Accelerate Emerging Computer Vision Models for Video Surveillance As A Service (VSaaS) using Edge TPU

Kent Anh BUI[1], Nguyen Trung TRAN[1], and Khoi Cong TRAN[1], Trong Nhan LE[1*], Ngan Le-Song PHAM[2]

[1] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
[2] Marvell Technology Vietnam, Ho Chi Minh City, Vietnam
∗ Corresponding author: trongnhanle@hcmut.edu.vn

Abstract. Video Surveillance as a Service (VSaaS) is a newly created term that refers to the cloud-based storage, management, and remote monitoring and alerting of live video surveillance footage. In order to establish a reliable and sustainable edge AI system for remote alerts in a VSaaS system, several crucial factors need to be considered. The performance and accuracy aspects are of utmost importance, particularly as the system tackles increasingly complex tasks. One effective solution to address these challenges is to optimize the existing devices with an Edge TPU (Tensor Processing Unit) to accelerate the inference speed as well as inference accuracy. This paper will consider deeply into these categories, and providing insightful statistics and analysis on the topic

Keywords: edge computing, edge devices, machine learning, computer vision, embedded, video surveillance as a service, VSaaS

1 Introduction

VSaaS, or Video Surveillance as a Service, is a solution that empowers organizations for efficient management, utilization, and automatic their video surveillance camera systems control through cloud-based platforms. It offers cost-effective access to cloud computing resources and extensive storage capabilities, enabling organizations to record, store, access, detect, and analyze live video feeds without the hassle of managing the infrastructure themselves associated with companies' huge investment for equipment configurations. VSaaS simplifies this process by centralizing computational resources and sharing them among multiple clients, making it economically viable for both service providers and consumers. Apart from corporations, smaller settings like homes and offices that monitoring solution is also a need but the settings acquire small-scale infrastructure. They sought a cloud-based services for managing and live-streaming video feeds, eliminating the need for complex infrastructure maintenance.

However, despite the advantages of the current Video Surveillance as a Service (VSaaS) system and cloud computing in general, a significant limitation persists in the performance of detection and analysis processes. Due to the delegation of resource-intensive tasks to the cloud, there is a noticeable delay in executing real-time AI-related or other computationally intensive tasks. In response to this challenge, researchers and developers have proposed a solution to enhance performance by transferring resource-intensive tasks, such as detection and analysis, from centralized cloud servers to edge devices. This approach optimizes computational efficiency and minimizes latency in real-time AI-related tasks [1].

In this paper, the methodology of **offloading VSaaS intensive response from centralized cloud to edge devices** is approached for optimization by **implementing the Google Coral AI Accelerator**, a hardware device equipped with **a custom-built system-on-chip (SoC) featuring a powerful neural processing unit (NPU) tailored for running AI models at the edge**. Then, the effectiveness of the Edge TPU in **key criteria like inference time, precision and frame processed in a unit of time is assessed to illustrate the performance optimization of Google Coral TPU within the VSaaS architecture**.

Following this section, related works are provided with reference topics about co-design of software and hardware for deploying Edge AI on Edge devices in Section 2. The proposal of design is approached in Section 3 including system architecture, frameworks & machine learning models, and system criteria & implementation. Section 4 is resulted for design outcomes and further analysis regarding to the suitability of the devices for specific Edge AI applications.

2 Related Works

When deploying Edge AI, particularly in the field of computer vision, several factors come into play in [2], Which include the size, architecture, required input, and supported operations of the model. Together, these factors determine the model's inference speed, accuracy, and the number of frames processed within a unit of time. The importance of on-the-edge inference in AI, as demonstrated in previous works, has motivated further extensive research in the field of Edge AI. Additionally, numerous researchers have focused on techniques that strike a balance between accuracy and performance in order to develop a lightweight models that could be use by edge devices to optimize mentioned factors. These techniques include quantization [3], and create mobile-specific models [4]. Commercially, hardware [5] and frameworks on edge devices have also undergone upgrades to keep pace with the evolving demands for Edge AI within their system.

In addition, prior work by Xu et al. [6] provided a clear insight into the current landscape of Video Surveillance as a Service. As the significance of cameras

rises in smart systems (such as smart agriculture, retailing, and smart cities), there is an increasing demand for large-scale, cloud-based multi-camera processing systems. Their approach has aimed at utilizing smart security cameras with processing capabilities as the foundation for video analytics, which is AI-on-camera (AI at the edge). It requires deep integration with cameras from different manufacturers. The presented idea is intriguing, and foundational models for implementation in the proof-of-concept are provided by the paper.

All together, the deployment of Edge AI in a VSaaS (Video Surveillance as a Service) environment is well-consideration in the fields of detection and management; which provides ability to enhance the overall system performance compared to cloud-based AI analytics. In a demonstration outlined in [7], the paper demonstrates how efficient edge deployment can be achieved by combining YOLOv5 with the Coral Edge TPU, emphasizing the importance of matching software and hardware for successful edge AI applications. YOLOv5's model, utilizing a 320x256 input, was utilized alongside micro-services for inference and visualization, resulting in a raw inference time of roughly 300 ms and an overall power consumption of approximately 3.5W; despite being connected via USB2 without SIMD, the TPU yielded a 10x speedup compared to the CPU.

Expanding upon these findings and achievable results, the target of this system carry out is to evaluate the effectiveness of the Coral Edge TPU on various emerging computer vision models with modifications for generalization purposes upon key criteria outcomes.

3 Design proposal

3.1 System architecture

The central emphasis of the system lies in the Edge of the VSaaS architecture, where the performance of diverse machine learning models on different Edge devices is evaluated, with a particular focus on assessing the influence of incorporating the Coral Edge TPU. The objective is to gauge how integrating the Coral Edge TPU improves the performance of machine learning models in Edge computing contexts. These experiments aim to provide insights into the effectiveness and advantages of deploying the Coral Edge TPU for accelerating and optimizing machine learning tasks at the Edge.

The architecture in figure 1 illustrates the general system architecture when employing Edge AI to VSaaS as Edge Devices. Edge devices are principle target of the system as well processing module media server, other services and database being responsible for receiving and distributing video streams from IP cameras to VSaaS edge devices. There are two main components in the edge devices, which are the Edge Processor and the Coral Edge TPU.

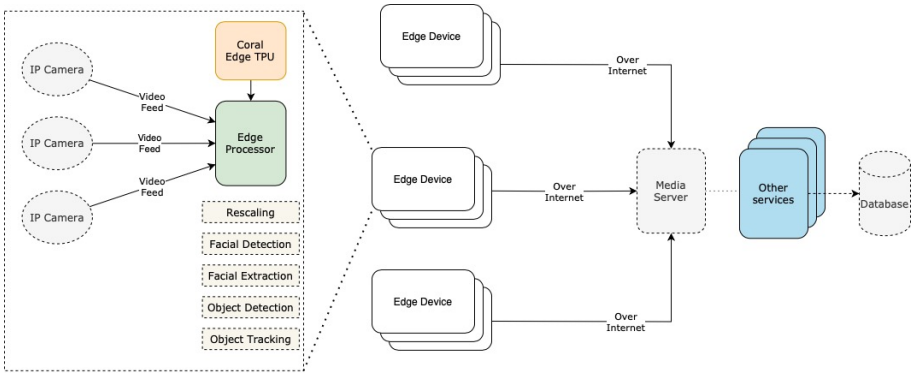


Fig. 1: System architecture

Edge Processor: The Edge Processor selection was enumerated:

- **Raspberry Pi 4 model B**: Created by the Raspberry Pi Foundation, this 4th version of the Raspberry Pi series is a single-board, light-weight embedded computer, equipped with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor which is suitable for simple video analysis projects [8] and has proven to be easily accessible yet effective in Edge computing.
- **NUC Intel Pentium Silver**: This advanced machine has the capability to simultaneously perform multiple popular neural network tasks, including tasks like image classification, human detection, and object tracking. With remarkable efficiency, it adeptly manages parallel operations essential for these applications, ensuring swift and precise computational prowess.

Coral Edge TPU: Each Edge Device integrates a processing unit and utilizes the Edge Coral USB Accelerator hardware, featuring a Tensorflow Processing Unit co-processor. This enables high-speed machine learning inferencing across various systems by connecting it to a USB port. The inclusion of the Tensorflow Processing Unit (TPU) allows execution of Tensorflow Lite (.tflite) models and facilitates conversion to Edge TPU Tensorflow Lite models, maximizing efficiency when used with the Coral USB Accelerator.

	GPU	**CPU**	**RAM**
Raspberry Pi 4 model B	Broadcom VideoCore VI	ARM Cortex-A72	2GB LPDDR4 SDRAM
NUC Intel Pentium Silver	Intel UHD 605 Graphics	Intel Pentium Silver J5040	4GB*2 DDR4-2400
Coral USB Accelerator	**N/A**	Quad-core ARM.A53	**N/A**

Table 1: Edge devices' specification

3.2 System Implementation

3.2.1 Framework and Machine Learning Models Several frameworks exist for running Machine Learning models on Edge AI devices. To optimize the advantages of the Coral Edge TPU, the system meticulously integrates a dedicated TensorFlow Processing Unit designed for executing (.tflite) files. The choice of TensorFlow Lite is not solely based on its simplicity; instead, it encompasses a range of tools enabling on-device machine learning. These tools empower developers to deploy their models efficiently on mobile, embedded, and edge devices, ensuring precise and swift execution [9].

Three well-known models were taken advantages in design for object detection with low inference time and light-weight on Edge devices for performance testing on Edge TPU.

- **Yolo V8 Nano [10]**: is a cutting-edge, state-of-the-art model, enhances performance and flexibility in object detection and tracking. Performance testing on various hardware platforms is vital for assessing its efficiency in real-time applications, particularly on resource-limited edge devices and when accelerated by the Google Coral Edge TPU.
- **SSD MobileNet V2 [11]**: MobileNet Single Shot Detector (MobileNetSSDv2) is a real-time object detection model with *267 layers and 15 million parameters*, optimized for devices such as smartphones or edge devices. It consists of a base MobileNetV2 network with a SSD layer for image classification. Performance testing on an edge-optimized model demonstrates the Edge TPU's acceleration capabilities.
- **EfficientDet Lite [12]**: EfficientDets are a family of object detection models, which achieve state-of-the-art 55.1mAP on COCO test-dev. Selecting this model for testing serves to provide a benchmark reference point against other models, offering a clearer perspective on their respective performance and capabilities.

Model name	Base validation dataset	Input size	Model size
Yolo V8 Nano	90 objects COCO	320×320×3	5.3 MB
SSD MobileNet V2	90 objects COCO	300×300×3	6.7 MB
EfficientDet Lite	90 objects COCO	320×320×3	5.7 MB

Table 2: Object Detection Models' specification

When comparing a Tensorflow Lite model running on the Coral Edge TPU with a model that doesn't utilize the Coral Edge TPU, it is crucial to perform quantization on the model. Quantization involves converting the model 32-bit into its 8-bit version, which is necessary to ensure compatibility with the Edge TPU. By quantizing the model, it ensures the capabilities of the Edge TPU for enhanced performance and efficiency.

For quantization model, the process of transforming the original 32-bit float

number TensorFlow model into an 8-bit version and subsequently converting it into an Edge TPU model using the provided compiler will involve employing quantization. For simplicity, quantization-aware training will be chosen over post-quantization [13]. A "fake" quantization modules, called Q/DQ (quantize then de-quantize) was used during training to imitate the behavior of the testing or inference phase. For example, the weights of the neural network, are rounded or clamped to 16-bit floating point or 8-bit integer representations during training. The quantization process (scaling, clipping, and rounding) is incorporated into the training process, allowing the model to be trained to retain its accuracy even after quantization, leading to benefits during deployment (lower latency, smaller model size, lower memory footprint).

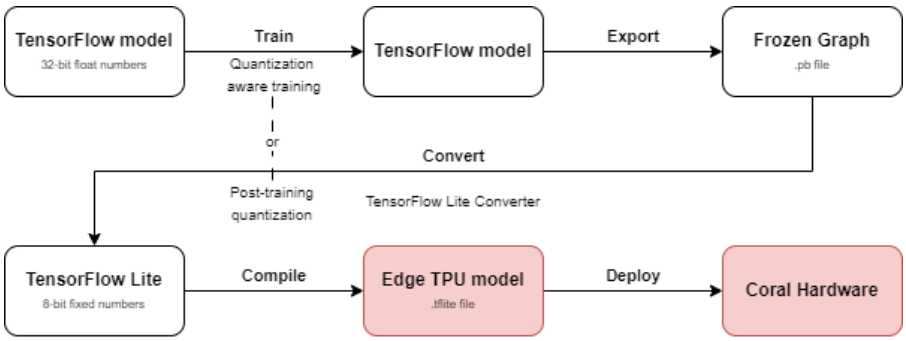


Fig. 2: Quantization process

After having the 8-bit TensorFlow Lite trained model file, we need to compile it using the provided Edge TPU compiler from Coral to compile it into a format that Coral Edge TPU could understand, illustration of the compiling tflite to edgetpu compatible process is referenced below:

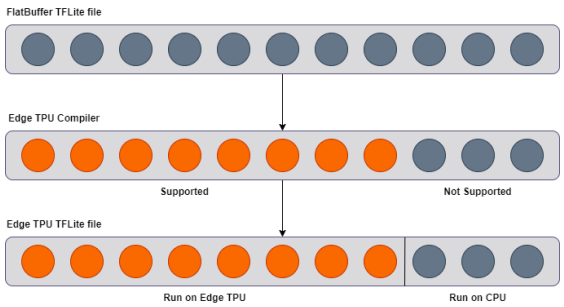


Fig. 3: Compilation process

If the model does not meet all the required quantization, it continuously compiles and runs, but only a portion of the model is executed on the Edge TPU. At the first point in the model graph where an unsupported operation occurs, the compiler partitions the graph into two parts. The first part of the graph that contains only supported operations is compiled into a custom operation that executes on the Edge TPU, and everything else executes on the CPU.

3.2.2 Experiment Criteria

Video Feed FPS:

Unlike static images or isolated sensor data, the temporal dimension of video data offers a continuous stream of evolving information crucial for tasks such as object tracking, behavior analysis, and prolonged detection. In video surveillance systems, the frames per second (FPS) processed is a vital factor, as higher FPS enables more accurate analysis by capturing additional temporal information and minimizing the risk of overlooking critical events.

Rather than conducting the experiment on any particular dataset, footage captured by surveillance cameras at the edge will be utilized for outcome comparison. Two edge devices have been selected: the Raspberry Pi 4 model B, paired with the External Rapoo C200 FullHD 720p 24fps camera, and the NUC Intel Pentium Silver, also equipped with the same camera.

Models' Average Inference Time:

Another criteria to be examined is the inference time with and without the utilization of the Edge Coral TPU. Inference time holds significance in application solutions involving computer vision. Real-time applications frequently encounter varying resource availability for each inference, and modern deep learning models based on transformer architecture can entail considerable computational expenses. The utilization of the Coral Edge TPU can adapt model execution to fulfill real-time application resource constraints, facilitating efficient inference without additional training.[14].

Models' Average Accuracy:

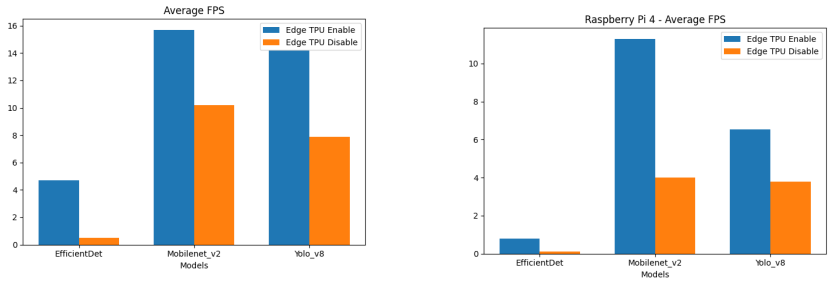
Ensuring a harmonious blend of speed, processing capability, and precision is fundamental in optimizing Edge AI effectively. While aspects like swift inference and enhanced frame processing are pivotal, accuracy remains paramount, particularly in surveillance applications. Emphasizing both speed and accuracy guarantees dependable and efficient Edge AI solutions for surveillance cameras.

4 Results

Based on experimental criteria, the outcome of each target has been resulted and performed analysis by comparison table between Edge Processor outputs and illustrative figures.

4.1 Average frame processed in a unit of time

Utilizing the webcam on edge devices enables frame processing rates per unit of time. Figure 4a visually represents the frame processing results for each model. Two runs were conducted for each model: one with the Edge TPU turned on for testing purposes, and another with it turned off. Similarly, the experiment was repeated on a Raspberry Pi 4, and the corresponding outcomes are depicted in Figure 4b.



(a) Frame per second on NUC

(b) Frame per second on Raspberry Pi 4

Fig. 4: Average FPS on Edge Devices

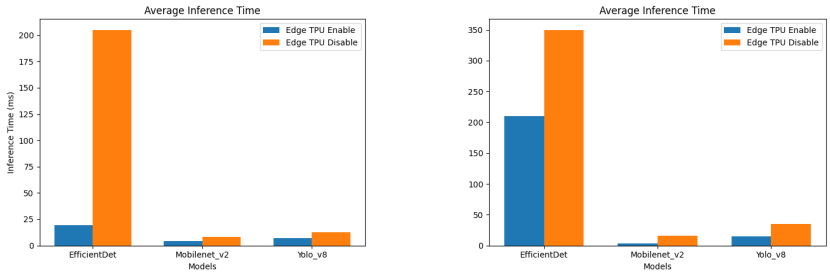
Based on the results obtained from table 3, it is evident that all three models exhibit improved performance in terms of frames per second (FPS) when the Coral Edge TPU is enabled.

Devices	**Models**	**With TPU**	**Without TPU**
NUC	EfficientDet	5	< 1
	MobileNetV2	16	9
	Yolov8	15	7
Raspberry Pi 4	EfficientDet	1	< 1
	MobileNetV2	15	4
	Yolov8	6	4

Table 3: Average FPS comparison On Edge Devices

4.2 Average model inference time

For this experiment, the Intel IoT Devkits sample videos specifically designed to test object detection models [15] were selected as the testing input.



(a) Average inference time on NUC (b) Average inference time on Rasp Pi 4

Fig. 5: Average inference time on Edge devices

Devices	Models	With TPU	Without TPU
NUC	EfficientDet	20	200
	MobileNetV2	4	15
	Yolov8	8	18
Raspberry Pi 4	EfficientDet	210	330
	MobileNetV2	12	28
	Yolov8	20	40

Table 4: Average inference time (ms) comparison on edge devices

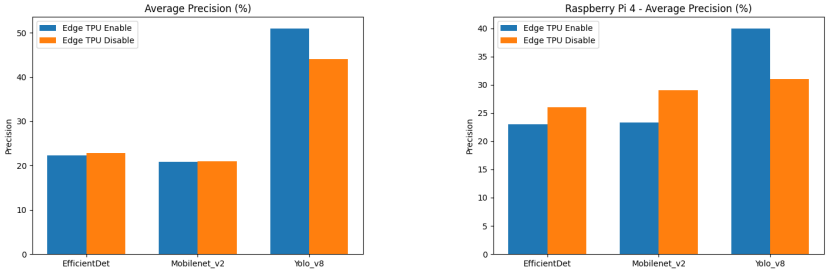
EfficientDet's extended inference time primarily stems from TensorFlow Lite runtime's incomplete optimization for quantized models on various platforms. Yet, integrating the Coral Edge TPU yields a notable reduction, typically by 3-5 times for most models, with a remarkable near 10 times decrease observed for EfficientDet on NUC. This enhancement results from transferring a substantial portion of computational tasks to the Coral Edge TPU, thereby accelerating overall processing speed.

From the result, it can be confirmed that: EfficientDet, its own, was not optimal for deployment on weak CPU devices but with Coral Edge TPU assistance, it becomes much more viable. Conversely, the lightweight MobileNet emerged better choice for Edge devices, while the widely-used Yolo model performed adequately, albeit averagely, on such devices. Though, the YoloV8 model might present greater viability with appropriate training. Additionally, without the Coral Edge TPU, the NUC was shown to be (30-40%) faster on average than Raspberry Pi 4. In general, the Coral Accelerator equalized speed differences between NUC and Raspberry Pi 4 for optimized models, offering developers a cost-effective option. Instead of buying costly new devices, developers can choose a Coral Edge TPU at a lower price. This enables comparable performance without extra expenses on hardware.

4.3 Average model precision

The Intel IoT Devkits sample videos (Intel)[15] were chosen as the test input for these experiments. While the Coral Edge TPU's impact may not be as striking

as with previous criteria, it still notably enhances performance, especially considering its affordability compared to full-fledged edge devices. Moreover, the variance in accuracy among different edge devices is not significant, suggesting that developers can achieve satisfactory results with the Coral Edge TPU. This offers a cost-effective solution without compromising overall application accuracy.



(a) Average precision on NUC (b) Average precision on Raspberry Pi 4

Fig. 6: Average precision on edge devices

Devices	Models	With TPU	Without TPU
NUC	EfficientDet	22%	23%
	MobileNetV2	20%	21%
	Yolov8	50%	45%
Raspberry Pi 4	EfficientDet	23%	26%
	MobileNetV2	24%	29%
	Yolov8	40%	31%

Table 5: Average precision comparison on edge devices

The recorded precision results closely matched the official documentation provided by Coral Edge TPU. Documentation include 22.4% for SSD MobileNetV2, 30.4% for EfficientDet, and 52.5% for YoloV8 (Models), according to[16].

5 Conclusion

This research had analyzed prediction accuracy, inference time, and frame processing within a VSaaS system on Edge devices, alongside examining the computation process of the Edge TPU. Utilizing the Google Coral Edge TPU notably enhances the performance of machine learning models across various edge devices. The presented outcomes aim to guide users in informed decisions regarding Edge AI adoption for specific applications, considering available frameworks and platforms. It's crucial to acknowledge that findings are based on limited device availability and potential biases that may have influenced the results.

Future work could involve further exploration and optimization of emerging computer vision models tailored for Video Surveillance as a Service (VSaaS)

applications, with a specific focus on leveraging Edge TPU for accelerated inference. The evaluation targets to hardware with the Coral Edge TPU's impact on heat generation, pricing, and energy consumption. The objective of this study is to offer insights into the performance and efficiency of Edge AI deployments, aligning with specific application requirements. By considering these factors, users can make informed decisions for sustainable, cost-effective, and low-maintenance Edge AI implementations in the long term.

Acknowledgment

The work is acknowledged from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM.

References

1. C. F. Andrea Pazienza, Giulio Mallardi and F. Vitulano, "Artificial intelligence on edge computing: a healthcare scenario in ambient assisted living,"
2. V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus,"
3. Y. B. M. Courbariaux and J.-P. David, "Training deep neural networks with low precision multiplications," 2014.
4. M. Z. A. Z. M. Sandler, A. Howard and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.
5. Y. X. B. A. T. K. R. Hadidi, J. Cao and H. Kim, "Characterizing the deployment of deep neural networks on commercial edge devices," *EEE International Symposium on Workload Characterization*, 2019.
6. M. Xu, Y. Liu, and X. Liu, "A case for camera-as-a-service," *IEEE Pervasive Computing*, vol. 20, 2021.
7. R. Prokscha, M. Schneider, and A. Hoess, *Efficient Edge Deployment Demonstrated on YOLOv5 and Coral Edge TPU*, pp. 141–155. 08 2023.
8. T. Gizinski and X. Cao, *Design, Implementation and Performance of an Edge Computing Prototype Using Raspberry Pis*, pp. 0592–0601. 2022.
9. J. Dai, "Real-time and accurate object detection on edge device with tensorflow lite," *Journal of Physics: Conference Series*, vol. 1651, p. 012114, 11 2020.
10. G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023.
11. M. Z. A. Z. L.-C. C. Mark Sandler, Andrew Howard, "Mobilenetv2: Inverted residuals and linear bottlenecks," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 01 2018.
12. N. Jyeniskhan, A. Keutayeva, G. Kazbek, M. Ali, and E. Shehab, "Integrating machine learning model and digital twin system for additive manufacturing," *IEEE Access*, vol. PP, pp. 1–1, 01 2023.
13. T. C. E. T. Documentation, "Tensorflow models on the edge tpu," *https://coral.ai/docs/edgetpu/models-intro/*.
14. K. Sreedhar, J. Clemons, R. Venkatesan, S. Keckler, and M. Horowitz, "Enabling and accelerating dynamic vision transformer inference for real-time applications," 12 2022.
15. I. IoT, "Intel sample iot video," *https://github.com/intel-iot-devkit/sample-videos*.
16. StereoLabs, "Performance benchmark of yolo v5, v7 and v8," *https://www.stereolabs.com/blog/performance-of-yolo-v5-v7-and-v8*.