(100)

Profiling the Performance and Energy Efficiency of Edge Accelerators in the Context of Computer Vision

R. B. Ayala Meza and Jérémie Farret
Inmind Technologies Inc., Montreal, Canada
Tel.: + 1(514)871-0470
E-mail: rayalameza@inmindtechnologies.com

Abstract: Deep neural networks (DNNs) are extensively used in many complex artificial intelligence (AI) applications despite of being compute-intensive and hence causing substantial energy consumption. Nowadays, with the ramping up of Internet of Things (IoT) devices, emerging Edge Computing and Edge AI practices; diverse types of embedded accelerators have appeared. These equipment come armed with multi-processor system-on-chip (MP-SoC) such as graphics processing units (GPU) and tensor processing units (TPU) accommodating requirements for different applications. However, to the wide deployment of DNNs in these constrained hardware platforms, power management needs to be improved without sacrificing application performance. This article will evaluate Google's Coral dev board and Nvidia's Jetson TX2 with benchmarks and comparison metrics from the standpoint of the process to create a model that is compatible with each edge accelerator to the thermal regulation of power through dynamic voltage and frequency scaling (DVFS).

Keywords: Convolutional neural networks, GPU, TPU, Embedded system, Hardware architecture, Edge.

1. Introduction

Neural nets were proposed in 1940 but they have seen a breakthrough in the field of computer vision in 2012, when the AlexNet system won the ImageNet challenge in image recognition [1][2]. Since then, deep neural networks (DNNs) have been successfully used in multiple applications because of its ability to extract high level features from huge amounts of raw data and state-of-the-art accuracy. Many DNNs models with different network architectures (number of layers, layer types, layer shapes, etc) have been developed. The tendency is that in order to achieve higher accuracy, the depth in number of convolutional layers must increase accordingly [3]. However, this comes at the cost of high computational complexity and hence more energy consumption [4],

DNNs processing has different computational needs. For example, training is usually done in the cloud meanwhile inference can occur in the cloud or in the edge. In applications such as industrial processes and drone navigation, real-time inference near the sensor is longed [5]. Accordingly, embedded accelerators are mainly deployed to reduce connectivity dependency, latency and security risks. Thus, they have limited energy consumption, compute and memory [5, 6]. Although these accelerators are equipped with highly-parallel compute paradigms, including both temporal (CPU, GPU) and spatial architectures (TPU) to address this issue, thermal management also has to be taken into consideration.

Dynamic voltage and frequency scaling (DVFS) is a technique which dynamically varies frequency and voltage based on the present thermal constraints on the processors, and thus optimizes energy conservation [7, 8].

This article will focus in the task of image classification, specifically in the multiple versions of GoogleNet (also known as Inception). To the best of our knowledge, this is the first work addressing DVFS to evaluate the devboards Google's Coral [9] and Nvidia's Jetson TX2 [10] (see Fig. 1) for the case of real-time applications (batching is not allowed) in low precision inference (16 bits and 8 bits). Furthermore, it will be discussed the workflow and frameworks to create a compatible model for each one.

The remainder of this article is organized as follows:

- Section 2 provides background and reviews previous works.
- Section 3 introduces the characterization methodology and discusses its results.
- Section 4 depicts the protocol adopted to this study, according to the research program overseeing it, and the hardware architectures applicable for such results.
- Section 5 presents the key insights from this work and possible future research directions.

Fig. 1. Google's Coral (left) and Nvidia's Jetson TX2 (right) devboards.

2. Background and Related Work

Temporal and spatial Architectures

These are highly-parallel compute paradigms to achieve performance. Temporal architectures, can be found in CPU and GPU. Some techniques are SIMD (Single Instruction Multiple Data) which uses vectors, and SIMT (Single Instruction Multiple Thread) which employs parallel threads. The main characteristic is that the ALUs (Arithmetic Logic Units) cannot communicate directly, depend on a centralized control and can only fetch data from a memory hierarchy. In contrast, spatial architectures found in ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array), use dataflow processing. Here, the ALUs can communicate directly and sometimes have their own control logic and local memory [3].

Dynamic Voltage and Frequency Scaling (DVFS)

It is an energy conservation technique that aims to optimize performance and power consumption (which retrospectively affects temperature) of processors during runtime. The notion behind it is to transition between different system clocks whenever needed. But, the dilemma lies in finding the right time to change frequency and the right value of the frequency itself. [8]

The manner in which the processor's frequency is scaled is determined by the scaling algorithm and the current processor's load. These algorithms can be part of the kernel code or of a specialized firmware and can be classified as proactive and reactive.

- Reactive: actions are taken when a certain state is reached.
- Proactive: actions are taken when a certain state is determined to happen in the future.

The state could be temperature (a trip_point) or workload. And the actions could be voltage and frequency control.

Optimization techniques

These aim to reduce the storage, memory requirement and computational cost by compressing the model size. And thus decrease bandwidth requirements and energy consumption, which are important in edge devices.

- Quantization: It reduces the size of weights and activation functions by converting all the 32-bit floating point (FP32) numbers to the nearest lower precision number. For example 16-bit floating point (FP16) or 8-bit fixed point (INT8). Although these representations can be less precise, the inference accuracy of DNNs is not significantly affected.
- Pruning: This technique eliminates low-weight connections between the layers by setting them to zero.
- Layer fusion: Consists in combining layers, which have similar operations and parameters, in a single one. This reduces the number of transfers from memory to registers and vice versa.

GoogleNet models

These models were heavily engineered to force performance in terms of speed and accuracy, as prior to their launch, most DNNs will just pile up many convolutional layers hoping to get better performance. Each version is a refinement over the previous one: Inception_v1 (I_v1), Inception_v2 (I_v2), Inception_v3 (I_v3) and Inception_v4 (I_v4).

The authors in [7] have investigated the impact and energy conservation of DVFS but on datacenter's GPUs. A similar analysis has been done in [11] with edge accelerators, but thermal management has been studied using a thermal camera. Finally, the investigation in [12] profiled the DVFS mechanism but only on a high-end embedded CPU (ARM Cortex-A15 available in an Odroid-XU4 board). Unlike those works, this investigation targets embedded GPU and TPU.

3. Experimental Setup and Results

The objectives are the following: (i) understand when successive inference induces temperature violations (trip points); (ii) quantify the performance (latency, inference time, memory usage) of each embedded accelerator under thermal management by DVFS and different artificial intelligence (AI) models.

For this experiment, the 4 versions of the Inception model were used. All the evaluation metrics were collected after classifying the same image (187KB) 70000 times with both devboards operating headless (no mouse, keyboard, screen, Wi-Fi were present during the inference execution).

3.1. Software Setup - Creating a Compatible Model

Both devboards are hardware-constrained. So, the models need to be compressed to run inference. Fig. 2 shows the process used to create a compatible model.

- Zoo: It depicts a repository were the state-of-the-art models are stored in different format: frozen (.pb), saved (.saved), checkpoint (.ckpt), etc. All the Inception models were downloaded from the official TensorFlow repository. [13]
- Model format: For both devboards, frozen models have been used.
- Network definition: Here, models are serialized, for efficient reading, using parsers like TFLite (Coral) and onnx (TX2).
- Optimization: Models are tuned up to increase performance and reduce memory usage. Each devboard has their own method. On one hand, the TX2 uses TensorRT, an algorithm optimizer, which will benchmark different techniques and pick the most suitable for the target GPU, batch size and other parameters (it is important to do this step within the devboard) and lower precision to FP16. On the other hand, a requirement to use Coral's TPU is to have a model with precision

INT8. Other available techniques are pruning and weight clustering.

- Save file: Models are transformed into a format to store and use at a later time for inference.

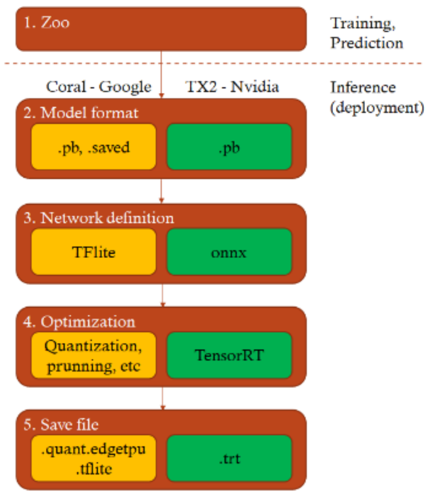

Fig. 2. Process to create a compatible model

3.2 Hardware Setup

In Table 1, each multi-processor system-on-chip (MP-SoC) is described.

3.3 Results

In Fig. 3, it is observed Coral's temperature, used memory and inference time. In Iv_1, Coral never throttles. Meanwhile in I_v4, it has throttled 10 times reaching 65.55 °C as maximum temperature. This amount of throttling is correlated to the model's size. I_v1 is the smallest one and I_v4 is the biggest one, as can be seen in Table 2. But there is a chance of throttling in I_v1 if repeating this experiment after Coral has reached its idle temperature of 59.5 C.

The biggest amount in memory usage is once again with the biggest model (I_v4 = 741.12MB). Also, all models show the highest usage at the end of the experiment.

The first inference time in all models is the slowest one because the time includes loading the model into the TPU and initializing it. Additionally, as in the other figures, it is always the biggest model (I_v4) that is more computationally expensive with 121.83ms. Finally, it seems that I_v3 and I_v4 have more fluctuations in the inference timing than the other 2 models.

In Fig. 4, it is observed that the maximum temperature reached is 63°C in I_v4. But unlike Coral it has throttled only once in this model. So, it seems that there is a type of control because temperature does not descend and ascend abruptly, which explains why the inferences are faster in TX2 than Coral. Additionally, TX2 can maintain a lower idle temperature (31.5 °C) than Coral (59.5 °C).

In term of memory usage, the TX2 uses more than Coral. It has used approximatively 3200MB in the 4 models, but this is related to the type of optimization applied to the models. TensorRT, optimizes by benchmarking different techniques and to do this it has to execute inference. When configuring TensorRT, the amount of memory to use has been set to 4096 MB. This can be reduced but it will affect to certain amount the accuracy of the model as less techniques can be benchmarked.

Table 1. Specifications of MP-SoC.

	Coral devboard	**Jetson TX2**
MP-SoC	**CPU:** Cortex-M4F, Quad Cortex-A53 **GPU:** GC7000 Lite **TPU**: edge coprocessor	**CPU:** Dual Denver2, Quad Cortex-A57 **GPU:** 256-core Nvidia Pascal
DRAM	1GB LPDDR4	8GB LPDDR4
Architecture	Spatial: ASIC	Temporal: SIMT
Frequency (MHz)	MAX: 500	MAX-Q: 854 MAX-N: 1302
Trip_points (°C)	**CPU:** 65, 75, 80, 85, 90 **TPU:** 84.8, 89.8, 94.8	**CPU:** 95.5, 99.5, 100.5 **GPU:** -40, -5, 30, 65, 95.5, 100, 100.5, 101 **FAN:** 0, 51, 61, 71, 82, 140, 150, 160, 170, 180
Power (W)	10 -> 15	7.5 -> 15
Frameworks	a) TensorFlow b) TensorFlow Lite (optimization)	a) TensorFlow b) TensorRT (optimization)
Precision	INT8	FP32, FP16
Optimization tools	Quantization, pruning, weight clustering	Layer&Tensor fusion, quantization, kernel autotuning, dynamic tensor memory

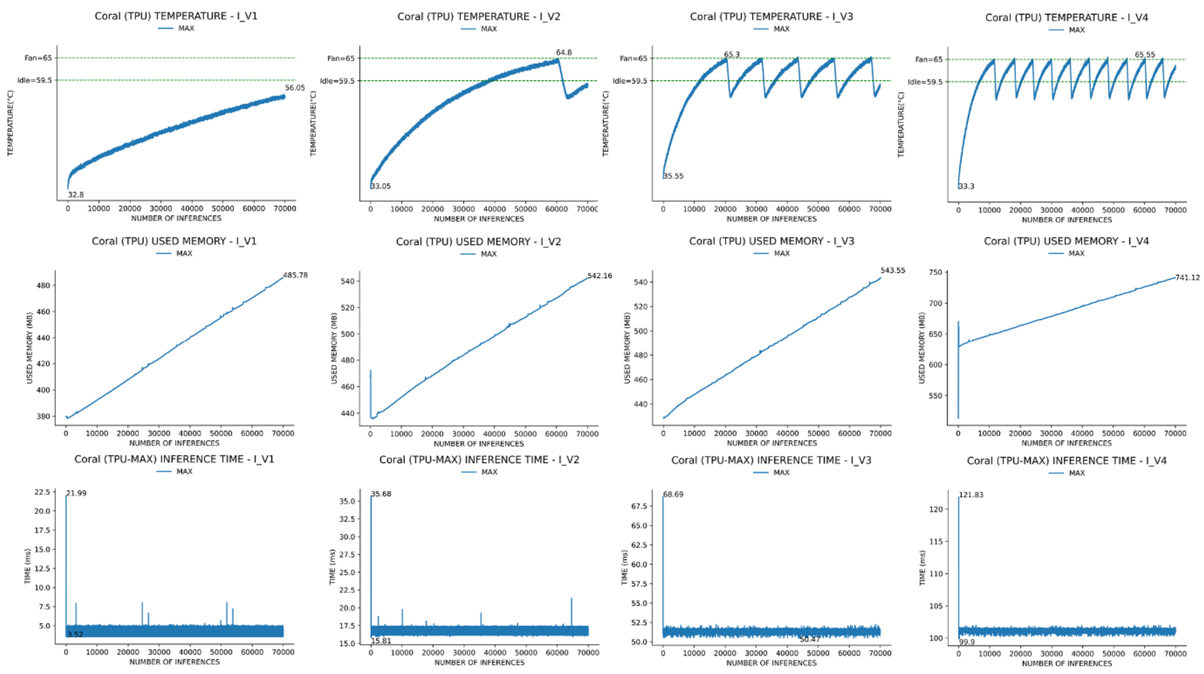

Fig. 3. Coral's evaluation metrics.

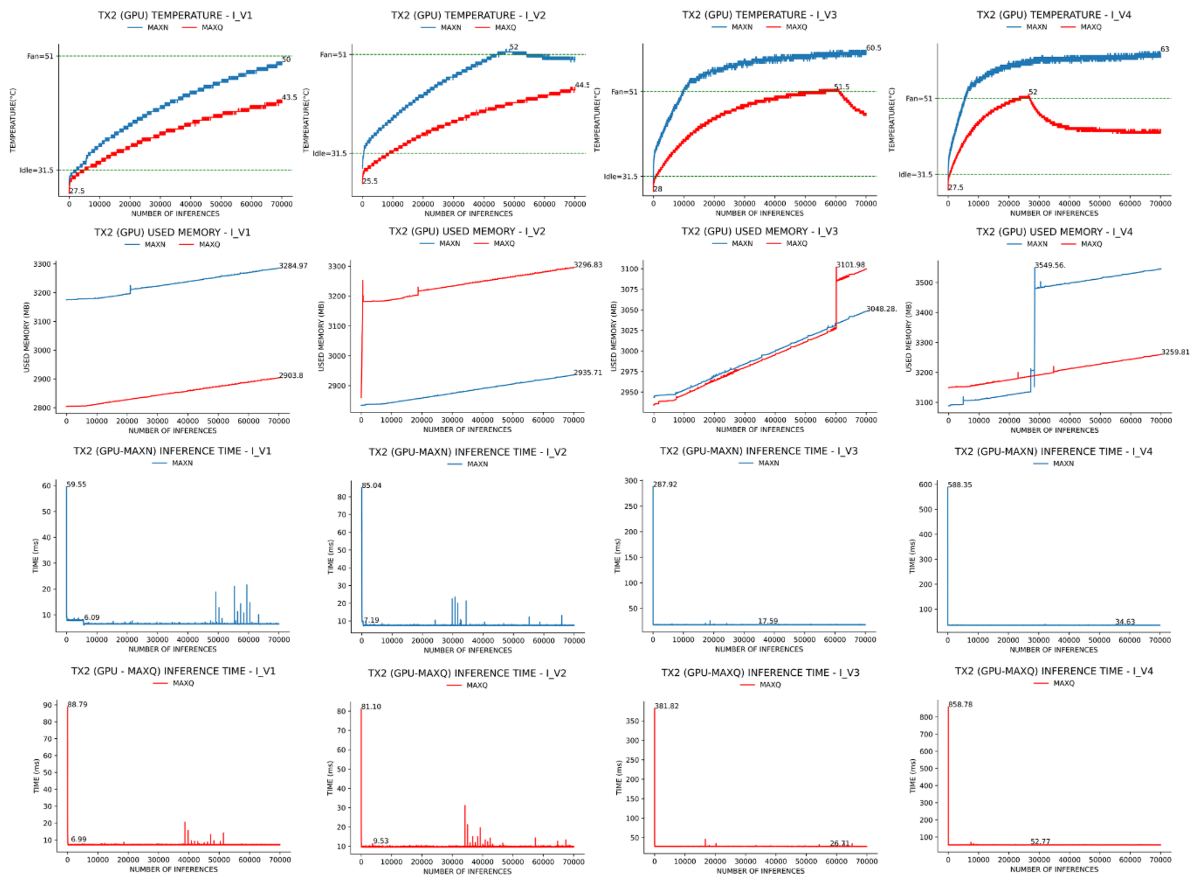

Fig. 4. TX2's evaluation metrics.

Likewise Coral, in the inference figures, it is observed that in all models the first inference is the slowest one. Also, Coral's first inference is by far faster than TX2. A possible explanation for this is that it is an ASIC and therefore it has a favorable memory hierarchy. Meanwhile, the TX2 has more data movement to do. But there is chance to reduce this timing because this devboard has a shared memory between the CPU and GPU which has not been used in this experiment because of some software limitations.

Finally, when comparing MAXN and MAXQ, it is observed that in general the former is faster because of the clock's frequency. Also I_v3 and I_v4 display less inference peaks and this is related to having a stable temperature which can be seen in these 2 models.

Table 2. Metrics.

Models (KB)	Metrics	Coral	TX2	
		MAX	MAXQ	MAXN
I_v1 (7.037)	Latency (ms)	17.97	81.58	53.11
	Score	0.64	0.65	0.65
I_v2 (12.165)	Latency (ms)	18.98	71.36	77.55
	Score	0.96	0.92	0.92
I_v3 (24.401)	Latency (ms)	17.33	354.5	270.02
	Score	1	0.99	0.99
I_v4 (43.865)	Latency (ms)	20.7	805.13	552.89
	Score	1	0.97	0.97

Fig. 5 displays the total amount of time for executing the 70000 inferences. It is observed that both devboards have used more time in I_v4 which effectively is the biggest model. But it is interesting to see that Coral, which has more compression (INT8) and is an spatial architecture, takes more time to finish the whole round of inferences than the TX2 (MAXQ and MAXN) in all models except for I_v1. This is because it uses a reactive control. By default Coral is at maximum performance all the time (500MHz) and whenever the TPU's temperature reaches the fan trip point (65°C), its clock frequency will decrease. This temperature fluctuation around the trip point affects the voltage-frequency operating point and over a sustained period of time has a negative impact on each inference as it makes their execution time more unpredictable. This can also be confirmed in Fig. 6, as once again it is seen that each inference of Coral takes more time than the TX2 in all models except for I_v1 which is the smallest one.

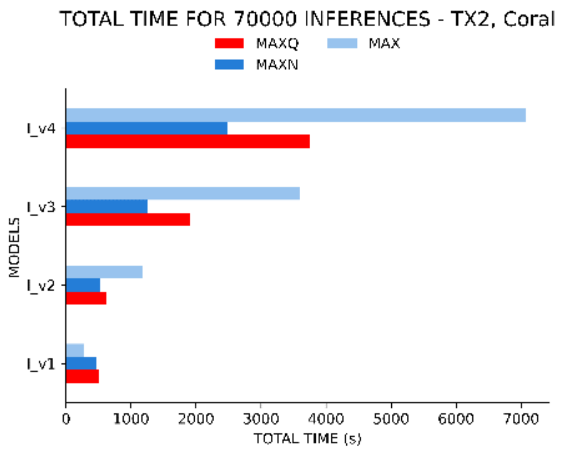

Fig. 5. Total time for 70000 inferences – TX2, Coral.

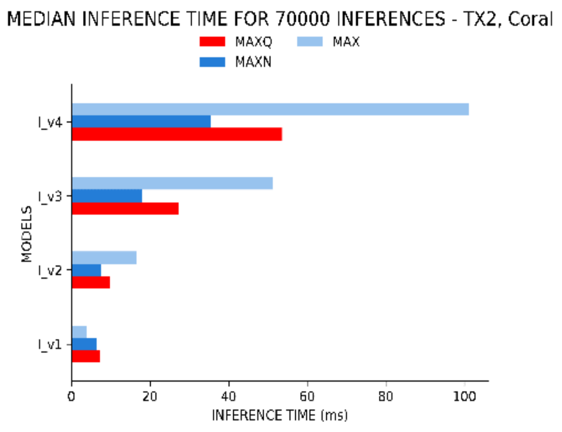

Fig. 6. Median inference time – TX2, Coral.

Coral only performs better than TX2 (MAXQ and MAXN) in Iv1 because it is the only model where it never throttles. And this proves the importance of optimizing DVFS control. Also, in both figures it is observed that the mode MAXQ takes more time than MAXN in all models. This is because it uses a lower frequency clock (from 114 MHz to 850MHZ).

4. Experimental Protocol

The experimental setup, shown in Fig. 7, includes an integrated solution, Mind in a Box (M/B), to support real time collection of the sensing data (thermal, cooling system, …) and the computational load in Edge devices such as the two target architectures, the Google Coral (TPU) and Nvidia Jetson (GPU). This architecture, processing this information, can then prescribe load balancing and computational resources tuning. Thus optimizing the global performance of hybrid architectures leveraging Cloud, Edge and Fog Computing capabilities.

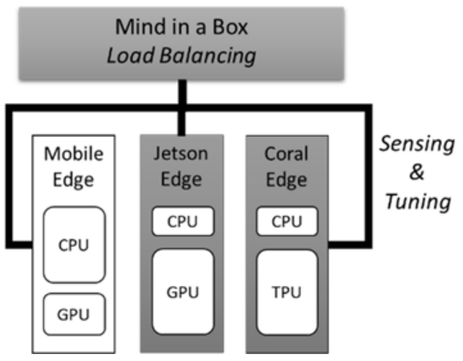

Fig. 7. Edge Computing Setup, Experimental Protocol.

This setup is typical of new Hybrid Analytics and Hybrid AI applications where Edge devices take charge of a sizeable workload in terms of local automatic processing and classification on the field, whether it is to process local information or to propose additional processing capabilities available to the architecture as a whole.

In this context, since the cost of ownership of such devices is relatively limited and their numbers can become massive, the question of optimizing performances and balancing workloads becomes an essential component which requires both current work load, multisensing consumption, and control over each Edge device computational load and hardware parametrization (such as a process unit's frequency, for instance). This experimental protocol aims at producing results which would be applicable in comparable Hybrid & Edge Computing architectures.

5. Conclusions

In this abstract, Google's Coral (TPU) and Nvidia's Jetson TX2 (GPU) edge accelerators have been compared from the angle of hardware architecture (processors, thermal regulation, etc) and software architecture (frameworks, workflow, etc) employing the multiple versions of GoogleNet (image classification) and assessing the quality of DVFS. Results suggest a great potential for further improvement in thermal regulation for the TPU. Even though it has an advantage in hardware architecture and model compression over the GPU, its performance is dismissed because of the reactive control which degrades the inference time.

References

[1]. A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, *Commun. ACM*, 60, 6, 2017.

[2]. O. Russakovsky *et al.*, ImageNet Large Scale Visual Recognition Challenge, *Int. J. Comput. Vis.*, Vol. 115, No. 3, 2015, pp. 211–252.

[3]. V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, Efficient Processing of Deep Neural Networks: A Tutorial and Survey, in *Proceedings of the IEEE*, 2017, pp. 2295–2329.

[4]. M. Horowitz, Computing's energy problem (and what we can do about it), in *Digest of Technical Papers (ISSCC) - IEEE International Solid-State Circuits Conference*, 2014, pp. 10–14.

[5]. S. Li, L. Da Xu, and S. Zhao, The internet of things: a survey, *Inf. Syst. Front.*, Vol. 17, No. 2, 2015, pp. 243–259.

[6]. M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, Machine Learning at the Network Edge: A Survey, Jul. 2019, https://arxiv.org/abs/1908.00080.

[7]. Z. Tang, Y. Wang, Q. Wang, and X. Chu, The impact of GPU DVFS on the energy and performance of deep Learning: An Empirical Study, in - *Proceedings of the 10th ACM International Conference on Future Energy Systems e-Energy'2019*, 2019, pp. 315–325.

[8]. A. K. Singh, S. Dey, K. R. Basireddy, K. McDonald-Maier, G. V. Merrett, and B. M. Al-Hashimi, Dynamic Energy and Thermal Management of Multi-Core Mobile Platforms: A Survey, *IEEE Des. Test*, Vol. 37, No. 5, pp. 25–33, 2020.

[9]. Google, Coral dev board, 2020. https://coral.ai/products/dev-board (accessed Feb. 21, 2020).

[10]. Nvidia, Jetson TX2 module, 2020. https://developer.nvidia.com/embedded/jetson-tx2 (accessed Feb. 21, 2020).

[11]. R. Hadidi, J. Cao, Y. Xie, B. Asgari, T. Krishna, and H. Kim, Characterizing the Deployment of Deep Neural Networks on Commercial Edge Devices, in *Proceedings of the 2019 IEEE International Symposium on Workload Characterization (IISWC' 2019)*, 2019, pp. 35–48.

[12]. V. Peluso, R. G. Rizzo, and A. Calimera, Performance profiling of embedded convnets under thermal-aware DVFS, *Electron.*, Vol. 8, No. 12, p. 1423, 2019.

[13]. TensorFlow Zoo. https://github.com/tensorflow/models/tree/master/research/slim