Comparative Evaluation of Machine Learning Inference Machines on Edge-class Devices

Amanatidis, Petros; Iosifidis, George; Karampatzakis, Dimitris

DOI
10.1145/3503823.3503843

Publication date
2021

Document Version
Final published version

Published in
Proceedings - 25th Pan-Hellenic Conference on Informatics, PCI 2021

Comparative Evaluation of Machine Learning Inference Machines on Edge-class Devices

Petros Amanatidis
peamana@cs.ihu.gr
Department of Computer Science,
International Hellenic University
Kavala, Greece

George Iosifidis
g.iosifidis@tudelft.nl
Delft University of Technology
Delft, Netherlands

Dimitris Karampatzakis
dkara@cs.ihu.gr
Department of Computer Science,
International Hellenic University
Kavala, Greece

ABSTRACT

Computer science and engineering have evolved rapidly over the last decade offering innovative Machine Learning frameworks and high-performance hardware devices. Executing data analytics at the edge promises to transform the mobile computing paradigm by bringing intelligence next to the end user. However, it remains an open question to explore if, and to what extent, today's Edge-class devices can support ML frameworks and which is the best configuration for efficient task execution. This paper provides a comparative evaluation of Machine Learning inference machines on Edge-class compute engines. The testbed consists of two hardware compute engines (i.e., CPU-based Raspberry Pi 4 and Google Edge TPU accelerator) and two inference machines (i.e., TensorFlow-Lite and Arm NN). Through an extensive set of experiments in our bespoke testbed, we compared three setups using TensorFlow-Lite ML framework, in terms of accuracy, execution time, and energy efficiency. Based on the results, an optimized configuration of the workload parameters can increase accuracy by 10%, and in addition, the class of the Edge compute engine in combination with the inference machine affects execution time by 86% and power consumption by almost 145%.

CCS CONCEPTS

• **General and reference** → **Evaluation**; • **Computing methodologies** → *Object detection*; **Machine learning**.

KEYWORDS

edge computing, machine learning, inference machine

ACM Reference Format:
Petros Amanatidis, George Iosifidis, and Dimitris Karampatzakis. 2021. Comparative Evaluation of Machine Learning Inference Machines on Edge-class Devices. In *25th Pan-Hellenic Conference on Informatics (PCI 2021), November 26–28, 2021, Volos, Greece.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3503823.3503843

PCI 2021, November 26–28, 2021, Volos, Greece

ACM ISBN 978-1-4503-9555-7/21/11…$15.00
https://doi.org/10.1145/3503823.3503843

1 INTRODUCTION

In recent years, advances in computer science and engineering have created a new technological ecosystem where new technologies like the Internet of Things (IoT), Edge-Fog-Cloud computing, Machine Learning (ML) and wireless communication networks are shaping new applications of Cyber-physical interconnected devices.

On one hand, Cloud computing is an established technology of data production and consumption from billions of online users and systems. A typical Cloud-based system is productive in many use cases but as applications becoming more complicated and demanding, many critical issues arise related to network efficiency, systems scalability, and privacy of data [10].

On the other hand, computing technologies at the Edge of modern networks are coming as a promising solution to all these critical issues. A lot of research is focused on Edge computing to enable applications which require the execution of computing tasks in proximity with users. The Edge-class devices are available at affordable prices and consist of modern high-performance hardware components (CPU, memory systems), power-efficient modes, and configurations with dedicated Application Specific Integrated Circuit (ASIC) hardware for ML acceleration. In combination with the hardware, the ML frameworks offer rich software libraries and APIs for developers to create robust applications for demanding environments [3].

Meanwhile, Machine Learning (ML) on power-efficient devices is an emerging technology. While utilization of ML at the cloud has been studied extensively, the question of running efficiently, in terms of performance and power consumption, ML tasks at Edge-class devices remains an open question. In this approach the ML development splits into two separate procedures, as presented in Figure 1. The ML workload (model) training is performed at the Cloud using GPUs and high-performance infrastructures, and the ML workload inference is deployed on Edge-class devices [7].

Our focus is exactly on this topic, which we investigate by deploying a bespoke testbed and launching a battery of experiments with representative Edge-class devices and ML software libraries. In this work, we pre-train the workload using Google Cloud infrastructures and use the trained model to perform a comparative evaluation of two ML inference machines on different Edge-class compute engines. Our goal was to implement hardware-software ML configurations and to provide figures for the impact of substantial metrics like execution time and energy efficiency.

The rest of the paper is organized as follows. In the following section, research work related to ML technologies and applications is presented. In Section 3, we describe our experimental setup and the metrics for the evaluation of ML inference machines, and in

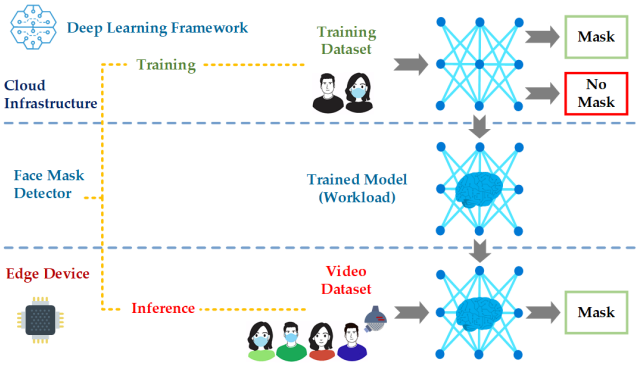

Figure 1: ML training and inference processes

Section 4 we present and discuss the results of the comparative evaluation. We conclude the paper providing our main findings and suggestions for further work.

2 RELATED WORK

In recent years, much work has been done related to ML frameworks development for Edge Computing applications. S. Nikouei et al. [8] supports the idea that Edge Computing allows efficient execution of computational tasks on edge devices. Today, time-sensitive applications could be performed on Edge-class devices to reduce latency or even allow real-time decision making. Object detection is a typical application where loading a huge amount of video streaming data to the cloud takes valuable time and overloads communication networks. As a solution to this challenge, a light-weight Convolutional Neural Network (CNN) was introduced by the authors for human detection at the Edge. The proposed model was trained based on the ImageNet dataset. The results showed that this method is promising for surveillance tasks because of its decent accuracy and good processing speed.

A. Castello et al. in [2] developed a framework that performs efficiently model inference with deep neural networks on multi-core Arm processors. This inference machine which is called PyDTNN is very user-friendly and supports Deep Learning (DL) networks such as CNNs and transformers. The results showed that the PyDTNN is highly competitive compared to other inference machines such as TensorFlow-Lite and Arm NN. The results highlighted the importance of image's batch size for response time (images/sec). The Arm NN inference machine is better than the PyDTNN in terms of the processing time by 23%.

K. Park et al. [9] introduced a model using MobileNet V2.0 plus SSD, and a quantization system that detects face mask using Google Edge TPU (Tensor Processing Unit). This work approves that Google Edge TPU is capable to deploy a real-time application. The quantized model compared with the 32-bit floating point (FP32) model, shows extremely low-latency without sacrificing accuracy, which practically means that processing cost and network delays could be avoided.

A. Jainuddin et al. in [1] presented remarkable results in terms of inference time for Deep Learning (DL) application. Google Edge TPU accelerator which was developed specially for edge devices is presented. The authors deployed DL models like MobileNet V1.0 and MobileNet V2.0 with Input Shape 224x224x3, on Edge-class devices. Furthermore, they compared the performance of the edge devices with and without the accelerator using TensorFlow Deep Learning models. The results showed that Google Edge TPU device performs faster than the other.

Lastly, Y. Hui et al. [6] proposed a benchmarking methodology to evaluate three different Edge AI processors (i.e., Google Edge TPU, NVIDIA Xavier and NovuTensor). The evaluation of these Edge AI processors was measured using metrics of latency, accuracy, and energy efficiency. These metrics demonstrate the performance of a Deep Learning application. The measured value for accuracy was the mAP (Mean Average Precision), which is the most popular metric to evaluate object detection workloads with images. The results of this work showed that the accuracy is for all Edge AI processors almost the same, but comparing it in terms of latency the NovuTensor is faster than Google Edge TPU and NVIDIA Xavier

3 EXPERIMENTAL SETUP

We deployed three experimental setups with the same ML framework and workload. In the two first setups, we chose a CPU-based Edge-class device in conjunction with two inference machines, and in the last setup we used a TPU accelerator. In detail, our testbed is described as follows.

3.1 Edge-class Device Hardware

The main Edge-class device for our experimental setup is a Raspberry Pi 4 Model B Single Board Computer (SBC). This SBC device has an Arm Cortex A72 (ARM v8) 64-bit SoC with 4 cores running at 1.500 Frequency (MHz) and 4.000 RAM Size (MB) of PDDR4-3200 SDRAM type. The SBC operating system is Linux raspberrypi 5.4.42-v8+ aarch64.

Additionally, we used a Google Edge TPU ML accelerator (Coral USB Accelerator). Coral has an advanced ASIC implementation for the TPU that provides high performance acceleration supported by an Arm 32-bit Cortex-M0+ MCU. Also, Coral has a USB 3.1 Gen 1 port to communicate with Raspberry Pi 4. Coral is specially developed for processing tensors which makes it ideal for AI and computer vision applications and performs 4 trillion operations (tera-operations) per second (TOPS), using 0.5 watts for each TOPS (2 TOPS per watt).

3.2 ML Framework and Inference machines

Each CPU and accelerator manufacturer provides a ML framework to support the development of AI applications. A framework user utilizes all the available libraries and API functions to inference various models on hardware edge devices [12].

In our work we performed our evaluation with ML TensorFlow-Lite (TFLite) framework. TFLite is an open-source, product ready, cross-platform deep learning framework that converts a pre-trained model in TensorFlow to a special format that can be optimized for speed or storage. Each ML frameworks supports software libraries for workload inference. On the basis of TFLite framework we deployed two inference machines (i.e., TFLite and Arm NN) on CPU's and TPU's hardware, as presented in Figure 2, and described as follows:

- TensorFlow-Lite: is an optimized version of TensorFlow ML framework targeting mobile and edge devices. In our setup we used the inference machine for models without metadata and the TensorFlow-Lite Interpreter API (Python 3.7).
- Arm NN: is a ML inference API for Linux developed by Arm. Arm NN is built on top of the Arm Compute Library (ACL) and provides acceleration for Arm CPUs with Neon (Neon Backend). This inference engine bridges the gap between the existing neural networks (NN) frameworks, such as TFLite, allowing it to run efficiently and optimized, across Arm Cortex-A CPUs. Neon architecture using SIMD (Single Instruction Multiple Data) technology provides additional instructions that can perform mathematical operations in parallel on multiple data streams. In our test we used an Arm Cortex A72 CPU, PyArmNN 21.0.0, and Arm NN 19.08 inference machine with TFLite API Loader. Arm NN provides a TFLite parser for loading neural networks defined by TFLite FlatBuffers files into the Arm NN Runtime.
- Google Edge TPU: the Coral uses the Edge TPU compiler software. This software compiles a compatible TFLite model into a native binary for deployment on Coral. We worked with TFLite Python API (Python 3.7) and Edge TPU Runtime.

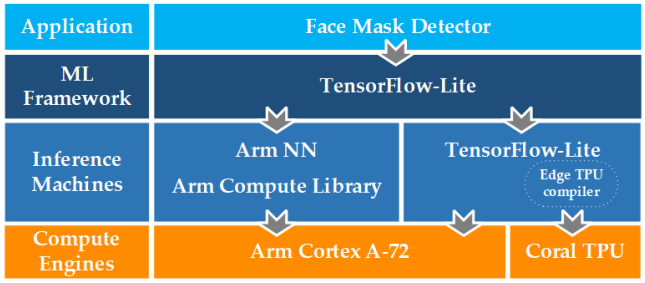

Figure 2: A block diagram of the components of our experimental setup

3.3 ML Workload

Firstly, for our evaluation we chose the MobileNet V1.0 workload [5] which is one small image classification network and we trained it with Transfer Learning in Colab. In this approach, the host computer runs a Python development environment and TensorFlow, Keras and OpenCV libraries are installed. In our implementation we added some new layers over the frozen layers to train the model on a Kaggle training dataset for a face mask object detection problem [4]. More specifically, the training dataset contains 7.553 images, of which 3.725 are labeled (Tag) with-mask and 3.828 are labeled without-mask. The model was trained for 20 epochs.

The trained model was converted to TensorFlow-Lite (quantized) before the deployment on the Edge-class device. There are two options to quantize a FP32 model: the quantization-aware training and the full integer post-training quantization. For this work we used the full integer post-training quantization. The full integer post-training quantization is a technique to convert a previously-trained model (FP32) into a quantized model (INT8). The next step is to deploy the workload for inference. Inference refers to the process of using a trained workload to make a prediction. In our workload the system predicts if a person wear or not a face mask. Each ML inference machine has various parameters but the basic configuration for our experiments is presented in Table 1.

Table 1: Inference machines configuration in our evaluation experiments

Inference Machine	Workload	Data type	Input Shape
Arm NN 19.08	MobileNet V1.0	INT8	128x128x3
TensorFlow-Lite	MobileNet V1.0	INT8	224x224x3
Google Edge TPU	MobileNet V1.0	INT8	224x224x3

For the Arm NN inference machine the Input Shape is different than the other two because Arm NN 19.08 does not support the MobileNet V1.0 with Input Shape 224x224x3.

3.4 Video Dataset

The video dataset for the evaluation consists of 15 videos captured in real conditions with a webcam. Each video contains a person wearing a face mask or not. The file format of the videos is .mp4 (we named them V1.mp4, V2.mp4, etc.) with resolution 640x480 pixels and a file size between 5 to 9 MB each.

3.5 Metrics

In literature, benchmark methodologies and metrics for measuring ML inference machine performance are presented in works [6, 11]. Some of these metrics were adapted to our experimental measurements. The metrics of this work are as follows:

- Frame Count: is the number of frames of a video file.
- Accuracy: is the fraction of successful predictions of our model divided by the total number of predictions, as presented in the following equation. In our binary workload, Accuracy is calculated in terms of positive and negative predictions. We checked frame by frame all four possible outcomes: TP = True Positives, TN = True Negatives, FP = False Negative, and FN = False Positives, for all the files of our video dataset.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Execution Time: is the period of time for inference processing.
- Throughput: is a measurement in Deep Learning to determine the performance of various models for a specific application. Throughput refers to the number of data units processed in one unit of time. For video data is frames per second (fps).
- Energy Efficiency: is a very important metric for Edge-class devices and measures the number of frames processed per unit power, which is usually expressed as performance per watt or frames per second per watt.
- Total Power Consumption: This metric includes the total power consumed by the SBC for all the tasks during the execution of each video file. This metric includes power

consumption for video loading, inference processing and results logging.

A low-cost USB Voltmeter/Ammeter-Power Capacity Tester was used for power consumption measurements. This device measures USB port and device operating current and the output voltage. To calculate Energy Efficiency metric, Average power was used. Average power (in watt) is the average of the instantaneous power values dissipated over the time period of ML inference.

4 RESULTS

The experimental results, as presented in Table 2, highlight the superiority

of the hardware acceleration compared to the CPU-based inference machines. Based on these results, TFLite inference machine with Coral accelerator in execution time is faster 50% than the Arm NN and almost 86% than the CPU-based TFLite. Focusing on the Accuracy metric, TFLite with and without the Coral accelerator has almost the same results. TFLite gives higher Accuracy than Arm NN and as a result performs a better prediction. This occurs because the TFLite used model has different Input Shape than the Arm NN model.

Comparing Energy Efficiency results, it is obvious that TFLite with Coral accelerator is the best solution for ML workload processing at the Edge. As illustrated in Figure 3, the Energy Efficiency metric for TFLite with Coral accelerator is at least 3 times higher compared to Arm NN. In more detail, the results showed that TFLite with Coral consumes almost 145% less power than without Coral. Compared to the Arm NN, the TFLite with Coral consumes almost 103% less power. Finally, TFLite consumes 64% more energy than Arm NN. In Figure 4, we present the Total Power Consumption results for running the application.

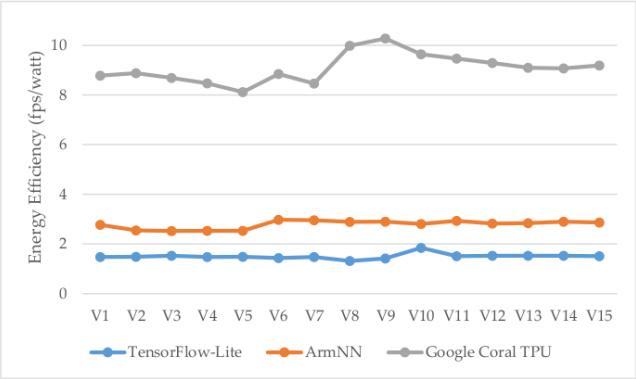

Figure 3: Energy Efficiency metric results

Summarizing, the Coral accelerator performs better, as expected, comparing to the other two inference machines running on the CPU-based Edge-class device. The reason is that Coral is dedicated to accelerating forward-pass operations, which make it an efficient solution for performing inferences. Also, Coral's ASIC hardware offers real-time performance as it performs at >30 frames per second. Comparing the CPU-based setups, Arm NN and TensorFlow-Lite, Arm NN is much faster because of the ARM v8 Neon technology, which enables SIMD technology and optimizes inference process.

Admittedly, the price cost of ML inference machines is a factor that must be taken into account. The Arm NN and TFLite inference machines without the Google Coral TPU need only a SBC with 4 GB RAM for the Arm NN and up to 1 GB RAM for the TFLite. Using the Google Coral TPU accelerator with a SBC increases the cost by almost 85 Euros.

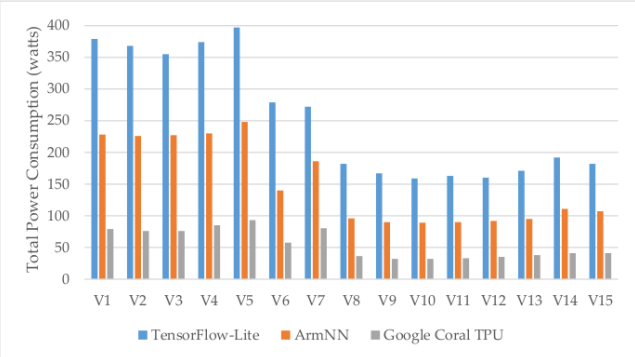

Figure 4: The Total Power consumption for the video dataset

Finally, a critical factor for the selection of an Edge AI setup is the difficulty of integration and installation of the ML frameworks and inference machines. The TFLite ML inference machine is the easiest to integrate on an SBC by using the appropriate available Python packages. On the other hand, building Arm NN libraries, backends, and its parsers from source is not always easy to accomplish.In order to take advantage of the Google Coral, TPU resources the developers use TFLite and the only restriction is to compile the TFLite model with the Edge TPU compiler.

5 CONCLUSIONS

In this paper, we present a comparative evaluation of ML inference machines using two compute engines on Edge-class devices. The experimental results approve that Edge computing applications are able to perform advanced ML workloads and will solve emerging issues in critical applications. The advanced hardware architectures in conjunction with efficient software libraries are becoming useful tools for engineers to develop energy-efficient and optimized solutions at the Edge. The testbed results show that an optimized configuration of the ML model parameters increases accuracy by 10%, the type of the Edge compute engine in combination with the inference machine affects execution time by 86% and power consumption by almost 145%. For future work, we strongly believe that evaluation methodologies, benchmarks, and ML architecture selection tools are necessary for Edge Computing ML applications development and research efforts must be focused in this area.

ACKNOWLEDGMENTS

This work was supported by the MPhil program "Advanced Technologies in Informatics and Computers", hosted by the Department of Computer Science, International Hellenic University, Greece.

Table 2: Experimental Results

Inference Machine	Video .mp4	Frame Count	Throughput fps	Exec. Time sec	Acc. %	Infer. Power watts	Energy Effic. fps/watt	Total Power watts
TFLite	V5	568	5.7	100.0	92.0	382.0	1.49	397.0
Arm NN	V5	568	10.5	54.0	81.0	224.0	2.53	248.0
Edge TPU	V5	568	39.4	14.4	92.0	69.9	8.12	93.0
TFLite	V6	387	5.4	71.0	83.0	268.0	1.43	279.0
Arm NN	V6	387	11.0	33.0	74.5	122.0	2.98	140.0
Edge TPU	V6	387	40.7	9.5	82.5	43.7	8.85	57.5
TFLite	V9	222	5.2	42.0	86.0	154.0	1.42	167.0
Arm NN	V9	222	10.5	21.0	77.5	76.0	2.90	90.0
Edge TPU	V9	222	41.1	5.4	85.5	21.6	10.28	32.0

REFERENCES

[1] Ahmad Ammar Asyraaf Jainuddin, Yew Cheong Hou, Mohd Zafri Baharuddin, and Salman Yussof. 2020. Performance Analysis of Deep Neural Networks for Object Classification with Edge TPU. In *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*. IEEE, Selangor, Malaysia, 323–328. https://doi.org/10.1109/ICIMU49871.2020.9243367

[2] Adrián Castelló, Sergio Barrachina, Manuel F. Dolz, Enrique S. Quintana-Ortí, and Pau San Juan. 2021. High performance and energy efficient inference for deep learning on ARM processors. *arXiv:2105.09187 [cs]* (May 2021). http://arxiv.org/abs/2105.09187 arXiv: 2105.09187.

[3] Jiasi Chen and Xukan Ran. 2019. Deep Learning With Edge Computing: A Review. *Proc. IEEE* 107, 8 (2019), 1655–1674. https://doi.org/10.1109/JPROC.2019.2921977

[4] Omkar Gurav. [n. d.]. Face Mask Detection Dataset | Kaggle. https://www.kaggle.com/omkargurav/face-mask-dataset

[5] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]* (April 2017). http://arxiv.org/abs/1704.04861 arXiv: 1704.04861.

[6] Yujie Hui, Jeffrey Lien, and Xiaoyi Lu. 2020. Early Experience in Benchmarking Edge AI Processors with Object Detection Workloads. In *Benchmarking, Measuring, and Optimizing*, Wanling Gao, Jianfeng Zhan, Geoffrey Fox, Xiaoyi Lu, and Dan Stanzione (Eds.). Vol. 12093. Springer International Publishing, Cham, 32–48. https://doi.org/10.1007/978-3-030-49556-5_3 Series Title: Lecture Notes in Computer Science.

[7] He Li, Kaoru Ota, and Mianxiong Dong. 2018. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Network* 32, 1 (2018), 96–101. https://doi.org/10.1109/MNET.2018.1700202

[8] Seyed Yahya Nikouei, Yu Chen, Sejun Song, Ronghua Xu, Baek-Young Choi, and Timothy R. Faughnan. 2018. Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN. In *2018 IEEE International Conference on Edge Computing (EDGE)*. IEEE, San Francisco, CA, 125–129. https://doi.org/10.1109/EDGE.2018.00025

[9] Keondo Park, Wonyoung Jang, Woochul Lee, Kisung Nam, Kihong Seong, Kyuwook Chai, and Wen-Syan Li. 2020. Real-time Mask Detection on Google Edge TPU. *arXiv:2010.04427 [cs]* (Oct. 2020). http://arxiv.org/abs/2010.04427 arXiv: 2010.04427.

[10] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[11] Peter Torelli and Mohit Bangale. [n. d.]. Measuring Inference Performance of Machine-Learning Frameworks on Edge- class Devices with the MLMark™ Benchmark. https://www.eembc.org/mlmark

[12] Amir Yazdanbakhsh, Kiran Seshadri, Berkin Akin, James Laudon, and Ravi Narayanaswami. 2021. An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks. *arXiv:2102.10423 [cs]* (Feb. 2021). http://arxiv.org/abs/2102.10423 arXiv: 2102.10423.