Efficacy of topology scaling for temperature and latency constrained embedded convnets

Journal of
Low Power Electronics and Applications

MDPI

Article

Efficacy of Topology Scaling for Temperature and Latency Constrained Embedded ConvNets

Valentino Peluso [1], Roberto Giorgio Rizzo [2] and Andrea Calimera [2,*]

[1] Interuniversity Department of Regional and Urban Studies and Planning, Politecnico di Torino, 10129 Torino, Italy; valentino.peluso@polito.it
[2] Department of Control and Computer Engineering, Politecnico di Torino, 10129 Torino, Italy; robertogiorgio.rizzo@polito.it
* Correspondence: andrea.calimera@polito.it

Received: 3 February 2020; Accepted: 9 March 2020; Published: 13 March 2020

Abstract: Embedded Convolutional Neural Networks (ConvNets) are driving the evolution of ubiquitous systems that can sense and understand the environment autonomously. Due to their high complexity, aggressive compression is needed to meet the specifications of portable end-nodes. A variety of algorithmic optimizations are available today, from custom quantization and filter pruning to modular topology scaling, which enable fine-tuning of the hyperparameters and the right balance between quality, performance and resource usage. Nonetheless, the implementation of systems capable of sustaining continuous inference over a long period is still a primary source of concern since the limited thermal design power of general-purpose embedded CPUs prevents execution at maximum speed. Neglecting this aspect may result in substantial mismatches and the violation of the design constraints. The objective of this work was to assess topology scaling as a design knob to control the performance and the thermal stability of inference engines for image classification. To this aim, we built a characterization framework to inspect both the functional (accuracy) and non-functional (latency and temperature) metrics of two ConvNet models, MobileNet and MnasNet, ported onto a commercial low-power CPU, the ARM Cortex-A15. Our investigation reveals that different latency constraints can be met even under continuous inference, yet with a severe accuracy penalty forced by thermal constraints. Moreover, we empirically demonstrate that thermal behavior does not benefit from topology scaling as the on-chip temperature still reaches critical values affecting reliability and user satisfaction.

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

1. Introduction

Recent advances in deep learning have enabled the deployment of Convolutional Neural Networks (ConvNets) on tiny end-nodes powered by general-purpose cores. Embedded ConvNets push human-like perception on ubiquitous devices producing highly informative data on the edge. This is the keystone of several applications, e.g., context-sensing [1], health-monitoring [2], and object tracking [3], for which uploading raw data to the cloud would be unpractical due to the high costs of communication, uncertain response time, and privacy protection.

ConvNets are computing- and memory-intensive models that need aggressive compression to fit low-power CPUs. Along with custom compression pipelines based on quantization [4], pruning [5] and neural architecture search [6], a common trend today is to offer end-users a portfolio of pre-trained models with the same back-bone *topology* but variable size and hence, a different latency–accuracy

trade-off [7–11]. One can pick the implementation that best fits the available computing architecture and the application requirements, therefore reducing the design time.

The assembly of such models encompasses the concept of *topology scaling*, where the re-scaling of the ConvNet geometry is implemented by leveraging two design parameters, namely *input resolution* and *number of filters*. The way the two knobs work is intuitive. Input patterns with a lower resolution take faster to be processed, but they bring less information as some fine-grain detail may get lost, resulting in a lower accuracy. A lower number of convolutional filters implies that some features are dropped during the inference process, resulting in less memory and compute resources request, but also weakened expressive power and hence, less accuracy. Figure 1 gives a pictorial example of multiple implementations of the same ConvNet projected into the latency–accuracy space. The blue dots are the available Pareto optimal implementations, each with a different setting of input resolution and number of filters, represented by different radii. Under a given latency constraint, shown as the dashed vertical line, the best option is the one with a higher accuracy, the blue dashed circle in the plot.

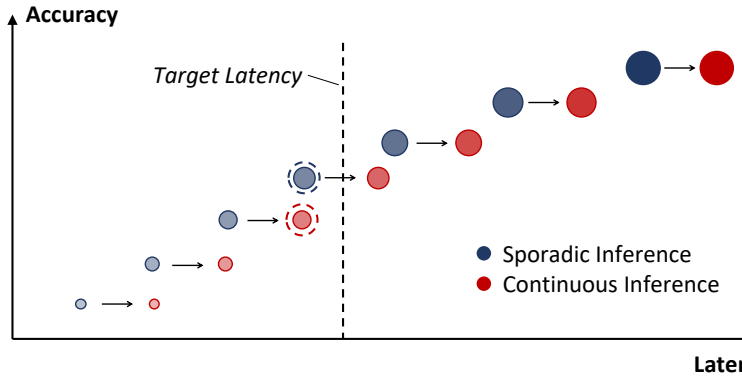


Figure 1. Latency-Accuracy trade-off of Mobile ConvNets in sporadic and continuous inference.

Unfortunately, this optimistic choice ensures latency only for sporadic execution since it ignores an important, yet often neglected aspect: the limited thermal design power of embedded systems prevents the execution of intensive workloads at maximum performance for a long runtime. Indeed, embedded systems (like mobile phones, for instance) integrate high-performance System-on-Chips (SoCs) on a small form-factor with no room for heat spreaders or active cooling systems (e.g., fan). When a power-demanding task, like inference, is run for a sustained period over continuous streams of data, the on-chip temperature increases, overstepping the safety threshold just after a few seconds. To avoid irreversible damages and ensure comfortable hand feeling, the operating system counteracts the high generation of heat by lowering the voltage-frequency level of the active cores. The low-power state is retained until the temperature gets back to safety and then left to recover speed. Known as thermal throttling, this control mechanism has a significant impact on performance, becoming the primary source of the latency mismatch compared to nominal operating conditions, as qualitatively illustrated with the red dots in the plot of Figure 1.

Thermally induced performance degradation should be considered in the early design stages through some form of over-design strategy and/or by implementing more efficient management policies. The problem is well established in the literature of embedded systems, and different thermal-aware hw/sw co-design techniques are available [12]. However, the interesting aspect here is to understand and explore solutions that leverage the statistical nature of ConvNets. It is clear that a more aggressive topology scaling is mandatory to ensure compliance with constraints under high thermal stress, which leads to weaker configurations, as shown in the plot by the dashed red circle. Therefore, the nominal accuracy often used as the main metric to assess the quality of an inference model is much lower than expected.

This work aims to provide the assessment of the impact of topology scaling in power-constrained systems over both functional (accuracy) and non-functional (latency and temperature) constraints. The

objective is to understand whether there exists a topology that does match the requirements even in continuous inference and, in case it does, to quantify the induced accuracy drop and whether it falls within acceptable ranges or not. Moreover, it is natural to ask whether topology scaling improves the power profile as well, and if it can serve as leverage to improve thermal stability. To address these issues, we built a characterization framework validated and tested for a chip-set powered with the ARM Cortex-A15 CPU that we used as a test bench to probe the performance of two state-of-art ConvNets, MobileNet [7] and MnasNet [11]. The collected results reveal that the gap between sporadic and continuous inference is large and that temperature-aware topology scaling is too weak a strategy that affects accuracy severely without any further savings in terms of temperature reduction.

The remainder of this paper is organized as follows: Section 2 reviews state-of-the-art ConvNets for mobile applications, topology scaling strategy of ConvNets, and thermal management on embedded systems. Section 3 recalls previous studies on the performance of ConvNets on general-purpose embedded systems. Section 4 introduces the characterization framework that we developed to conduct our analysis. Section 5 describes the experimental setup and analyzes the collected results. Finally, Section 6 discusses the main insights and guides the reader towards novel perspectives for the power optimization of ConvNets.

2. Background

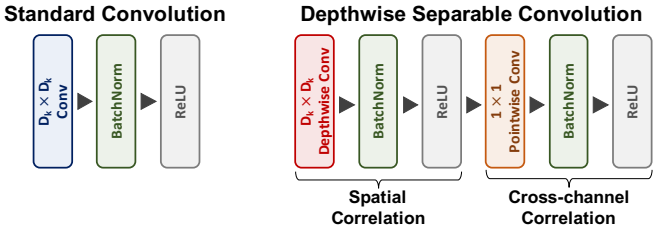
2.1. ConvNets for Mobile Applications: MobileNet and MnasNet

In the beginning, ConvNets were mainly optimized to improve accuracy. This led to deeper and more complex models with increased size [13–15]. The growing demand for portable applications drove the research towards more compact and fast ConvNets. Among the many existing ones, in this work we picked two representative examples considered state-of-the-art: (i) *MobileNet*, which introduces a convolution operator that decouples spatial and cross-channel correlations to reduce the computational complexity of the network; (ii) *MnasNet*, which is built using an automated neural architecture search based on reinforcement learning to identify the ConvNet topology that achieves the best trade-off between accuracy and inference latency for designing mobile ConvNets. MobileNet is designed by hand with the specific goal of reducing hardware resources. MnasNet can be considered the latest evolution, where the design is automated through an intelligent algorithm that improves over handcrafted solutions.

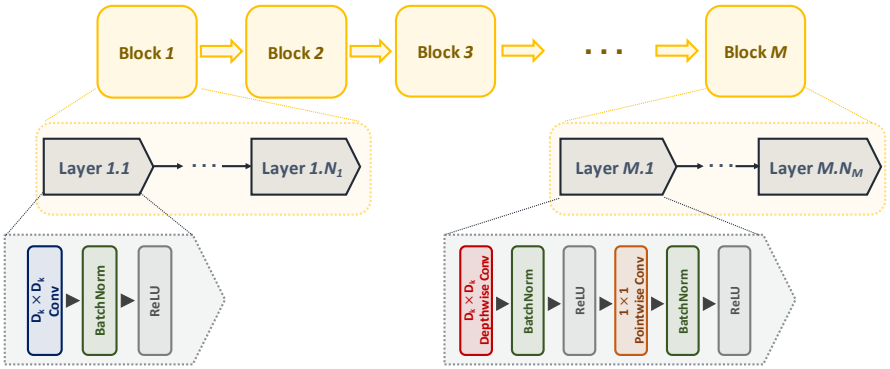
MobileNet [7]. As depicted in Figure 2a, MobileNet is based on a primitive block that implements a convolution operation called *depthwise separable convolution*, originally introduced in [16]. The standard convolution is factorized as two consecutive stages: (1) a *depthwise convolution*, i.e., a spatial convolution done independently over each input channel, followed by (2) a *pointwise convolution*, i.e., a 1×1 convolution to map cross-channel correlations onto a new channel space. The result is an approximation of the standard convolution with fewer arithmetic operations; indeed, the resulting reduction factor w.r.t. standard convolution is $\frac{1}{N} + \frac{1}{D_K^2}$, where N is the number of output channels and D_K is the kernel size of the depthwise convolution. The MobileNet architecture is built on a sequence of depthwise separable convolutions, except for the first layer, which is a full convolution. These layers down-sample the extracted features through strided convolution. A final average pooling reduces the spatial resolution to 1 before the fully connected layer that feeds into a softmax layer for classification. Counting depthwise and pointwise convolutions as separate layers, MobileNet has 28 layers.

MnasNet [11]. MnasNet is a model generated automatically using a new strategy called neural architecture search (NAS), which is a multi-objective problem optimized by means of a reinforcement learning policy that concurrently optimizes accuracy and performance. Unlike previous NAS solutions that adopted indirect metrics to account for the network complexity, e.g., the number of parameters, MnasNet makes use of on-board latency, measured by deploying the generated models on the actual device, as feedback to drive the optimization loop. As reported in Figure 2b, the MnasNet strategy is built on the concept of a hierarchical search space. This methodology factorizes a ConvNet into a

sequence of parametric blocks and then searches the best layer configuration within each block such that constraints are met. MnasNet allows different layers to use different operators, e.g., standard convolution, depthwise separable convolution, and inverted residual convolution [8], but forces all layers in each block to share the same structure, offering a proper balance between search space size and layer diversity.



(**a**) MobileNet basic convolution operation.



(**b**) MnasNet Architecture.

Figure 2. (**a**)-left: Standard convolutional layer with batch normalization and relu. (**a**)-right: depthwise separable convolution with depthwise and pointwise layers followed by batch normalization and relu (from [7]). (**b**) Factorized hierarchical search space of MnasNet (from [11]).

MobileNet and MnasNet represent state-of-the-art ConvNets for mobile applications, guaranteeing fast computation, and high accuracy thanks to their optimized architecture. Both support the topology scaling strategy discussed in this work (more details in Section 2.2) to deploy models on different hardware platforms, while other ConvNets do not. Furthermore, pre-trained models with different topology configurations are freely available as part of open-source repositories, hence, their use does not require time-consuming re-training stages and allows the replicability of the experiments.

2.2. ConvNets Topology Scaling

A modular re-scaling of a ConvNet topology is achieved by playing with the resolution of the input, i.e., *Input Resizing*, and/or the total number of convolutional filters, i.e., *Filter Pruning*. Figure 3 provides a graphical definition of these two knobs. Input resizing affects the dimensions of data fed as input to the ConvNet. A lower resolution lacks fine-grain details, thus leading to a lower accuracy, yet, the overall number of operations to be executed during the inference dramatically drops along the whole chain of layers. Referring to the picture, the inner features can be scaled by adopting an image resolution $\rho' < \rho$, where ρ is the input resolution of the baseline model. Specifically, the size of the inner features gets lower, from D_F to $\delta \cdot D_F$ for inputs and from $D_F^{'}$ to $\delta \cdot D_F^{'}$ for outputs, where $\delta = \rho'/\rho$. The scaled model is thereby unloaded by factor δ^2. Although we referred to squared images of size ρ, the technique can be adapted for rectangular images as well.

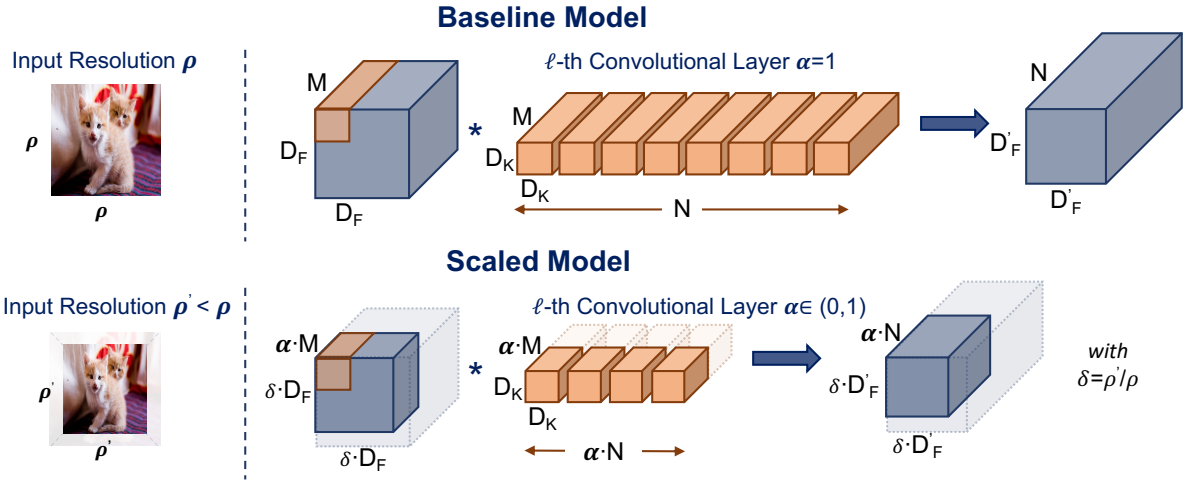


Figure 3. Depiction of topology scaling knobs over a convolution layer: the *input resolution* ρ cuts the size of the convolution input; the *width multiplier* α reduces the number of convolutional filters.

Filter pruning is implemented by tuning a hyper-parameter, named the width multiplier α, which defines the ratio of input and output channels within each convolution layer. As shown in the picture, the model can be scaled by setting $\alpha \in (0, 1]$, where α=1 denotes the baseline model. For a generic convolutional layer with M input channels and N output channels, the scaled model counts $\alpha \cdot M$ and $\alpha \cdot N$ channels, respectively, reducing the number of multiplications and filter parameters by roughly α^2.

2.3. Thermal Design Power Management

Off-the-shelf SoCs implement a reactive thermal management mechanism that dynamically controls the Voltage-Frequency (VF) operating point in order to reduce the power density and cool down the core. Dynamic Voltage-Frequency Scaling (DVFS) is an effective strategy to ensure thermal stability as the active power has a quadratic dependence from voltage and a linear dependence from frequency, resulting in theoretical cubic scaling. The efficiency of DVFS can be pushed even beyond these theoretical relationships through custom power distribution schemes, as shown in [17].

In order to match a target inference latency constraint (L_t), active cores operating for high-performance are set to the highest voltage and the maximum frequency available, which we refer to as VF_{max}. When the on-chip temperature reaches a critical threshold, T_{max}, the operating system invokes a safety mechanism—thermal throttling—leading cores towards a low-power mode with a lower VF that restores thermal equilibrium. A qualitative analysis of this strategy is depicted in Figure 4. Maximum performance pushes temperature on the critical threshold T_{max} and forces the SoC to throttle the performance, from VF_{max} to VF_{low} as reported in the middle plot. As soon as the temperature falls below T_{max}, the SoC switches back to VF_{max}, and the temperature starts rising again. Fluctuations around the safety threshold continue until the end of the task, and so does the VF.

For sporadic inference, cores operating at VF_{max} ensures a latency meeting the constraint L_t, as illustrated in the bottom plot of Figure 4 (green area, $L_1 < L_t$). When inference is held for long time intervals, two side effects do emerge: (i) working at VF_{low} introduces a performance penalty that turns into multiple latency constraint misses ($L_i > L_t$) within each inference run INF_i (successive red areas); (ii) the cyclic swapping among different VF levels makes latency prediction more uncertain (L_i different at each INF_i).

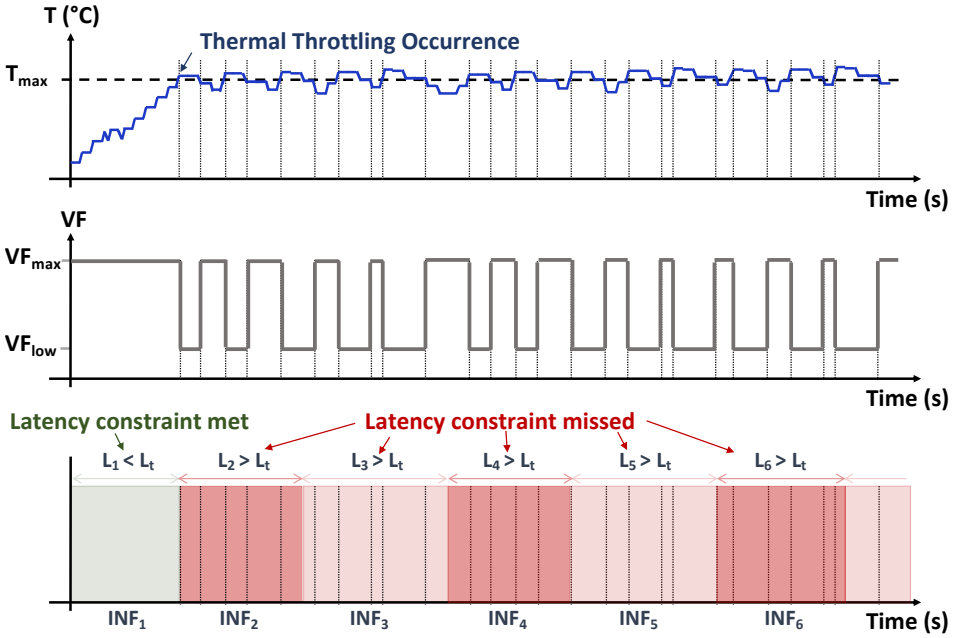


Figure 4. Qualitative analysis of temperature, VF and inference latency evolution over time under reactive thermal management.

3. Related Works

Understanding the performance of ConvNets on general-purpose embedded systems is paramount for the rapid diffusion of edge intelligence. In this regard, several studies [18–20] investigated opportunities and limits offered by commercial off-the-shelf platforms for inference with state-of-the-art ConvNets. However, they were mostly restricted to sporadic inference, neglecting the thermal stability of the hosting hardware.

Profiling the thermal behavior of inference tasks is a much less explored field. Preliminary analyses were conducted in [21], in which the authors compared the inference latency across different inference engines. The results have shown that temperature has similar trends across different frameworks and reaches critical value after few seconds of execution. In [22], it is shown that mobile GPUs also experience performance reduction due to thermal throttling, even if with a smoother gradient than CPUs. In our previous work [23], we conducted a parametric analysis to study the effects of two thermal management strategies based on DVFS. Although a proactive thermal governor could reduce thermal-induced latency degradation, it is not a viable solution for execution under tight constraints.

Dedicated co-processors would guarantee lower power consumption, hence lowering temperature [24], but the lack of stable tool-chains and the high design cost still limit their diffusion on a large scale. This aspect motivates the need for attacking the problem from a software perspective. In this regard, our work aims to assess the efficacy of topology scaling on mobile CPUs considering concurrent latency and temperature constraints.

4. Latency and Temperature Characterization

We developed an automated framework that profiles the execution time and the CPU temperature during both sporadic and continuous inference. As illustrated in Figure 5, the framework includes two main components: an online characterization flow compiled and executed on the hosting hardware that collects the statistics, and an off-line procedure that returns the topology setting meeting a given latency constraint.

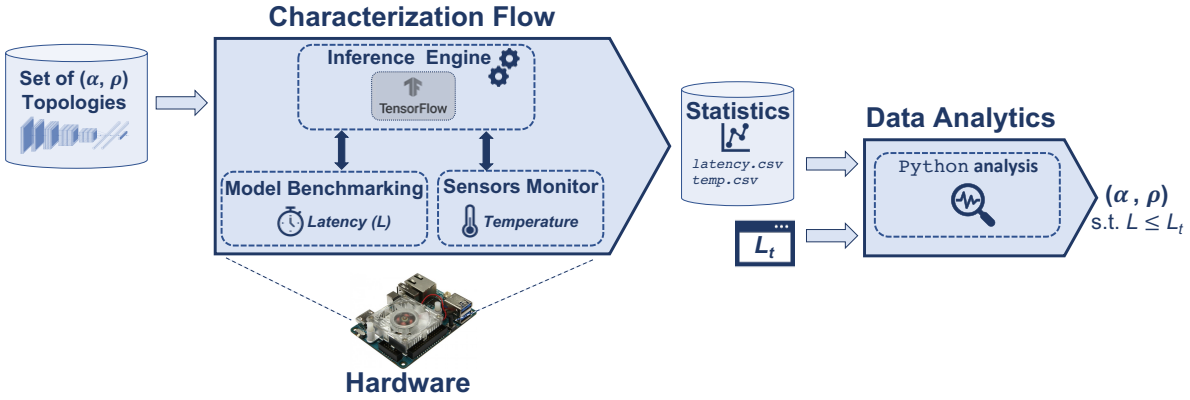


Figure 5. Schematic view of the proposed characterization framework.

The characterization flow integrates three software units: (i) an inference engine that processes the ConvNet on-chip; (ii) a model benchmarking tool that measures the inference latency (L); (iii) a monitoring routine that reads the CPU temperature from the on-chip sensors. The characterization process takes as the input a pre-trained ConvNet topology and profiles all the available (α, ρ) configurations under both sporadic and continuous inference. In our setup, we used *TensorFlow Lite* (TFL), an inference engine optimized for mobile platforms. For the latency measurements, we adopted a modified version of the *TensorFlow Lite Model Benchmark* utility integrated into TFL. Specifically, our implementation enables us to register the execution time of each inference. For the temperature measurements, the sampling rate was set to 10 ms.

The statistics collected by the characterization flow were screened by a procedure written in `Python` that identifies the pair (α, ρ) meeting the latency constraint (i.e., $L \leq L_t$).

5. Experimental Setup and Results

The objective of our analysis is threefold: (i) identify the topology configuration (if any) that meets the latency requirements under continuous inference; (ii) quantify the accuracy degradation induced by thermal management; (iii) assess the relationship between the thermal profile of the processing units and topology scaling. In this section, we first outline the hardware specifications of the board used in the experiments, together with the software environment adopted for the deployment. Secondly, we introduce the ConvNets taken as benchmarks. Finally, we analyze the collected results, with emphasis on both functional and non-functional metrics. Table 1 summarizes the notations used throughout the text, together with their definitions.

Table 1. Table of notations.

Notation	Description
α	Width multiplier
ρ	Input resolution
L_t	Target latency
T_{max}	Safety temperature threshold
T_{idle}	Idle temperature
VF_{max}	Voltage-frequency level for maximum performance
VF_{low}	Voltage-frequency level for thermal throttling
(α_s, ρ_s)	Solution meeting L_t in sporadic inference
(α_c, ρ_c)	Solution meeting L_t in continuous inference

5.1. Experimental Setup

Hardware and Software. The test-bench is the Odroid-XU4 platform [25] powered with a Samsung Exynos 5422 chip-set [26] that integrates a quad-core ARM Cortex-A15 CPU with a 32 kB L1 data cache on each core, 2 MB of the L2 cache and 2 GB of LPDDR3 RAM. At maximum performance,

the cores operate at $VF_{\max}$ = 1.3625 V @ 2 GHz. In our experiments, all four cores were active and worked in parallel. Notice that the SoC also integrates a low-power ARM Cortex-A7 CPU, which was disabled.

The board runs Ubuntu Mate 16.04, kernel Version 3.10.106-154, released by Hardkernel. The built-in thermal governor scales the operating point of the A15 cores to VF_{low} = 0.8875 V @ 900 MHz as soon as the temperature exceeds the threshold $T_{\max}$ = 90 °C. The idle on-chip temperature is T_{idle} = 65 °C. Even though the board is equipped with a cooling fan, we switched it off to emulate fan-less systems deployed into mobile devices, e.g., smartphones and tablets.

As the inference engine, we adopted *TensorFlow Lite 1.14* [27] by Google. The tool offers a collection of deep learning routines optimized to run on the ARM Cortex-A architecture. Specifically, the convolutional operators were compiled with SIMD instructions to leverage the parallelism offered by the ARM NEON unit. TensorFlow Lite was cross-compiled using the GNU ARM Embedded Toolchain (Version 6.5) [28].

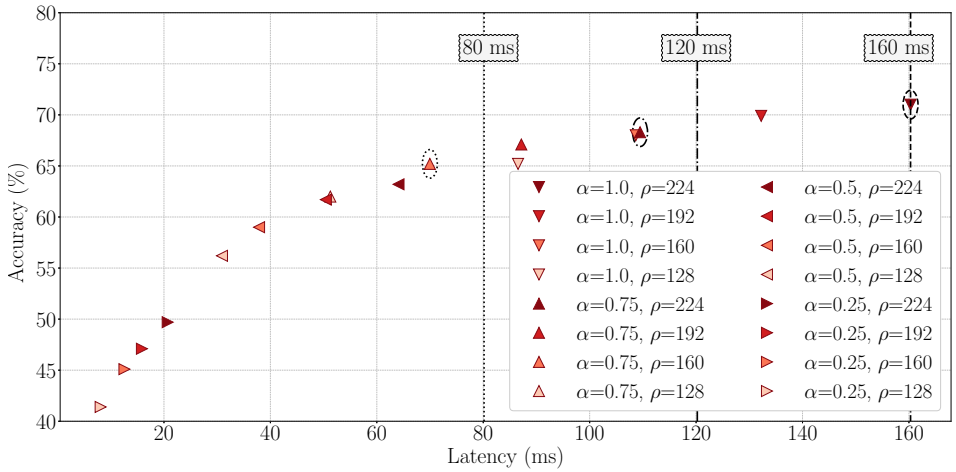
ConvNet Benchmarks. As benchmarks, we used two representative ConvNets optimized for embedded systems, namely MobileNet and MnasNet trained on the ImageNet dataset; both are open-sourced through TensorFlow Hosted Models [29]. For MobileNet, the available topology scaling options are $\alpha = \{0.25, 0.5, 0.75, 1.0\}$ and ρ = {224, 192, 160, 128}; MnasNet is provided with $\alpha = \{0.5, 0.75, 1.0\}$ and ρ = {224, 192, 160, 128, 96}.

Table 2 reports the size of the models used in our analysis. As explained in Section 2.2, α modulates the model size by cutting the number of convolutional filters, while ρ reduces the size of the activations in the inner layers.

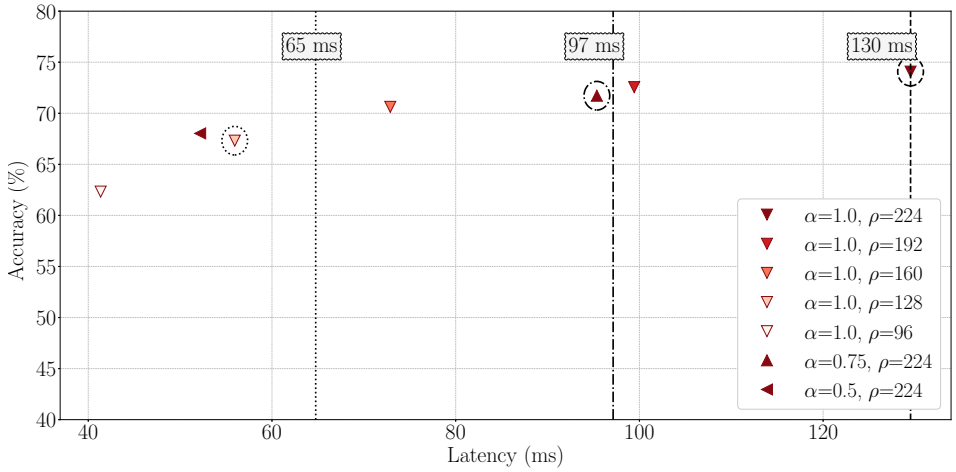
Table 2. MobileNet and MnasNet model size.

Model	α	ρ	Size (MB)
MobileNet	1.0	224, 192, 160, 128	16.9
	0.75	224, 192, 160, 128	10.3
	0.5	224, 192, 160, 128	5.3
	0.25	224, 192, 160, 128	1.9
MnasNet	1.0	224, 192, 160, 128, 96	17.0
	0.75	224	12.0
	0.5	224	8.5

Theoretically, topology scaling can be implemented by any value of α and ρ. However, in our analysis, we only considered the pre-trained configurations available in the TensorFlow Hosted Models repository (please refer to Figure 6). Fortunately enough, the available pairs of (α, ρ) cover the worst-case scenario widely: the tiniest networks have a considerably low value of top-1 accuracy, 41.4% and 68.03% for MobileNet (α = 0.25, ρ = 128) and MnasNet (α = 0.50, ρ = 224), respectively. Configurations with lower accuracy may be a critical choice as they would be of no practical use in real-life applications.



(**a**) MobileNet v1.



(**b**) MnasNet.

Figure 6. Latency–Accuracy trade-off of the selected benchmarks for sporadic inference.

Figure 6 shows the latency–accuracy trade-off offered by the selected benchmarks for all the available (α, ρ) pairs. The reported values refer to sporadic inference and give the minimum latency that can be achieved on the hosting hardware. Specifically, we measured the average execution time at maximum performance ($VF_{\max}$) over 100 inference runs, each of them interleaved by a two-second pause to restore the idle temperature. We observed that not all the available topology configurations are Pareto optimal. Indeed, optimality is strictly related to the hosting hardware [20], and those reported in the plots only refer to our target CPU.

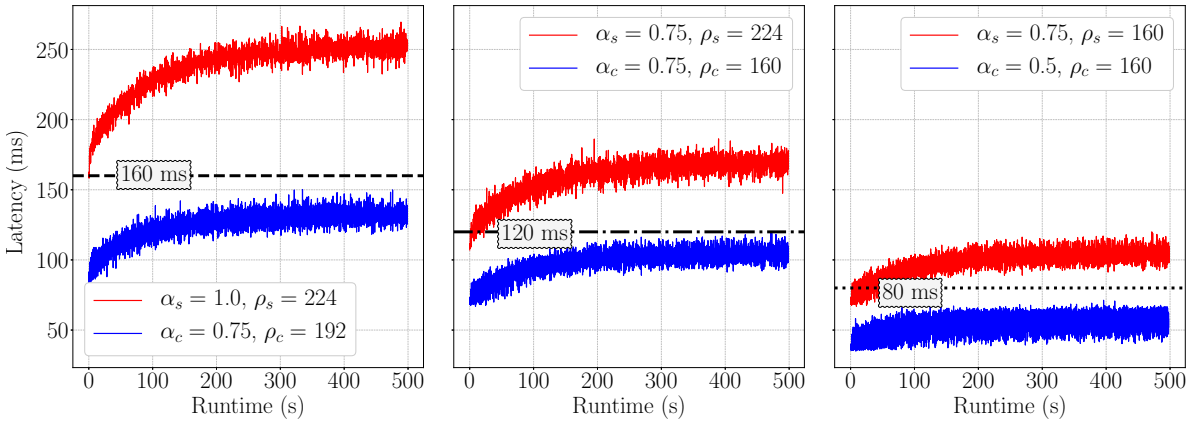
The plots also highlight (circled markers) the topology settings (α_s, ρ_s) that satisfy the latency constraint L_t for a sporadic inference. We analyzed three different targets: (i) $L_{\max}$ (dashed line), (ii) $0.75 \cdot L_{\max}$ (dash-dotted line), (iii) $0.50 \cdot L_{\max}$ (dotted line); $L_{\max}$ denotes the latency of the largest topology in sporadic execution, 160 ms and 130 ms for MobileNet and MnasNet respectively. The selected constraints are arbitrary and may change depending on the application specifications; however, they serve well the purpose of our characterization.

5.2. Results

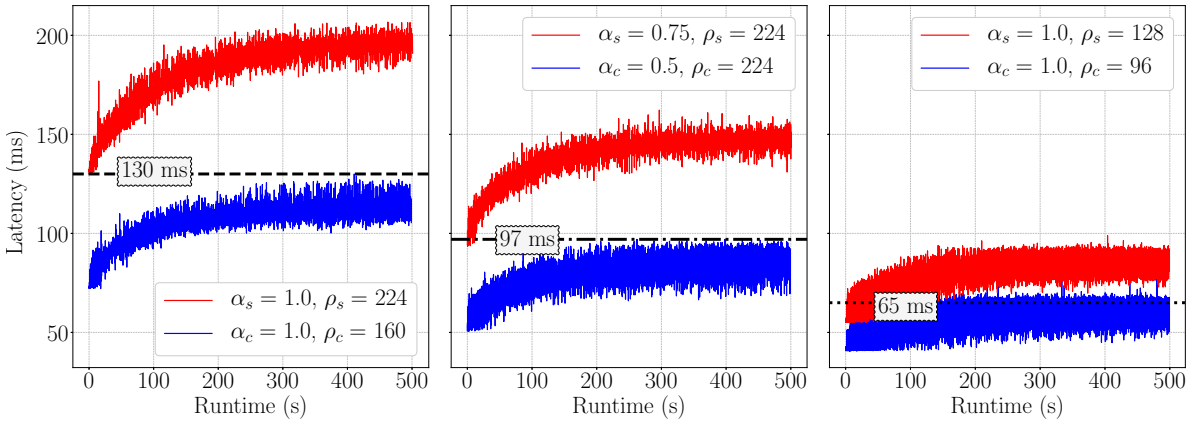
To assess the latency mismatch induced by thermal throttling during continuous inference, we measured the execution time of each inference during an overall runtime of 500 s, without any system pause between the successive runs. The analysis was conducted for all the pairs (α, ρ) in order to

identify the topology configurations (α_c, ρ_c) that strictly satisfy the target latency L_t during the whole 500-s time window.

Figure 7a,b summarize the main results. For each latency constraint (horizontal dashed line), the plots compare the latency curves for two different topology configurations: (α_s, ρ_s), i.e., the configuration selected for sporadic inference; (α_c, ρ_c), i.e., the configuration meeting L_t in continuous inference. The comparison aims to assess the performance gap between a trivial selection that would use (α_s, ρ_s) also for continuous inference with respect to a thermal-aware characterization. Specifically, the analysis of the figures reveals three key observations. First, as topology scaling creates an additional timing margin, operating at (α_c, ρ_c) keeps the execution time under the horizontal line for most cases. The only exception is MnasNet for L_t = 65 ms (right-most plot in Figure 7b), where even the most compact configuration violates the constraints. Since our analysis only considers the available pre-trained models, it is fair to assume that there exists at least one additional version that could meet the target latency (e.g., with lower ρ). Secondly, the configurations (α_s, ρ_s) miss the constraint after few seconds of execution. For instance, MnasNet with (α = 1, ρ = 224) enters in violation after 2.37 s, i.e., 18 continuous inferences. Moreover, for both sporadic and continuous inference, the latency reaches a stable level defined by the thermal constant of the board. Third, thermal management generates a significant latency variation across adjacent runs as the CPU continuously switches between $VF_{\max}$ and VF_{low}, making latency unpredictable, as evident from the noisy curves.



(**a**) MobileNet v1



(**b**) MnasNet

Figure 7. Latency trend of the ConvNet benchmarks running continuous inferences for 500 s.

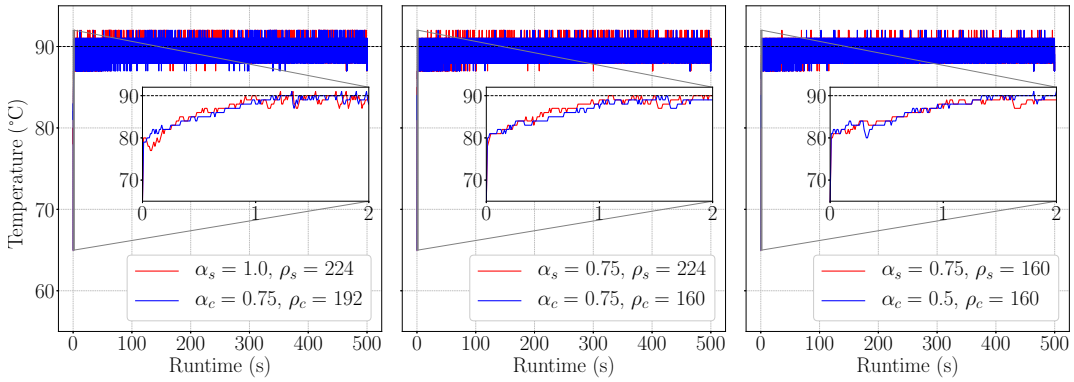
Aggressive topology scaling decreases the computation requirements at the cost of lower prediction quality. Table 3a,b quantify the penalty for the two benchmarks under analysis. Specifically, for each L_t, the first row reports the top-1 classification accuracy at (α_s, ρ_s), the second at (α_c, ρ_c). In the

worst case, MnasNet experiences an accuracy degradation of 4.99% under L_t = 65 ms, whereas for MobileNet the highest loss is 6.2% under L_t = 80 ms. As expected, continuous inference induces a larger accuracy loss. Although quite intuitive, this observation is neglected by current optimization pipelines for embedded ConvNets that do not consider temperature.

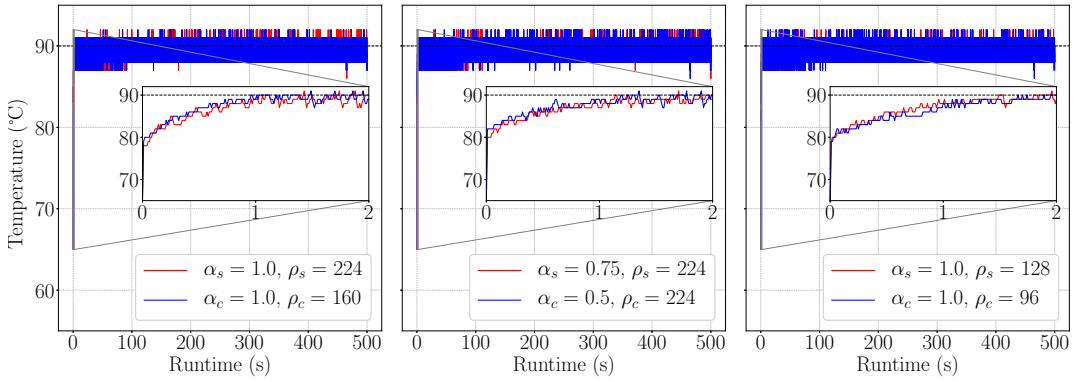
Table 3. Top-1 Accuracy under latency constraint for sporadic and continuous inference.

(a) MobileNet v1.					(b) MnasNet.				
L_t (ms)	Inference	α	ρ	Top-1 (%)	L_t (ms)	Inference	α	ρ	Top-1 (%)
160	Sporadic	1.0	224	71.0	130	Sporadic	1.0	224	74.08
	Continuous	0.75	192	67.1		Continuous	1.0	160	70.63
120	Sporadic	0.75	224	68.3	97	Sporadic	0.75	224	71.72
	Continuous	0.75	160	65.2		Continuous	0.5	224	68.03
80	Sporadic	0.75	160	65.2	65	Sporadic	1.0	128	67.32
	Continuous	0.5	160	59.0		Continuous	1.0	96	62.33

Beyond accuracy and latency, the temperature is also a primary design concern. Indeed, high temperatures harm the reliability of the device as they accelerate aging [30] and worsen the user experience [31] as hand devices get hot. We used on-board temperature sensors to study the CPU thermal profile during continuous inference. The plots in Figure 8 reveal that topology scaling has negligible effects on power consumption since both (α_s, ρ_s) and (α_c, ρ_c) show quasi-overlapping temperature profiles. In all the cases under analysis, the peak temperature reaches 90 °C after few seconds, then, it stays almost constant until the end of the execution; fluctuations are due to the continuous switching between VF_{max} and VF_{low} forced through the thermal governor.



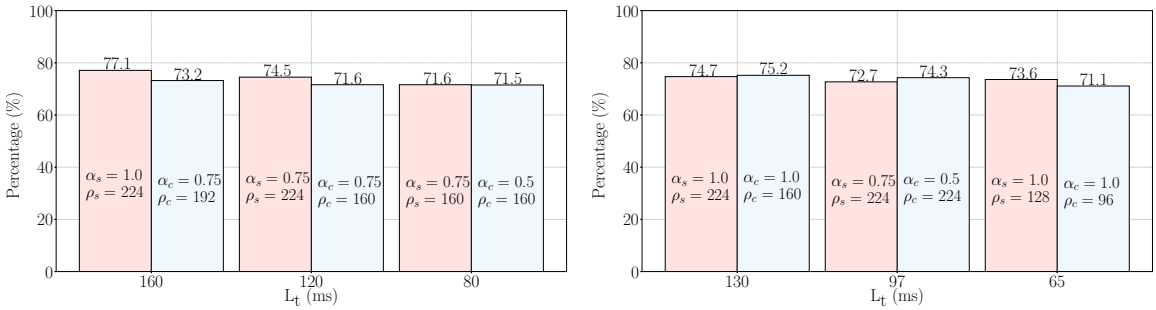
(**a**) MobileNet v1



(**b**) MnasNet

Figure 8. Temperature profile of the ConvNet benchmarks running continuous inferences for 500 s.

Moreover, Figure 9 reports the percentage of time spent at VF_{low} over the 500-s runtime to quantify the amount of thermal-throttling. We recorded small differences between sporadic and continuous inference, the highest was 3.9% (MobileNet for $L_t = 160$ s). Surprisingly, scaled MnasNets show higher throttling percentage for $L_t = 130$ s and $L_t = 97$ s. Once again, our findings confirm the inefficiency of topology scaling for power reduction.



(**a**) MobileNet v1 (**b**) MnasNet

Figure 9. VF throttling time percentage.

Since the adopted benchmarks cover a wide selection across the accuracy–complexity space, the observed trends suggest that thermal throttling is not only independent of the network topology and/or complexity, but also unavoidable due to the massive workload of ConvNets. In other words, exploring ConvNets with a larger number of parameters as additional benchmarks would not affect the outcome of our analysis.

6. Discussion and Final Remarks

Our study demonstrates that topology scaling enables us to meet tight latency constraints, even in systems with limited thermal design power at the cost of additional accuracy degradation. However, a more accurate analysis of the temperature trends reveals that topology scaling represents a workaround rather than a definitive solution. Indeed, topology scaling has no direct effect on the source of performance degradation, that is, the high power consumption; instead, it just creates additional timing slack, which is rapidly burnt by thermal management. The system temperature still remains high around a critical value; hence, reliability and user satisfaction might remain open problems.

Topology scaling only reduces the number of operations, not their intensity. Even a lightweight configuration still requires full utilization of the available resources, keeping the power consumption near the peak value, as suggested by the collected temperature values.

Overall, our analysis opens up novel optimizations for energy-aware training of ConvNets. Whereas state-of-art techniques focus on the energy savings achieved through latency reduction, an effective deployment on embedded systems calls for power reduction. For this purpose, better cooperation between hardware resources and software optimizations is paramount: from the hardware side, with the development of more efficient thermal management policies, from the software side, with the introduction of power consumption as direct optimization objective in the compression/training pipeline.

The concept of run-time model adaptation gives another dimension to explore. As discussed in this work, the profiling of the thermal behavior identified a set of ConvNets able to meet different latency constraints under continuous inference. This might suggest a new form of topology scaling at run-time: depending on the actual system/application requirement, the right model, among those available, namely the one satisfying the constraint, can be deployed and run. This strategy would imply the availability of a large set of models, which is memory-consuming. In fact, each scaled model comes with its own set of weights learned for that specific geometry in order to maximize the accuracy.

In other words, to jump from a configuration to another at run-time is not possible unless multiple models are stored on-chip. Enabling this feature more efficiently is part of our active research.

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. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. 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. Multimed. Comput. Commun. Appl. (TOMM)* **2017**, *13*, 35. [CrossRef]
2. 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.
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. 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, 18–22 June 2018; pp. 2704–2713.
5. 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.
6. 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; p. 136.
7. 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.
8. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
9. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
10. Dai, X.; Zhang, P.; Wu, B.; Yin, H.; Sun, F.; Wang, Y.; Dukhan, M.; Hu, Y.; Wu, Y.; Jia, Y.; et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 11398–11407.
11. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 2820–2828.
12. Khan, O.; Kundu, S. Hardware/software co-design architecture for thermal management of chip multiprocessors. In Proceedings of the 2009 Design, Automation & Test in Europe Conference & Exhibition, Nice, France, 20–24 April 2009; pp. 952–957.
13. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.

14. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
15. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
16. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
17. Peluso, V.; Rizzo, R.G.; Calimera, A.; Macii, E.; Alioto, M. Beyond ideal DVFS through ultra-fine grain vdd-hopping. In *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 152–172.
18. 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.
19. 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; pp. 1–6.
20. Peluso, V.; Rizzo, R.G.; Cipolletta, A.; Calimera, A. Inference on the Edge: Performance Analysis of an Image Classification Task Using Off-The-Shelf CPUs and Open-Source ConvNets. In Proceedings of the 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Granada, Spain, 22–25 October 2019; pp. 454–459.
21. Velasco-Montero, D.; Femández-Bemi, J.; Carmona-Gálán, R.; Rodríguez-Vázquez, A. On the Correlation of CNN Performance and Hardware Metrics for Visual Inference on a Low-Cost CPU-based Platform. In Proceedings of the 2019 International Conference on Systems, Signals and Image Processing (IWSSIP), Osijek, Croatia, 5–7 June 2019; pp. 249–254.
22. Lee, J.; Chirkov, N.; Ignasheva, E.; Pisarchyk, Y.; Shieh, M.; Riccardi, F.; Sarokin, R.; Kulik, A.; Grundmann, M. On-device neural net inference with mobile gpus. *arXiv* **2019**, arXiv:1907.01989.
23. Peluso, V.; Rizzo, R.G.; Calimera, A. Performance Profiling of Embedded ConvNets under Thermal-Aware DVFS. *Electronics* **2019**, *8*, 1423. [CrossRef]
24. 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.
25. Hardkernel. Odroid-XU4 User Manual. Available online: https://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf (accessed on 8 November 2019).
26. 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).
27. TensorFlow Lite. Available online: https://www.tensorflow.org/lite (accessed on 8 November 2019).
28. Linaro Toolchain. Available online: https://www.linaro.org/downloads/ (accessed on 8 November 2019).
29. TensorFlow Lite Hosted Models. Available online: https://www.tensorflow.org/lite/guide/hosted_models (accessed on 8 November 2019).
30. 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]
31. Egilmez, B.; Memik, G.; Ogrenci-Memik, S.; Ergin, O. User-specific skin temperature-aware DVFS for smartphones. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 1217–1220.