model size, inference accuracy, and performance improvement factor. In contrast, our work differs from this study as we measure energy consumption and evaluate YOLOv8 on Raspberry Pi 3, Pi 4, Pi 5, and the Jetson Orin Nano.

Furthermore, Tian et al.[24] provides a comprehensive evaluation of deep learning-based object detection algorithms on the COCO benchmark, focusing on their applicability in smart city environments. The models evaluated include Faster Region-Based Convolutional Neural Network (Faster R-CNN), Mask Region-Based Convolutional Neural Network (Mask R-CNN), Deformable Region-Based Fully Convolutional Network combined with Scale Normalization for Image Pyramids (D-RFCN + SNIP), Neural Architecture Search for Feature Pyramid Network (NAS-FPN), Detector Recursive Feature Pyramid and Switchable Atrous Convolution (DetectorRS), and Dynamic Head (DyHead). The study utilizes performance metrics such as Average Precision, Average Precision at 50% Intersection over Union (AP50), Average Precision for Small Objects (APS), Average Precision for Medium Objects (APM), and Average Precision for Large Objects (APL) to assess the models. On the other hand, our paper focuses not only on accuracy, but also on inference time and energy consumption of these models on limited-resource devices.

Also, Kamath and Renuka [10] examine the efficacy of EfficientDet models, employing integer quantization, for real-time object detection on a Raspberry Pi. The research discusses the trade-offs among model size, precision, recall, and frame rate, ultimately recommending EfficientDet0 and EfficientDet1 for applications on resource-constrained devices. However, this work does not evaluate a wider range of object detection models across diverse edge devices, nor does it measure energy consumption, which are key aspects addressed in our paper.

Additionally, Kang and Somtham [11] evaluation utilized YOLOv4-Tiny and SSD MobileNet V2 models to assess object detection tasks on modern edge devices equipped with various accelerators, including GPUs and TPUs. This study compared detection accuracy, inference latency, and energy efficiency across devices such as the Google Coral Dev Board Mini, NVidia Jetson Nano, and Jetson Xavier NX. Yet, this work differs from our study as it does not examine the EfficientDet model or the latest version of YOLO such as YOLOv8, and also does not include deploying these models on the Raspberry Pi platform.

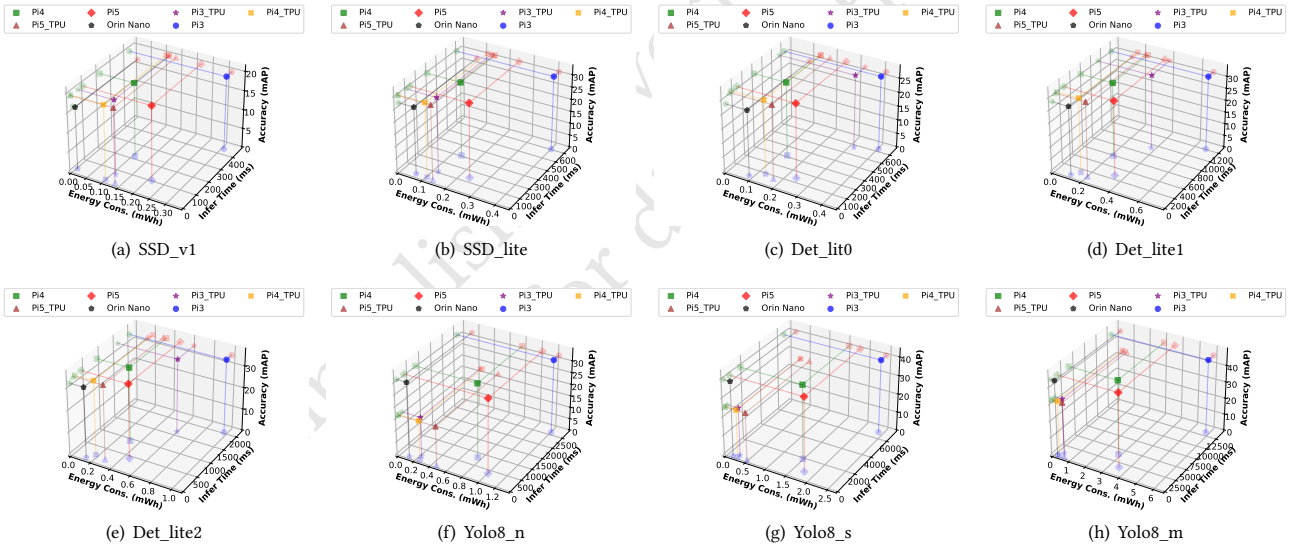
Next, Baller et al. [2] present DeepEdgeBench, a benchmarking framework for assessing the performance of Deep Neural Networks



(a) SSD_v1 (b) SSD_lite (c) Det_lit0 (d) Det_lite1

(e) Det_lite2 (f) Yolo8_n (g) Yolo8_s (h) Yolo8_m

Figure 7: Inference time per request vs accuracy for different object detection models, with (a) SSD_v1, (b) SSD_lite, (c) Det_lit0, (d) Det_lite1, (e) Det_lite2, (f) Yolo8_n, (g) Yolo8_s, and (h) Yolo8_m



(a) SSD_v1 (b) SSD_lite (c) Det_lit0 (d) Det_lite1

(e) Det_lite2 (f) Yolo8_n (g) Yolo8_s (h) Yolo8_m

Figure 8: Inference time per request vs accuracy vs Energy Consumption per request for different object detection models, with (a) SSD_v1, (b) SSD_lite, (c) Det_lit0, (d) Det_lite1, (e) Det_lite2, (f) Yolo8_n, (g) Yolo8_s, and (h) Yolo8_m

(DNNs) on various edge devices, including the Asus Tinker Edge R, Raspberry Pi 4, Google Coral Dev Board, Nvidia Jetson Nano, and Arduino Nano 33 BLE. The evaluation focuses on key metrics such as inference time, power consumption, and accuracy across different models and frameworks. The models tested include MobileNetV1, MobileNetV2, and SSD MobileNetV2, targeting applications in image classification and object detection. However, their evaluation does not cover the most recent object detection models, EfficientDet and YOLOv8.

Moreover, Bulut et al. [3] assess the performance of recent lightweight YOLO object detection models, including YOLOv5-Nano, YOLOX-Nano, YOLOX-Tiny, YOLOv6-Nano, YOLOv6-Tiny, and YOLOv7-Tiny, on the NVIDIA Jetson Nano edge device for applications in traffic safety. It evaluates various metrics such as average precision, inference time, memory usage, and energy consumption. In contrast, our evaluation is more comprehensive as it includes the

recent object detection models such as YOLOv8, SSD, and EfficientDet, which we have examined on the most popular edge devices, including the Raspberry Pi, Edge TPU, and Jetson Orin Nano.

Further more, Chen et al. [5] present a methodology for deploying deep learning object detection on cost-effective IoT devices. This approach employs SSD-MobileNets models and Raspberry Pi 3 units, augmented by Neural Compute Sticks (NCS) for enhanced performance. However, this work does not evaluate other popular object detection models such as EfficientDet and YOLOv8 on a range of edge devices including the Raspberry Pi, Edge TPU, and Jetson Orin Nano. Additionally, it does not consider the energy efficiency of these models when deployed on these edge platforms.

Similarly, Zagitov et al. [26] assess the trade-offs between accuracy, speed, and computational efficiency of various neural network models on Raspberry Pi and NVIDIA Jetson Nano devices. The evaluated models include MobileNetV2 SSD, CenterNet MobileNetV2 FPN, EfficientDet, YoloV5, YoloV7, YoloV7 Tiny, and YoloV8. The study employs metrics such as mean average precision (mAP), latency, and FPS to conduct the evaluations. Yet, it does not consider the energy efficiency of these models or their deployment on the Edge TPU accelerator.

In addition, Galliera and Suri [9] explore the integration of deep learning accelerators with IoT devices to enhance edge computing efficiency for object detection. The research aims to support low-latency, autonomous decision-making processes in both civilian and military applications by deploying advanced object detection models on energy-efficient, embedded devices such as the NVIDIA Jetson Nano, Jetson Xavier, Google Coral Dev Board, Google Coral USB Accelerator, and Intel Movidius Neural Computer Stick 2. The models assessed include YOLOv5, YOLOv5s, and YOLOv5m. In contrast, this study differs from our work as we evaluate the YOLOv8, SSD, and EfficientDet models, and we also measure the energy consumption of these models.

Finally, Lema et al. [12] assess the performance of YOLOv3, YOLOv5, and YOLOX object detection models on various edge computing devices, including the NVIDIA Jetson Nano, Jetson AGX Xavier, and Google Coral Dev Board. Utilizing the MS COCO dataset, the research analyzes the frames per second (FPS) relative to power consumption and cost, and offers practical recommendations for deployment in real-world scenarios. However, this work does not consider other object detection models such as SSD and EfficientDet, nor does it investigate these models on the Raspberry Pi.

5 CONCLUSIONS AND FUTURE DIRECTION

In this paper, we evaluated the performance of state-of-the-art deep learning object detection models including YOLOv8 (Nano, Small, Medium variants), EfficientDet Lite (Lite0, Lite1, Lite2), and SSD (SSD MobileNet V1, SSDLite MobileDet) on popular edge devices such as the Raspberry Pi 3, 4, and 5 with/without TPU accelerators, and the Jetson Orin Nano. We developed an object detection application using Flask-API and utilized various deep learning frameworks, such as TensorFlow Lite, Edge TPU, PyTorch, and TensorRT, to deploy models across different edge device architectures. Additionally, we assessed the accuracy of these models using the FiftyOne tool and COCO datasets, collecting the mean Average Precision (mAP) metric. We also developed an automated system using tools like Locust to examine the performance of the selected object detection models on the investigated edge devices regarding inference time and energy consumption. The base energy consumption of the evaluated edge devices was measured and reported to present the energy consumption per request, excluding the base consumption.

Our evaluation shows a trade-off between accuracy, energy consumption, and inference time. The SSD_v1 model exhibited the lowest energy consumption and fastest inference time among all evaluated models, but this came at the cost of accuracy, as it was the least accurate model. Conversely, the Jetson Orin Nano demonstrated that it is the fastest and most energy-efficient device without compromising the accuracy of YOLOv8 models. However, some models, such as SSD and EfficientDet, experienced a reduction in accuracy when converted to the TensorRT framework. The Edge TPU accelerator enhanced the performance of SSD and EfficientDet models without reducing accuracy, but significantly reduced the accuracy of YOLOv8 models.

For future work, we plan to train our object detection models and evaluate them further. We will also examine different quantized models, such as int8 and float16, when deploying them on Raspberry Pi devices and the Jetson Orin Nano, as these optimizations might significantly impact the results.

REFERENCES

[1] Abhishek Balasubramaniam and Sudeep Pasricha. 2022. Object detection in autonomous vehicles: Status and open challenges. *arXiv preprint arXiv:2201.07706* (2022).

[2] Stephan Patrick Baller, Anshul Jindal, Mohak Chadha, and Michael Gerndt. 2021. DeepEdgeBench: Benchmarking deep neural networks on edge devices. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 20–30.

[3] Anilcan Bulut, Fatmanur Ozdemir, Yavuz Selim Bostanci, and Mujdat Soyturk. 2023. Performance Evaluation of Recent Object Detection Models for Traffic Safety Applications on Edge. In *Proceedings of the 2023 5th International Conference on Image Processing and Machine Vision*. 1–6.

[4] David Cantero, Iker Esnaola-Gonzalez, Jose Miguel-Alonso, and Ekaitz Jauregi. 2022. Benchmarking object detection deep learning models in embedded devices. *Sensors* 22, 11 (2022), 4205.

[5] Chuan-Wen Chen, Shanq-Jang Ruan, Chang-Hong Lin, and Chun-Chi Hung. 2018. Performance evaluation of edge computing-based deep learning object detection. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*. 40–43.

[6] Coral. 2019. *USB Accelerator datasheet.* Technical Report. Google LLC, https://coral.ai/docs/accelerator/datasheet/.

[7] NVIDIA DEVELOPER. 2024. NVIDIA TensorRT. https://developer.nvidia.com/tensorrt#:~:text=NVIDIAÂőTensorRTâĎćis,highthroughputforproductionapplications.

[8] Raspberry Pi Foundation. 2024. About us. https://www.raspberrypi.org/about/

[9] Raffaele Galliera and Niranjan Suri. 2022. Object Detection at the Edge: Off-the-shelf Deep Learning Capable Devices and Accelerators. *Procedia Computer Science* 205 (2022), 239–248.

[10] Vidya Kamath and A Renuka. 2021. Performance analysis of the pretrained efficientdet for real-time object detection on raspberry pi. In *2021 International Conference on Circuits, Controls and Communications (CCUBE)*. IEEE, 1–6.

[11] Pilsung Kang and Athip Somtham. 2022. An evaluation of modern accelerator-based edge devices for object detection applications. *Mathematics* 10, 22 (2022), 4299.

[12] Darío G Lema, Rubén Usamentiaga, and Daniel F García. 2024. Quantitative comparison and performance evaluation of deep learning-based object detection models on edge computing devices. *Integration* 95 (2024), 102127.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 740–755.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 21–37.

Study	CPU	TPU	GPU	Yolov8	EfficientDet	SSD	Inference Time	Energy Consumption	Mean Average Precision (mAP)
Cantero et al.[4]	✓	✓			✓	✓	✓		
Tian et al.[24]									✓
Kamath and Renuka[10]	✓				✓		✓		✓
Kang and Somtham[11]		✓	✓			✓	✓	✓	✓
Baller et al.[2]	✓	✓	✓			✓	✓	✓	✓
Bulut et al.[3]			✓				✓	✓	✓
Chen et al.[5]	✓					✓	✓		✓
Zagitov et al.[26]	✓		✓	✓	✓	✓	✓		✓
Galliera and Suri[9]		✓	✓				✓		✓
Lema et al.[12]		✓	✓				✓	✓	✓
Our Work	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 4: Comparison of Studies Based on Device Architectures and Key Criteria

[15] Raspberry Pi Ltd. 2023. *Raspberry Pi 3 Model B+*. Technical Report. https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf.

[16] Raspberry Pi Ltd. 2024. *Raspberry Pi 4 Model B*. Technical Report. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf.

[17] Raspberry Pi Ltd. 2024. *Raspberry Pi 5*. Technical Report. https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf.

[18] Nvidia. 2024. NVIDIA Jetson Orin. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

[19] Abidha Pandey, Manish Puri, and Aparna Varde. 2018. Object detection with neural models, deep learning and common sense to aid smart mobility. In *2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI)*. IEEE, 859–863.

[20] PyTorch. 2024. GET STARTED. https://pytorch.org/

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[22] Mingxing Tan, Ruoming Pang, and Quoc V Le. 2020. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10781–10790.

[23] TensorFlow. 2024. For Mobile Edge. https://www.tensorflow.org/lite

[24] Jiya Tian, Qiangshan Jin, Yizong Wang, Jie Yang, Shuping Zhang, and Dengxun Sun. 2024. Performance analysis of deep learning-based object detection algorithms on COCO benchmark: a comparative study. *Journal of Engineering and Applied Science* 71, 1 (2024), 76.

[25] Voxel51. 2024. FiftyOne. https://voxel51.com/fiftyone/

[26] A Zagitov, E Chebotareva, A Toschev, and E Magid. 2024. Comparative analysis of neural network models performance on low-power devices for a real-time object detection task. *Computer* 48, 2 (2024).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009