electronics

Article

Performance Profiling of Embedded ConvNets under Thermal-Aware DVFS

Valentino Peluso, Roberto Giorgio Rizzo and Andrea Calimera *

Department of Control and Computer Engineering, Politecnico di Torino, 10129 Torino, Italy;
valentino.peluso@polito.it (V.P.); robertogiorgio.rizzo@polito.it (R.G.R.)
* Correspondence: andrea.calimera@polito.it

Received: 10 November 2019; Accepted: 26 November 2019; Published: 29 November 2019

Abstract: Convolutional Neural Networks (ConvNets) can be shrunk to fit embedded CPUs adopted on mobile end-nodes, like smartphones or drones. The deployment onto such devices encompasses several algorithmic level optimizations, e.g., topology restructuring, pruning, and quantization, that reduce the complexity of the network, ensuring less resource usage and hence higher speed. Several studies revealed remarkable performance, paving the way towards real-time inference on low power cores. However, continuous execution at maximum speed is quite unrealistic due to a fast increase of the on-chip temperature. Indeed, proper thermal management is paramount to guarantee silicon reliability and a safe user experience. Power management schemes, like voltage lowering and frequency scaling, are common knobs to control the thermal stability. Obviously, this implies a performance degradation, often not considered during the training and optimization stages. The objective of this work is to present the performance assessment of embedded ConvNets under thermal management. Our study covers the behavior of two control policies, namely reactive and proactive, implemented through the Dynamic Voltage-Frequency Scaling (DVFS) mechanism available on commercial embedded CPUs. As benchmarks, we used four state-of-the-art ConvNets for computer vision flashed into the ARM Cortex-A15 CPU. With the collected results, we aim to show the existing temperature-performance trade-off and give a more realistic analysis of the maximum performance achievable. Moreover, we empirically demonstrate the strict relationship between the on-chip thermal behavior and the hyper-parameters of the ConvNet, revealing optimization margins for a thermal-aware design of neural network layers.

Keywords: deep learning; convolutional neural network; dynamic voltage and frequency scaling; thermal management; continuous inference; edge computing

1. Introduction

Recent advancements in the field of deep learning theories and applications have enabled the deployment of Convolutional Neural Networks (ConvNets) on the edge, namely on resource constrained embedded systems. This opened new scenarios of applications where low power end-nodes can make sense of the sampled data and understand the surrounding context with limited energy budgets [1]. The shift to the edge has several implications and advantages for the growth of a sustainable Internet-of-Things (IoT) ecosystem: (i) guarantee user-privacy as data stay local; (ii) improve the whole energy efficiency as data transfers from/to the cloud become low; (iii) reduce the time-of-flight and make the service response predictable. These aspects are paramount for always-on applications that run real-time continuous inference over consecutive frames of data. Some examples are context sensing [2], health monitoring [3], and object tracking [4], for which ceaseless uploading of data in the cloud would be impractical due to high energy consumption and long/uncertain latency.

General-purpose CPUs are the most common design choice for embedded systems today. Millions of chip-sets powered by Reduced Instruction Set Cores (RISCs) are already in the field (e.g., in mobile phones) and can be employed for inference with a simple software update [5]. Unfortunately, ConvNets are complex and resource hungry models. To accelerate their processing and achieve reasonable performance even on low power budgets, there are several design actions to take. At the hardware level, a common practice is to adopt heterogeneous architectures that jointly integrate high-end multi-core CPUs with arithmetic accelerators. For instance, the ARM Cortex-A15 of the Samsung Exynos 5422 system-on-chip (SoC) [6] is a four-core CPU that can run up to 2.0 GHz; each core hosts the NEON unit [7], a Single-Instruction Multiple-Data (SIMD) data-path for parallel arithmetic. At the software level, algorithmic optimizations can be applied to reduce the cardinality of ConvNets. Among the many available, the most effective techniques include topology restructuring [8], pruning [9], quantization [10] (e.g., 8 bit fixed-point), or a mix of them [11]. When properly implemented, these methods reduce the memory footprint and the number of operations to run, helping to achieve higher execution speed and less energy consumption. Moreover, specialized routines for tensor operators (e.g., multi-dimensional matrix convolutions) are used at compile time to build highly efficient executable code [12].

As a result of the optimization chain, ConvNets become super-dense workloads that flood the hardware, saturating both memory and CPU utilization. While this may seem positive in terms of efficiency, it represents a serious concern over long execution intervals, when thermal issues arise affecting the reliability. This aspect is neglected in modern deep learning optimization frameworks. Since high-end cores are integrated into embedded devices with a small form factor, high utilization rates come at the cost of much higher power density, which in turn generates more heat than cooling systems can dissipate. The on-chip temperature increases quickly, reaching critical values even in short time windows. High temperatures do activate several degradation mechanisms that undermine the lifetime of the device [13] or affect the user experience. It is therefore paramount to adopt a control mechanism to prevent thermal runaway.

In its most general embodiment, the strategy for thermal control is to make active cores switching into a low power mode as soon as they reach a critical temperature. With lower power densities, the thermal equilibrium is restored, and temperatures cool down. Needless to say, thermal controls alter the CPUs speed, impairing the performance estimated at design time. As will be shown in the paper, state-of-the-art ConvNets, both accuracy optimized ConvNets (e.g., Inception [14]) and performance optimized ConvNets (e.g., MobileNets [15]), reach safety-critical temperature just after 1–3 s of continuous inference, making the average performance gap quite large. Neglecting this aspect may have dramatic impacts on the dependability of the system, causing functional failures in the worst case. Instead, a thermal-conscious deployment of ConvNets would help to improve several figures of merit. While thermal issues and thermal-aware hardware/software co-design are well established topics in the literature, the intersection with ConvNets is a less explored field. An in-depth analysis may reveal interesting trends with new insights for ConvNets' optimization. This is precisely the scope of this paper, which proposes a performance assessment of thermally managed ConvNets implemented into low power CPUs for mobile applications.

Looking at practical implementations, supply-voltage lowering is an effective knob to meet Thermal Design Power (TDP) constraints. If combined with frequency scaling, it enables a cubic reduction of dynamic power consumption. Moreover, lower voltages decrease the static power. Embedded CPUs come with integrated Dynamic Voltage and Frequency Scaling (DVFS) mechanisms offering a wide range of operating points in the power-temperature-performance space. Thermal governors can implement temperature driven DVFS to maximize performance within the available TDP budget. Identifying the best control policy is an interesting problem that has been extensively addressed in the literature. While sophisticated schemes based on workload prediction and/or temperature speculation are currently available [16], ConvNets are static graphs with data independent workloads. This offers a unique chance to profile the thermal-vs.-performance behavior at design-time. We thereby built an automatic framework that supports multiple ConvNet models allowing a parametric analysis over different use cases. The experiments, conducted over four state-of-the-art ConvNets for computer vision tasks deployed on an Odroid-XU4 board [17] powered by the ARM Cortex-A15 core, enables the following key achievements:

- Quantify the thermal headroom of ConvNets deployed for continuous inference. Our analysis identifies applications that can be critical for power constrained devices.
- Assess the performance of ConvNets under thermal-aware DVFS. The experiments cover two control policies, namely reactive and proactive.
- Identify the optimal operating points of voltage scaled ConvNets. The analysis provides useful guidelines to develop smarter control policies specialized for ConvNets.
- Demonstrate that the thermal profile of ConvNets depends on the network topology. The collected results reveal the need for new optimization techniques for training thermal-aware ConvNets.

The remainder of the paper is organized as follows: Section 2 reviews previous works. Section 3 summarizes the most common thermal management mechanisms adopted in off-the-shelf embedded systems. Section 4 introduces the characterization framework. Section 5 shows the experimental results and their analysis. Finally, Section 6 concludes the work.

2. Related Works

The assessment of ConvNets performance on embedded systems has been the target of extensive studies whose main objective was to quantify the gap with cloud services.The authors in [18] compared the performance of ConvNets running on workstations against those deployed onto high-end embedded systems in order to evaluate the portability of existing training frameworks. A similar analysis was performed in [19] comparing cloud platforms against edge devices, both embedded CPUs and GPUs. The authors in [20] estimated the performance boost achieved by custom accelerators for the mobile segment with respect to embedded CPUs. Although they guarantee more stable temperature and power consumption, accelerators are available only on a small fraction of existing embedded systems. Furthermore, they still lack stable software support, thus preventing the deployment of the most advanced ConvNet architectures. The investigation conducted in [21] evaluated the compatibility of state-of-the-art networks across different off-the-shelf components to prove that optimal deployment depends on the underlying hardware. Finally, the study in [22] showed that increasing the batch size during inference enabled a trade-off between latency and throughput. Once again, this work confirms that the optimal configuration is strictly related to the target hardware, specifically with the degree of parallelism offered by the arithmetic units.

Similar to previous works, our study focuses on the performance profiling of ConvNets running on high-end embedded CPUs. However, while previous works assessed inference time only in nominal conditions, i.e., single inference at maximum frequency, we provide a parametric analysis through an in-house characterization framework that integrates the effects of thermal management via voltage lowering. Our experiments prove that designers should carefully assess the thermal and power constraints of the hosting hardware to avoid mismatches between expected performance (at design-time) and run-time execution. The conducted analysis provides useful guidelines for future thermal-aware ConvNet optimization, such as neural network compression [23] and neural architecture search [24], which may exploit the power-thermal characterization of the target hardware.

3. Background

3.1. Thermal Management Strategies

Thermal management strategies change the operating point of the system to reduce the power consumption and control the on-chip temperature. Among the available options, DVFS represents one of the most effective knobs since the active power consumption shows a quadratic dependence on voltage and a linear dependence on frequency. Custom power distribution schemes, e.g., [25], can push the efficiency of DVFS even beyond these theoretical relationships.

Commercial CPUs offer a standard set of voltage and frequency (VF) levels (19 in the Cortex-A15), which enable a fine grained control on power and performance. As will be discussed in Section 5.3, we observed that other knobs are less efficient for controlling temperature during continuous inference. Each VF level identifies a specific operating point in the power-performance space. The maximum performance can be achieved using the highest voltage and the maximum frequency available, which we refer to as VF_{max}; within the Cortex-A15, VF_{max} = 1.3625 V @ 2 GHz. Changing the operating point at run-time enables managing the power-performance trade-off, which means controlling the temperature profile at the expense of some latency penalty.

An efficient management policy aims to guarantee thermal stability with minimum speed degradation. Off-the-shelf SoCs implements a reactive thermal management mechanism. To meet high computational demands, the active cores operate at VF_{max} and invoke a safety mechanism, thermal throttling, that reduces the VF level when the temperature reaches a critical threshold, thus preventing the processor, and the whole device, from overheating. For instance, the Cortex-A15 CPU down-scales the voltage-frequency level from VF_{max} to VF_{low} = 0.8875 V @ 900 MHz when the temperature exceeds T_{max} = 90 °C. A qualitative analysis of this strategy is depicted in Figure 1.

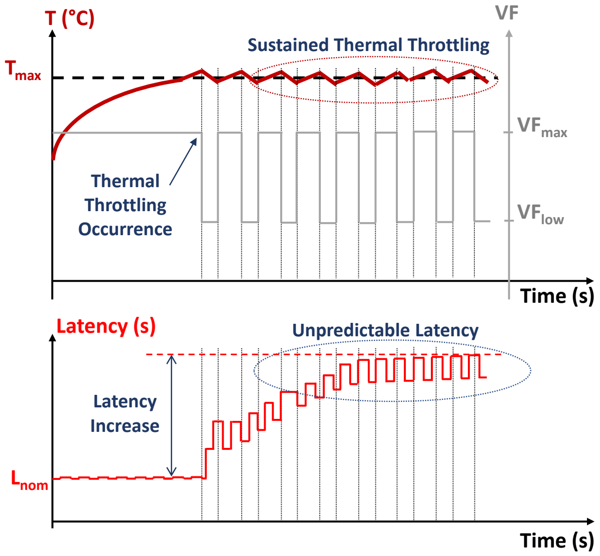


Figure 1. Qualitative analysis of temperature (**above**) and inference latency (**below**) evolution over time under reactive thermal management.

Under intensive workloads, like those of ConvNets, this mechanism may lead to significant performance degradation, especially when continuous inference is held for long time intervals. As shown in the top plot of Figure 1, running the cores at maximum performance pushes the temperature towards the critical threshold T_{max} and forces the SoC to throttle the performance of the cores switching from the high performance state VF_{max} to the low power state VF_{low}. As soon as the temperature falls below T_{max}, the SoC switches back to VF_{max}, forcing another invocation of thermal throttling in a very short time; the sequence repeats ceaselessly till the task ends. This working mode is called sustained thermal throttling: the temperature fluctuates around the safety threshold over a sustained period, and so does the voltage-frequency operating point, which moves up and down between VF_{max} and VF_{low}. As shown in the bottom plot of Figure 1, this has a negative impact on latency: (i) working at VF_{low} introduces an overhead with respect to the nominal latency L_{nom}; (ii) the cyclic swapping from high performance (VF_{max}) to low power (VF_{low}) modes makes the latency less predictable. For these reasons, reactive strategies turn out to be quite inefficient. In the specific case of continuous inference, we measured a latency overhead ranging from 30% to 43% depending on the ConvNet, together with an increase of variability up to 70× (see Section 5.3 for more details).

Proactive thermal management represents a more efficient alternative. It works ahead of time as it enacts a more stable voltage lowering before the temperature reaches critical limits. More precisely, the CPU is made to work at an intermediate operating point VF_{opt} between VF_{max} and VF_{low} from the beginning. The benefits are qualitatively shown in Figure 2, which provides a comparison against the reactive strategy. While the proactive approach introduces some performance overhead on the very first short term ($L_{\text{opt}} > L_{\text{nom}}$), it ensures substantial gains in the long term because it prevents the occurrence of the throttling events. Overall, the average latency improves, while the predictability is guaranteed for much longer. Under highly demanding workloads of a very long duration, the temperature might reach critical values even with a proactive control, and hence, thermal throttling (from VF_{opt} to VF_{low}) may still occur. However, its occurrence is less frequent.

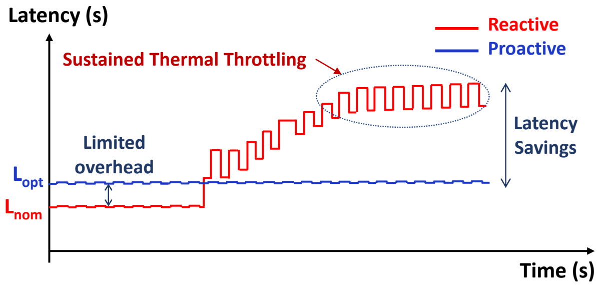


Figure 2. Inference latency in reactive (red) and proactive (blue) thermal management.

3.2. Optimal Trade-Off

For a given maximum temperature, there exists an optimal trade-off between power (i.e., the VF level) and performance. The plot reported in Figure 3 gives a graphical representation of such an optimality problem. It shows the average latency (L_{avg}) for different VF levels considering a pre-defined sequence of N inference runs. On the right side of the minimum latency point, achieved working at VF_{opt}, it happens that frequent throttling events induce performance penalty; on the left side of VF_{opt}, thermal throttling does not occur often, but latency increases due to a too conservative voltage-frequency scaling. The precise position of VF_{opt} is a function of the total active time, i.e., the number of inference runs N, and the topology of ConvNet (size, number of operations, and memory allocation).

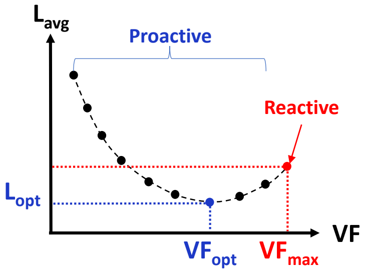


Figure 3. Average latency under different thermal management strategies.

3.3. Proactive Control Policies

Previous works conducted extensive study on proactive thermal management on embedded systems. They presented control policies that aimed to identify the optimal operating point of the system according to the current workload. An exhaustive taxonomy of the existing techniques can be found in [16].

The work described in [26] was a pioneer in this field. The authors proposed a closed-loop controller that adjusts the voltage and frequency level to reduce the error between the expected and measured performance. More advanced controllers make use of regression models to predict the future temperature and identify the operating point that achieves maximum performance, yet avoiding thermal violation. The model can be trained off-line on a set of representative benchmarks [27] or it can be continuously updated at run-time [28,29].

Motivated by the observation that ConvNets are static graphs that always execute the same flow of operations, we propose a characterization framework that enables extracting the thermal profile of a given ConvNet at design time. Rather than proposing a novel controller, this work aims to quantify the performance of ConvNets in a power/thermal constrained environment and to identify the best operating points for thermal management during continuous inference.

4. Thermal-Aware Performance Optimization and Characterization Framework

The problem of finding the optimal operating point of continuous-inference applications under proactive thermal management can be formulated as follows: given a pre-trained ConvNet, deployed on a given embedded CPU, and made to run for a fixed number of inferences N, find the voltage-frequency operating point VF_{opt} that minimizes the average latency. Considering the relatively low cardinality of the solution space, we opted for an exhaustive exploration conducted through the characterization framework shown in Figure 4.

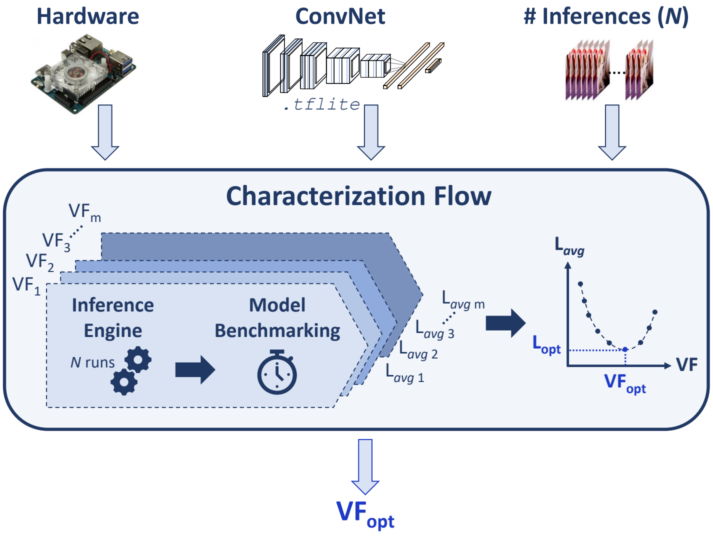


Figure 4. Schematic view of the proposed characterization flow.

The framework consists of two main components: (i) an inference engine that runs the ConvNet workload and (ii) a benchmarking tool that is in charge of assessing the performance, i.e., the inference latency. The inference engine is based on TensorFlow Lite (TFL) by Google, i.e., a collection of software routines for deep learning highly optimized to run tensor graphs on multi-core processors integrating an SIMD data-path. Furthermore, TFL integrates a benchmarking utility, called the TensorFlow Lite Model Benchmark, that allows the measurement of the inference time on the target device by randomly assembling inputs from the dataset. The tool collects several statistics recorded on-board, in particular the average latency and the standard deviation over multiple runs.

The frameworks are fed with three main inputs: (i) a ConvNet architecture in `tflite` format; (ii) the number of continuous inferences N; (iii) the specifications of the device that hosts the ConvNet (i.e., the available VF levels). The framework is compiled and executed on the specified hardware to collect the average latency over N continuous inferences run for all available VF points. The main outcome is the minimum latency value L_{opt} and the corresponding optimal operating point VF_{opt}.

5. Experimental Setup and Results

The objective of the analysis reported in this section is threefold: (i) understand when continuous inference generates temperature violations; (ii) quantify the actual performance of ConvNets under reactive/proactive thermal management and different network architectures; (iii) identify the optimal operating points for voltage scaled ConvNets to guide the development of smart control policies oriented toward neural tasks. The contents are organized as follows. First, we describe the hardware board used in the experiments along with the software environment adopted for the deployment. Second, we introduce the ConvNets taken as benchmarks. Finally, we report the collected results and discuss the key insights.

5.1. Hardware Platform and Software Configurations

The hardware test bench was the Odroid-XU4 platform powered with the Samsung Exynos 5422 SoC. The CPU was an ARM Cortex-A15, which integrates four cores that can work at VF_{max} = 1.3625 V @ 2 GHz in nominal conditions. The board runs Ubuntu Mate 16.04, kernel Version 3.10.106-154, released by Hardkernel. The standard thermal governor (reactive) scales the operating point of the A15 cores to VF_{low} = 0.8875 V @ 900 MHz as soon as the temperature exceeds the threshold $T_{\text{max}} = 90\,^{\circ}\text{C}$. The kernel offers 19 voltage and frequency levels with a step of 100 MHz (the minimum frequency is 200 MHz). For the sake of simplicity, we denote the VF operating points just using the frequency value (in GHz). The board was cooled with an active fan controlled by pulse-width modulation (PWM); all the experiments were run at a constant fan speed of 36%. Unless explicitly specified, collected measurements refer to four thread execution.

The inference engine was TensorFlow Lite 1.14; it offers a collection of neural network routines optimized to run on the ARM Cortex-A architecture. Specifically, the convolutional operators make use of SIMD instructions to leverage the parallelism offered by the NEON unit [10]. In our setup, TensorFlow Lite was cross-compiled using the GNU ARM Embedded Toolchain (Version 6.5) [30].

5.2. ConvNet Benchmarks

The adopted ConvNets were picked from the TensorFlow Hosted Models [31] repository. In particular, we used two representative types of models: MobileNet and Inception. For each model, we investigated two different versions for a total of four ConvNets; their features are summarized in Table 1. Notice that all the ConvNets are converted using an 8 bit fixed-point representation, a common choice for edge inference as it ensures lower memory footprint and better performance with negligible accuracy loss with respect to the floating-point. The column Memory collects the size of the tflite, which contains the data structures needed to deploy the model on-chip, i.e., the network weights and the topology description. The column Top-1 refers to the top-1 classification accuracy measured on the ImageNet validation set. The column L_{nom} reports the nominal latency, that is the one obtained under maximum performance (VF_{max}). The reported numbers refer to the average over 100 inference runs, each of them interleaved by a two-second pause to avoid temperature variations of the chip. The column σ_{nom} reports the standard deviation of the nominal latency measured over 100 runs.

Table 1. Memory, accuracy, and nominal latency of the selected benchmarks.

ConvNet	Memory (MB)	Top-1 (%)	L_{nom} (ms)	σ_{nom} (ms)
MobileNet v1	4.3	70.0	31.99	0.06
MobileNet v2	3.4	70.8	30.24	0.06
Inception v1	6.4	70.1	87.84	0.13
Inception v4	41.0	79.5	658.06	0.57

MobileNets are compact networks optimized for high performance on embedded applications. Inception models are designed to achieve higher accuracy; therefore, they have a more complex architecture that requires more computational resources. Inception v4 guarantees 8.7% higher accuracy than MobileNet v2 at the cost of 12× more memory and 22× higher latency.

5.3. Results

Thermal headroom in continuous inference: Table 2 reports the number of continuous inferences N_{safe} and the execution time t_{safe} at VF_{max} = 2.0 GHz before the temperature exceeds the critical threshold T_{max} = 90 °C. For all the ConvNets, the critical threshold was reached after a low number of inferences, e.g., only four in Inception v4. This motivated the need for thermal management.

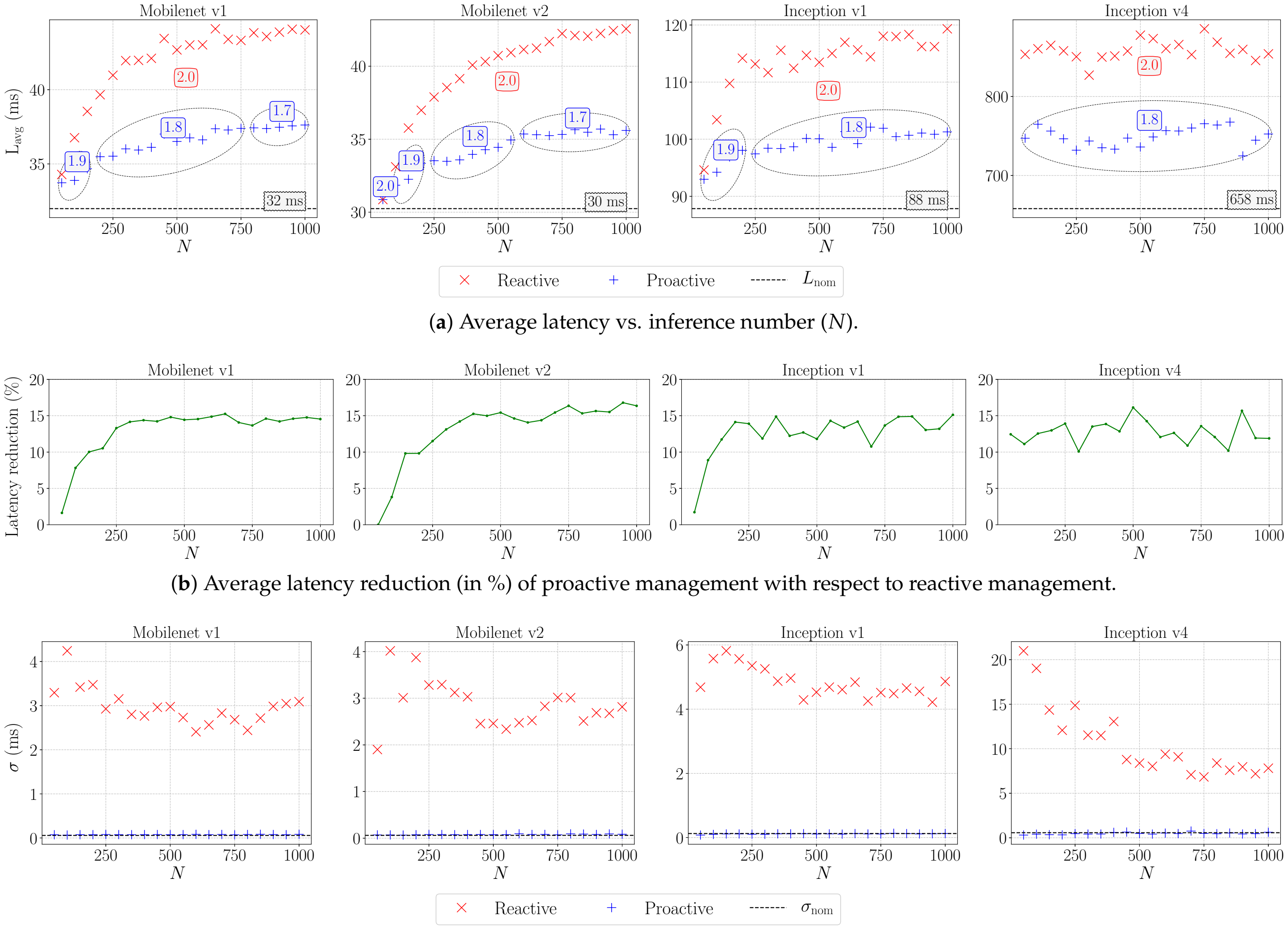
It is possible to observe different trends across the selected benchmarks. For instance, Inception v4 presented the first thermal throttling event 2.3× later than MobileNets v2 (2.93 s vs. 1.27 s). This finding suggests that the thermal gradient strictly depends on the topology of ConvNets, which is quite intuitive as different models come with a different number of layers of different sizes and cardinality. Obviously, the smaller the net, the larger the number of inferences run within t_{safe}.

Table 2. Thermal headroom of different ConvNets in continuous inference. N_{safe} and t_{safe} are the maximum number of consecutive inferences and the execution time at safe temperature values (i.e., $T < T_{\text{max}}$).

ConvNet	N_{safe}	t_{safe} **(s)**
MobileNet v1	39	1.26
MobileNet v2	42	1.27
Inception v1	25	2.21
Inception v4	4	2.93

Performance under thermal management: A more interesting analysis concerns the performance gap between reactive thermal management (i.e., working at VF_{max}) and proactive thermal management (i.e., working at VF_{opt}). For proactive, the optimal level VF_{opt} is extracted from the proposed characterization framework (see Section 4). The framework runs a continuous number of inferences N, with N ranging from 50 to 1000 with a step of 50 inferences. Figure 5 reports the collected results for all the benchmarks.

The plots in Figure 5a show the average latency as a function of N; the red "×" marker refers to reactive. whereas the blue "+" marker is for proactive. The black dashed line quantifies the nominal latency L_{nom} (the exact value is also reported in the label). In both cases, thermal management produces a performance overhead with respect to L_{nom}. As expected, proactive management outperforms reactive management as it mitigates the occurrence of thermal throttling. For longer execution time, i.e., larger N, the level of VF_{opt} scales down (value reported in the blue boxes, in GHz), as larger thermal headroom is needed to ensure safety. Again, the performance analysis reveals different trends depending on the ConvNet topology. First, MobileNets showed a higher performance overhead with respect to L_{nom} than Inception nets when running in continuous inference. In the worst case, the overhead was 43% and 30% for MobileNet v2 and Inception v4, respectively. Second, in proactive management, the value of VF_{opt} varied with N, but also with the kind of network. For example, in MobileNets, VF_{opt} scaled down to 1.7 GHz, whereas in Inception nets, the minimum value was 1.8 GHz.



(**a**) Average latency vs. inference number (N).

(**b**) Average latency reduction (in %) of proactive management with respect to reactive management.

(**c**) Latency standard deviation (σ) over N inferences.

Figure 5. Results of the characterization flow.

Figure 5b gives a more detailed analysis of the performance gains achieved with proactive management. Concerning the MobileNets, the savings against reactive increased up to 15.3% and 16.8% for v1 and v2, respectively; for the Inception nets, the performance savings were greater than 10% for $N > 100$, with peaks of 15.1% and 16.2% for v1 and v4, respectively.

Proactive management also guaranteed lower latency variability. This is shown in Figure 5c. The plots report the standard deviation σ measured at different N. For all the benchmarks, a proactive strategy kept σ close to the variability measured at nominal conditions (depicted by the black dashed line). These findings demonstrate that proactive management enables a more efficient and reliable neural task scheduling.

Topology impact on temperature: To better analyze the impact of different topologies on the temperature gradient, we collected the chip temperature with a sampling rate of 10 ms over an interval time of 100 s. Figure 6 (top) reveals that the Inception nets had a slower temperature gradient than that of the MobileNets, both in reactive and proactive management. This suggests there is room for power- and thermal-aware design of ConvNet operators that might help to achieve higher thermal stability. Current research trends in deep learning do not consider this aspect as the design of ConvNets is mainly driven by performance and energy optimization in nominal conditions, which is misleading indeed. Furthermore, we measured the latency of each inference over the same activity time in order to highlight the variations resulting from thermal management. The analysis is reported in Figure 6 (down). In reactive management (red line), the latency quickly increased, both in terms of absolute values and variability, as soon as the SoC entered the sustained thermal throttling regime. This condition held for all the benchmarks. On a long term period, proactive management (blue line) outperformed the reactive approach by far, ensuring better performance on average with a higher degree of stability. As expected, the latency under pro-active management became worse just in a very short term window at the beginning, when the cores were cold.

The bar plot in Figure 7 quantifies the percentage of time during which the cores were pushed to work in the low power mode (VF_{low}) due to the occurrence of thermal throttling. The percentage was much lower for proactive management indeed. Interestingly, Inception nets were less prone to thermal protection, even if their size is larger than MobileNets. Under proactive management, Inception v1 spent 277× less time in throttling than MobileNet v1, even if working at a higher VF level (1.8 GHz vs. 1.7 GHz, as shown in Figure 6 down). Overall, proactive management can be appreciated as an effective strategy to reduce the throttling time (2.77% as the worst case).

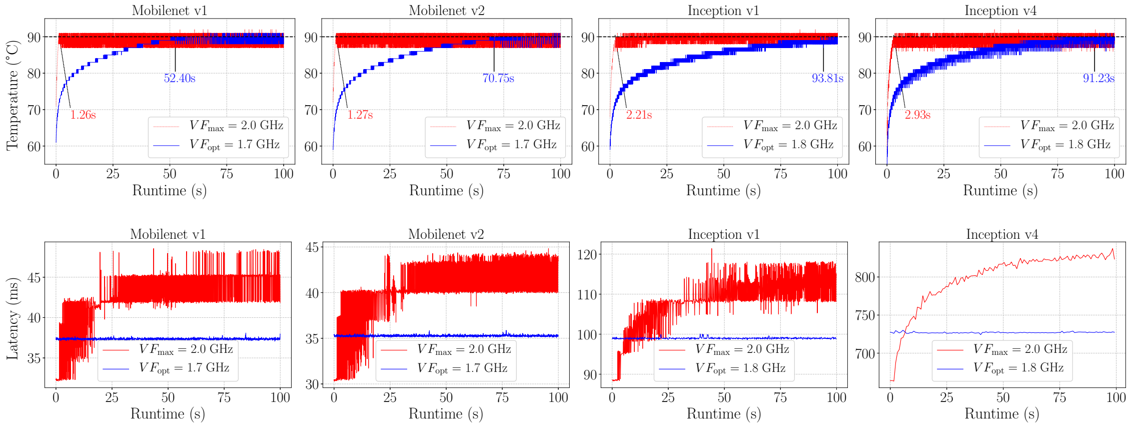


Figure 6. Temperature gradient (**top**) and inference latency (**down**) of continuous inference for 100 s. The annotations reports the occurrence of the first thermal throttling event.

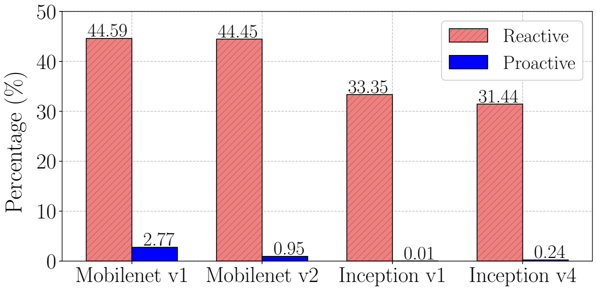


Figure 7. Percentage of execution time at VF_{low} = 900 MHz over a runtime of 100 s.

Efficiency of DVFS: Multi-core CPUs offer another power knob to control the thermal state of the chip: Dynamic Power Management (DPM) via core shutdown. This strategy adjusts the number of active cores to limit peak power consumption. Given the parallel nature of ConvNets, reducing the number of processing elements could have a severe impact on performance. Instead, supply voltage reduction enables a finer control on the power-performance trade-off. Table 3 provides empirical evidence of this observation. For each benchmark, the table reports the nominal latency of three thread execution (column $L_{\text{nom-3}}$) and the worst case latency measured under DVFS based proactive management at four thread execution (column $L_{\text{opt-wc}}$). Even in nominal conditions, i.e., at VF_{max} w/o thermal throttling, DPM had lower performance than thermal-aware DVFS ($L_{\text{nom-3}} > L_{\text{opt-wc}}$ in all cases). The collected results demonstrate that operating at low voltage is the most effective solution for the thermal management of ConvNets.

Table 3. Nominal inference latency at 3 thread execution vs. worst case latency under DVFS based proactive management at 4 thread execution.

ConvNet	$L_{\text{nom-3}}$ (ms)	$L_{\text{opt-wc}}$ (ms)
MobileNet v1	41	38
MobileNet v2	38	36
Inception v1	103	102
Inception v4	770	768

6. Conclusions

In this work, we introduced a parametric characterization of the performance of thermally managed ConvNets deployed on embedded CPUs. The study assessed the quality of thermal protection using DVFS, under both reactive and proactive schemes. The collected results provide useful guidelines for hardware-aware ConvNet optimization since they reveal a significant mismatch between nominal and actual performance in power constrained systems that need thermal management to meet the TDP constraints. A detailed analysis demonstrated that proactive management via DVFS could help to reduce the performance overhead and guarantee low performance variability. Finally, our research opens a new optimization margin for the design of thermal-aware neural network operators as it empirically demonstrated the existence of different thermal mismatches, which depend on the internal structure of the ConvNets. There might exist networks with the same memory footprint and latency, but different thermal profiles. Finding those configurations could help to improve the efficiency of continuous inference tasks.

Author Contributions: All the authors listed in the first page made substantial contributions to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. LiKamWa, R.; Priyantha, B.; Philipose, M.; Zhong, L.; Bahl, P. Energy characterization and optimization of image sensing toward continuous mobile vision. In Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services, Taipei, Taiwan, 25–28 June 2013; ACM: New York, NY, USA, 2013; pp. 69–82.
2. Seidenari, L.; Baecchi, C.; Uricchio, T.; Ferracani, A.; Bertini, M.; Bimbo, A.D. Deep artwork detection and retrieval for automatic context-aware audio guides. *ACM Trans. Multimedia Comput. Commun. Appl.* **2017**, *13*, 35. [CrossRef]
3. Wang, A.; Chen, G.; Yang, J.; Zhao, S.; Chang, C.Y. A comparative study on human activity recognition using inertial sensors in a smartphone. *IEEE Sens. J.* **2016**, *16*, 4566–4578. [CrossRef]
4. Yao, S.; Hu, S.; Zhao, Y.; Zhang, A.; Abdelzaher, T. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, Perth, Australia, 3–7 April 2017; pp. 351–360.
5. Wu, C.J.; Brooks, D.; Chen, K.; Chen, D.; Choudhury, S.; Dukhan, M.; Hazelwood, K.; Isaac, E.; Jia, Y.; Jia, B.; et al. Machine learning at facebook: Understanding inference at the edge. In Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), Washington, DC, USA, 16–20 February 2019; pp. 331–344.
6. Exynos 5 Octa 5422 Processor: Specs, Features. Available online: https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-5-octa-5422/ (accessed on 8 November 2019).
7. Jang, M.; Kim, K.; Kim, K. The performance analysis of ARM NEON technology for mobile platforms. In Proceedings of the 2011 ACM Symposium on Research in Applied Computation, Miami, FL, USA, 2–5 November 2011; ACM: New York, NY, USA, 2011; pp. 104–106.
8. Cheng, A.C.; Dong, J.D.; Hsu, C.H.; Chang, S.H.; Sun, M.; Chang, S.C.; Pan, J.Y.; Chen, Y.T.; Wei, W.; Juan, D.C. Searching toward pareto-optimal device-aware neural architectures. In Proceedings of the International Conference on Computer-Aided Design, San Diego, CA, USA, 5–8 November 2018; ACM: New York, NY, USA, 2018; p. 136.
9. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, 2–4 May 2016; Conference Track Proceedings.
10. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713.
11. Grimaldi, M.; Peluso, V.; Calimera, A. Optimality Assessment of Memory-Bounded ConvNets Deployed on Resource-Constrained RISC Cores. *IEEE Access* **2019**, *7*, 152599–152611. [CrossRef]
12. Peluso, V.; Cipolletta, A.; Calimera, A.; Poggi, M.; Tosi, F.; Mattoccia, S. Enabling energy-efficient unsupervised monocular depth estimation on armv7-based platforms. In Proceedings of the IEEE 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 1703–1708.
13. Brooks, D.; Dick, R.P.; Joseph, R.; Shang, L. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *IEEE Micro* **2007**, *27*, 49–62. [CrossRef]
14. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
15. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
16. Kim, Y.G.; Kong, J.; Chung, S.W. A survey on recent OS-level energy management techniques for mobile processing units. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 2388–2401. [CrossRef]

17. Hardkernel. Odroid-XU4 User Manual. Available online: https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwi7y4Sn94vmAhXayYsBHb7kCb0QFjAAegQIARAC&url=https%3A%2F%2Fmagazine.odroid.com%2Fwp-content%2Fuploads%2Fodroid-xu4-user-manual.pdf&usg=AOvVaw0iPReYKQAm-qcHwvYU8mde (accessed on 8 November 2019).
18. Zhang, X.; Wang, Y.; Shi, W. pcamp: Performance comparison of machine learning packages on the edges. In Proceedings of the USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18), Boston, MA, USA, 10 July 2018.
19. Xia, C.; Zhao, J.; Cui, H.; Feng, X. Characterizing DNN Models for Edge-Cloud Computing. In Proceedings of the 2018 IEEE International Symposium on Workload Characterization (IISWC), Raleigh, NC, USA, 30 September–2 October 2018; pp. 82–83.
20. Ignatov, A.; Timofte, R.; Chou, W.; Wang, K.; Wu, M.; Hartley, T.; Van Gool, L. Ai benchmark: Running deep neural networks on android smartphones. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
21. Almeida, M.; Laskaridis, S.; Leontiadis, I.; Venieris, S.I.; Lane, N.D. EmBench: Quantifying Performance Variations of Deep Neural Networks across Modern Commodity Devices. In Proceedings of the 3rd International Workshop on Deep Learning for Mobile Systems and Applications, Seoul, Korea, 21 June 2019; ACM: New York, NY, USA, 2019; pp. 1–6.
22. Hanhirova, J.; Kämäräinen, T.; Seppälä, S.; Siekkinen, M.; Hirvisalo, V.; Ylä-Jääski, A. Latency and throughput characterization of convolutional neural networks for mobile computer vision. In Proceedings of the 9th ACM Multimedia Systems Conference, Amsterdam, The Netherlands, 12–15 June 2018; ACM: New York, NY, USA, 2018; pp. 204–215.
23. Yang, T.J.; Howard, A.; Chen, B.; Zhang, X.; Go, A.; Sandler, M.; Sze, V.; Adam, H. Netadapt: Platform-aware neural network adaptation for mobile applications. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 285–300.
24. Cai, H.; Zhu, L.; Han, S. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
25. Peluso, V.; Rizzo, R.G.; Calimera, A.; Macii, E.; Alioto, M. Beyond ideal DVFS through ultra-fine grain vdd-hopping. In Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip, Tallinn, Estonia, 26–28 September 2016; Springer: Cham, Switzerland, 2016; pp. 152–172.
26. Sahin, O.; Varghese, P.T.; Coskun, A.K. Just enough is more: Achieving sustainable performance in mobile devices under thermal limitations. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Austin, TX, USA, 2–6 November 2015; pp. 839–846.
27. Isuwa, S.; Dey, S.; Singh, A.K.; McDonald-Maier, K. TEEM: Online Thermal-and Energy-Efficiency Management on CPU-GPU MPSoCs. In Proceedings of the IEEE 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 438–443.
28. Bhat, G.; Singla, G.; Unver, A.K.; Ogras, U.Y. Algorithmic optimization of thermal and power management for heterogeneous mobile platforms. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *26*, 544–557. [CrossRef]
29. Dey, S.; Guajardo, E.Z.; Basireddy, K.R.; Wang, X.; Singh, A.K.; McDonald-Maier, K. Edgecoolingmode: An agent based thermal management mechanism for dvfs enabled heterogeneous mpsocs. In Proceedings of the IEEE 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), Delhi, India, 5–9 January 2019; pp. 19–24.
30. Linaro Toolchain. Available online: https://www.linaro.org/downloads/ (accessed on 8 November 2019).
31. TensorFlow Lite Hosted Models. Available online: https://www.tensorflow.org/lite/guide/hosted_models (accessed on 8 November 2019).