TVFS: Topology Voltage Frequency Scaling for Reliable Embedded ConvNets

TVFS: Topology Voltage Frequency Scaling for Reliable Embedded ConvNets

Roberto Giorgio Rizzo, *Member, IEEE,* Valentino Peluso, *Student, IEEE,* and Andrea Calimera, *Member, IEEE*

***Abstract*—This work introduces Topology Voltage Frequency Scaling (TVFS), a performance management technique for embedded Convolutional Neural Networks (ConvNets) deployed on low-power CPUs. Using TVFS, pre-trained ConvNets can be efficiently processed over a continuous stream of data, enabling reliable and predictable multi-inference tasks under latency constraints. Experimental results, collected from an image classification task built with MobileNet-v1 and ported into an ARM Cortex-A15 core, reveal TVFS holds fast and continuous inference (from few runs, up to 2000), ensuring a limited accuracy loss (from 0.9% to 3.1%), and better thermal profiles (average temperature 16.4 °C below the on-chip critical threshold).**

***Index Terms*—Edge Computing; Deep Learning; Convolutional Neural Network; Continuous Inference.**

I. Introduction and Related Works

CONVOLUTIONAL Neural Networks (ConvNets) are a deep learning technology widely used to infer the semantic meaning of data. In the quest to push their implementation on portable low-power devices, the hardware-agnostic training procedures of the early years are now evolving towards multi-objective optimizations that cover other extra-functional metrics besides accuracy [1], e.g., memory space, throughput or energy consumption. The most advanced training pipelines available today leverage algorithmic transformations, such as pruning [2] and quantization [3], together with complex graph structures [4] and topology search algorithms [5], in order to meet latency and accuracy constraints on a target device [6].

Within such formulations, an often neglected aspect is that real-life applications might need multiple feed-forward passes to accomplish the task. This can happen when the final outcome is obtained by a consensus of predictions, like for test-time augmentation (TTA) [7], where copies of the same input generated through geometric transformations, e.g., rotation and flipping, are fed as sequential inputs to improve the quality of prediction, but also for time-series classification [8], [9], where input data streams are windowed and elaborated as sequence. In these cases, the latency is the overall time spent for completing N continuous inferences, with N known a-priori and defined by the application. Since ConvNets get optimized for maximal hardware usage, sustained high switching rates become a source of thermal instability. In fact, System-on-Chips (SoCs) deployed on high-end devices with small form-factor, like smartphones, tablets or smart cameras, have a low thermal design power (TDP) and may thus reach the critical temperature (around $90\,^{\circ}\mathrm{C}$ for most of the commercial boards) shortly when processing heavy workloads (two inference runs according to our experiments). High temperatures impact circuit reliability [10] and user's experience when handling the device [11].

To avoid thermal runaway, the use of a reactive control mechanism, known as thermal throttling, is a practical option. It leverages Voltage Frequency Scaling (VFS) to slow down the active cores and reduce the switching power, which in turn limits the heat generation rate restoring a safe on-chip temperature. As a side effect, performance gets unpredictable and timing violations may arise, undermining the optimization efforts done at training-time.

Thermal issues are not new in the field of embedded systems design, with plenty of practical HW/SW solutions tackling the problem from different angles [12]. Something more interesting here is to explore the margins made available by the statistical nature of ConvNets to mitigate the thermal effects. Some previous works, e.g. [13], [14], analyzed how ConvNets optimization could improve power consumption and latency. Since their focus was on high-performance platforms with high TDP, thermal constraints were not taken into consideration. Our prior work [15] addressed this specific problem with a preliminary study showing that ConvNets re-sized with Topology Scaling (TS) can meet the latency constraint even under sustained inference intervals, yet at the cost of lower prediction accuracy. However, the results revealed that TS is a way to compensate for thermally-induced effects, but not a solution to prevent them, which is prerogative of this work.

We hereby investigate on the cooperation between power-reduction techniques (i.e., VFS) and algorithmic optimizations (i.e., TS), introducing a latency-driven Topology Voltage Frequency Scaling (TVFS) scheme able to reach the optimal thermal-accuracy trade-off in multi-inference tasks. The experiments, conducted on a state-of-art ConvNet ported into an off-the-shelf mobile CPU, i.e., MobileNet-v1 [4] on the Cortex-A15 CPU by ARM, demonstrate that TVFS is a viable option to sustain long inference intervals (N up to 2000) under tight latency constraints. The accuracy loss (3.1% as a worst-case) is marginal compared to TS, and the average on-chip temperature keeps below the critical threshold ($-16.4\,^{\circ}\mathrm{C}$).

II. Background

A. Thermal-Aware Voltage-Frequency Scaling.

Modern SoCs offer a predefined set of Voltage-Frequency (VF) levels to adjust power and performance at run-time. CPUs with a low TDP can sustain high VF levels for a short time interval, as the high power consumption burns the available thermal headroom quickly. When the temperature exceeds the

Manuscript received May 14, 2020; revised July 16, 2020; accepted August 2, 2020. Date of publication XXXX XX, 2020; date of current version XXXX XX, 2020. This brief was recommended by Associate Editor X. XXXX. (Corresponding author: Andrea Calimera.)

R.G. Rizzo and A. Calimera are with the Department of Control and Computer Engineering, Politecnico di Torino, 10129 Torino, Italy (e-mail: andrea.calimera@polito.it).

V. Peluso is with the Interuniversity Department of Regional and Urban Studies and Planning, Politecnico di Torino, 10129 Torino, Italy.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier XX.XXXX/TCSII.XXXX.XXXXXXX

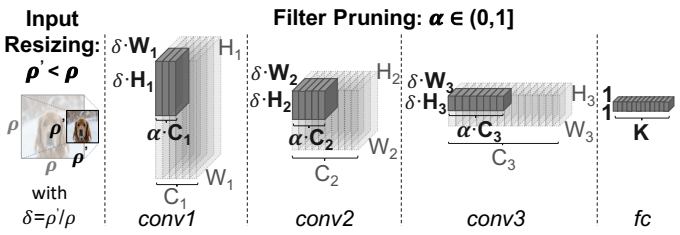

Figure 1: TS on a 4-layer ConvNet via input resizing (re-scaling factor ρ) and filter pruning (re-scaling factor α).

safety threshold $T_{\max}$, the protection mechanisms embedded into the operating system drive the active cores to a low power state with a low VF level until the on-chip temperature gets below a safe limit. This simple yet effective thermal throttling mechanism has a dramatic impact on performance, requiring a relaxation of the latency constraints.

Smarter control policies [16], [17] are built upon predictive models that infer the forthcoming resources demand, anticipating thermal trends, and thus preventing performance degradation via proactive VF scaling. However, ConvNets are static graphs, with no or less need for run-time predictions. This suggests that static proactive methods where the optimal VF level is defined at design time might serve the purpose. The characterization provided in [18] goes in this direction, but it shows that under the dense workload of a ConvNet, VF scaling alone is a too weak strategy. Indeed, to avoid thermal throttling would ask a too low VF level that makes the processing intrinsically slow. This motivates the need for a joint combination of circuit and algorithmic knobs.

B. ConvNets Topology Scaling

Topology Scaling (TS) [4] implements a reshaping of the ConvNet graph through *input resizing*, i.e., the lowering of the input resolution, and *filter pruning*, the drop out of convolutional filters within the layers. Through this modular approach, a set of pre-trained ConvNets with the same backbone topology but variable size and complexity can be made available to the end-user. Figure 1 graphically describes TS on a simple 4-layer ConvNet. Input data (e.g., images as reported in the picture) with a lower resolution require fewer operations along the whole chain of layers. Obviously, the classification may suffer from accuracy loss due to missing details. Assuming a square input of size $\rho' \times \rho'$, with $\rho' < \rho$, and ρ as the original size, the inner features get re-scaled by a factor $\delta = \rho'/\rho$, achieving a compression ratio of δ^2 for each layer. Besides resolution re-scaling, it is possible to play with the third dimension, i.e., the layers' width, by slicing the channels C. A hyper-parameter α, named *width multiplier*, is used to prune the input and output channels at each layer uniformly. The model can be scaled by setting $\alpha \in (0, 1]$, which implies that for each layer i, both the number of input channels C_{i} and output channels $C_{\text{i+1}}$ is re-scaled by a factor α. Therefore, the overall number of multiplications and weights reduces by roughly α^2 [4]. The smaller the α, the lower the expressive power of the network. Both α and ρ are knobs to reduce the inference latency, and their joint scaling offers a practical way to implement different (α, ρ) configurations in the accuracy-latency space.

III. Topology Voltage Frequency Scaling (TVFS)

A. Knobs and their effect on temperature and latency

Defined L_{s} as the nominal latency for a single forward pass of the baseline ConvNet topology (i.e., no TS) processed at maximum speed (i.e., VF_{max}), a classification task involving N runs would ideally take $L = N \cdot L_{\text{s}}$, which we consider as the nominal constraint. The picture changes when considering real embedded systems with limited TDP and low heat dissipation capability. As soon as the temperature reaches the safety threshold $T_{\max}$, the CPU enters a *sustained thermal throttling* state characterized by fast oscillations between VF_{max} and VF_{low} until the task ends. The plot of Fig. 2(a) shows a qualitative assessment of the thermal evolution, highlighting the temperature ripple around $T_{\max}$ (shaded red area) and the latency dilation, main source of timing overhead and constraint violation ($L > N \cdot L_{\text{s}}$).

The effect of a proactive use of the power knob VFS is shown in Fig. 2(b). At design time, the optimal VF level is selected so that the thermal gradient gets lower enough to avoid the occurrence of throttling events [18]. This is a viable option to prevent uncertainty, but still not enough to meet the latency constraint. In fact, a too low VF level might be needed, which reflects into slow processing and still timing overhead.

The latency knob implemented at the algorithmic-level through TS offers another possible option to achieve better thermal-latency trade-off without involving the VF scaling. Lighter ConvNets with fewer operations and memory accesses get intrinsically faster indeed. As demonstrated in [15], there exists a topology with a lower latency ($< L_{\text{s}}$) that creates enough slack to compensate for the delay degradation induced by the thermal throttling. This behavior is shown in Fig. 2(c). The temperature still reaches $T_{\max}$ with the same slope of the original ConvNet (Fig. 2(a)), but the N inferences end sooner. Also in this case, the latency of a single inference may vary over time, but the overall constraint is satisfied. To notice that TS may induce substantial losses of accuracy.

The proposed TVFS combines efficiency and functionality of VFS and TS, as is shown in Fig 2(d): it leverages VFS to control the thermal gradient and TS to speed-up the flow. To find the optimal balance turns to be an optimization problem.

B. Problem Formulation and Performance Trade-off

The TVFS optimization is as follows: *Given a classification task implemented via N consecutive forward passes of a static ConvNet, do search for the tuple (α_{opt}, ρ_{opt}, VF_{opt}), s.t. the latency constraint is met $L \leq N \cdot L_s$, and accuracy is maximal.*

The effect of TVFS can be appreciated through the qualitative space exploration depicted in Fig. 3. The horizontal dashed line defines the nominal constraint, while the gray gradient in the background highlights the percentage of thermal throttling (lighter is lower). The baseline implementation (no TS, VF_{max}) is represented by the topmost right implementation (red triangle); its latency is far from the constraint due to sustained throttling. With VFS (blue squares), the latency progressively reduces till the point of zero-throttling outside the gray area (circled blue square). This is a point of inflection, still far from the constraint, and any further reduction of VF

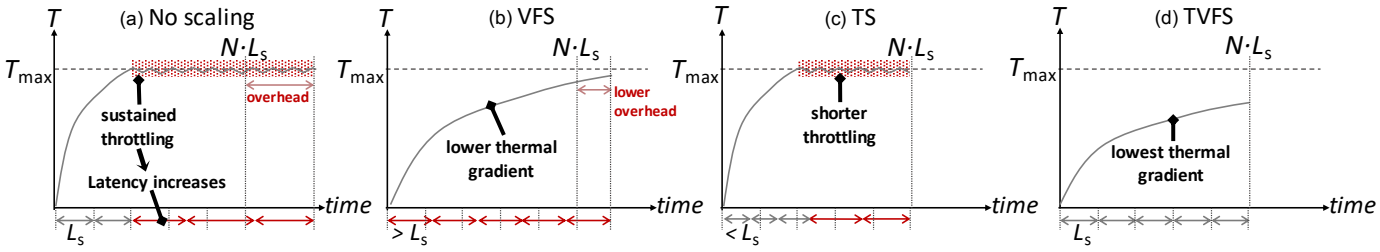

Figure 2: Temperature vs. Execution time for a multi-inference task.

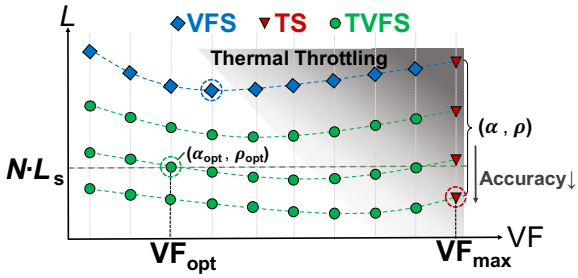

Figure 3: Classification latency under continuous inference at different TS and VF points.

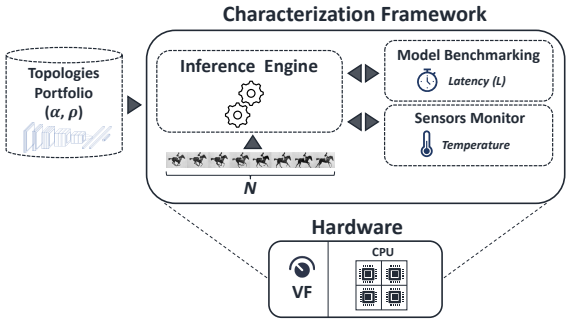

Figure 4: Characterization framework overview.

makes latency worse as the CPUs get excessively slow. As an orthogonal knob, TS does play on the opposite direction (red triangles) with faster topologies that approach and eventually cross (circled red triangle) the nominal constraint at the cost of prediction accuracy. Finally, TVFS (green circles) explores the diagonal direction, providing additional implementations with lower latency offset (brought by TS) and better thermal profiles (brought by VFS). In particular, the optimal solution (α_{opt}, ρ_{opt}, VF_{opt}) mentioned in the problem formulation (marked green circle) meets the target latency constraint with the largest topology possible and the best thermal profile. For the sake of completeness, the optimal solution depends on N: the gray area increases toward the left with larger N, pushing the optimal configuration towards the bottom-left corner.

C. Design Space Exploration

We opted for an offline exhaustive search across the three dimensions α, ρ, VF. This is justified by the following observations: (i) ConvNets are static graphs; (ii) N is known a-priori; (iii) the number of permutations (α, ρ, VF) is low.

Figure 4 shows an abstract view of the framework deployed to explore the design space. It takes as inputs the set of pre-trained ConvNet topologies (α, ρ) and the number of inferences N, and it returns the optimal configuration (α_{opt}, ρ_{opt}, VF_{opt}). There are three main components: (i) an inference engine to process the ConvNets on-chip through optimized neural kernels; (ii) a model benchmarking routine that collects the inference latency L; (iii) a software probe to read the CPU temperature periodically from the on-chip sensors. We used *TensorFlow Lite* as the inference engine, while we adopted a modified version of the *TensorFlow Lite Model Benchmark* utility to collect the execution times. During the exploration, each (α, ρ) model is deployed onto the board, made run N times for each VF level, and then off-loaded with the acquired samples of latency and temperature. Then, the collected results are processed by an off-line procedure that searches the optimal configuration (α_{opt}, ρ_{opt}, VF_{opt}).

IV. Experimental Setup and Results

A. Experimental Setup and Benchmarks.

As testbench, we used an Odroid-XU4 board hosting the Ubuntu Mate 16.04 operating system released by Hardkernel, version 3.10.106-154. The chip-set is the Samsung Exynos 5422, a mobile SoC with a quad-core ARM Cortex-A15 CPU controlled by a thermal governor with a set of 19 VF levels, from 200 MHz at 0.85 V to 2 GHz at 1.3625 V, step 100 MHz. When the on-chip temperature exceeds the threshold T=90 °C (defined by the vendor), the CPU switches into the low-power state VF_{low}=900 MHz at 0.8875 V, which is also the VF lower bound of the design space exploration. Hereafter, we denote each VF operating point just using its frequency value (in GHz). Even though the board is air-cooled, we switched the fan off to emulate the operating condition of portable devices. The ambient temperature of the setup is 25 °C. All the experiments were conducted with four active threads and the power governor set to *performance*. The integration of the additional low-power quad-core Cortex-A7 CPU available on-chip had shown no performance gain, and therefore it was disabled during the experiments. The framework of Fig. 4 was cross-compiled with the GNU ARM Embedded Toolchain v6.5 integrating TensorFlow Lite v1.14.

As ConvNet benchmark, we picked a state-of-art architecture designed for mobile applications: MobileNet-v1[1]. It is available in 16 different TS configurations pre-trained on the *ImageNet* dataset, with $\alpha = \{1.0, 0.75, 0.50, 0.25\}$ and $\rho = \{224, 192, 160, 128\}$. Each of them is quantized to 8-bit fixed-point in order to ensure smaller memory footprint and faster processing, yet with negligible accuracy loss. For the baseline model (α=1.0, ρ=224) at VF_{max}=2 GHz, the latency of a single-frame inference is L_s=32 ms, which has been used as target for the exploration. In the worst case (N=2000), the exploration of the design space, which counts 192 (α, ρ, VF) points (Sec. III-C), takes 2 h.

[1] www.tensorflow.org/lite/guide/hosted_models, visited on 2019/05/13

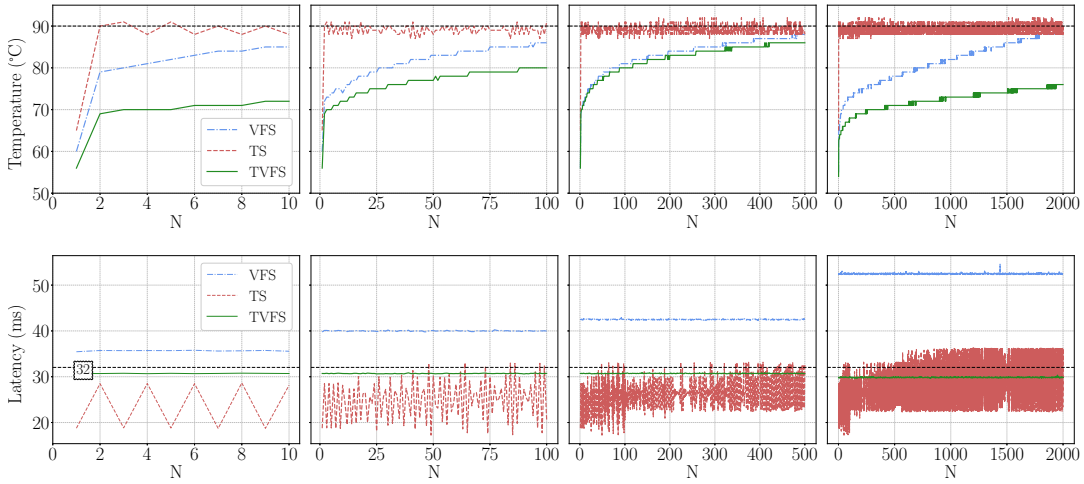

Figure 5: Temperature (top) and inference latency (bottom) trends over N={10, 100, 500, 2000} runs.

B. Results and Discussion.

The objective of this section is to provide an assessment of the proposed TVFS. We thereby provide a fair comparison against VFS and TS using different figures-of-merit, both functional and extra-functional. The former consists of the top-1 prediction accuracy (*Top-1*) evaluated on the ImageNet validation set. The latter include the latency for a single inference (L_{avg}) averaged over the N runs, the percentage of thermal throttling (*Th*), i.e., the amount of time the CPUs spent at VF_{low}, and the average on-chip temperature T_{avg} measured over the whole classification task. The experimental campaign of on-chip measurements is conducted for different values of N to cover a wide spectrum of possible use-cases, specifically we set N={10, 100, 500, 2000}. Lower values are common for TTA applications (e.g. N=10), whereas larger values are needed for time-series classification (from N=100 to 2000). According to our formulation, the latency constraint is $L \leq N \cdot L_{\text{s}}$, with L_{s}=32 ms as anticipated in the previous sub-section; thus, the constraint turns to be $L_{\text{avg}} \leq L_{\text{s}}$.

Table I reports the collected results in four multi-row sections, one for each value of N. Within them, the first row, labeled with NS (i.e., No Scaling), refers to the baseline model, that is the largest topology (α=1.0, ρ=224) processed at maximum voltage and frequency (VF_{max}), while VFS, TS, and TVFS rows are for the three scaling methods under analysis. For each strategy, we reported the optimal tuple (α, ρ, VF) as illustrated in Fig. 3 (circled markers). In addition, the plots in Fig. 5 show how the on-chip temperature (top-line plots) and the inference latency (bottom-line plots) evolve with N; the four plots have a different scale, one for each specific interval of N: {1-10}, {1-100}, {1-500}, {1-2000}.

The first observation is that TVFS outperforms the other strategies by far. It always meets the latency constraint, ensuring the highest accuracy with the lowest temperature profile, whereas other approaches cannot. As it can be inferred from the NS rows of Table I, the limited TDP of the system prevents the execution at maximum performance. Indeed, even for the shortest task, i.e., N=10, the percentage of thermal throttling is huge (46.3%), with an overall performance degradation of 37.8% with respect to the nominal constraint (from 32 ms to 44.1 ms). The picture gets worse for longer tasks. For instance, with N=2000 the throttling percentage rises up to 82.1%, with a latency overhead of 83.1% (from 32 ms to 58.6 ms).

Table I: Solutions (α, ρ, VF) for VFS, TS and TVFS under the latency constraint of L_{t}=$N \cdot 32$ ms.

N	Tech.	α	ρ	VF (GHz)	Top-1 (%)	L_{avg} (ms)	Th (%)	T_{avg} (°C)
10	NS	1.0	224	2.0	70.0	44.1	46.3	88.8
	VFS	1.0	224	1.8	70.0	35.7	0.0	82.4
	TS	1.0	160	2.0	66.9	23.1	38.5	89.4
	TVFS	**1.0**	**192**	**1.5**	**69.1**	**30.7**	**0.0**	**70.7**
100	NS	1.0	224	2.0	70.0	48.6	60.4	89.1
	VFS	1.0	224	1.6	70.0	40.0	0.0	82.1
	TS	1.0	160	2.0	66.9	24.9	53.7	89.1
	TVFS	**1.0**	**192**	**1.5**	**69.1**	**30.7**	**0.0**	**76.8**
500	NS	1.0	224	2.0	70.0	51.9	69.9	89.3
	VFS	1.0	224	1.5	70.0	42.5	0.0	83.5
	TS	1.0	160	2.0	66.9	26.3	63.3	89.2
	TVFS	**1.0**	**192**	**1.5**	**69.1**	**30.7**	**0.0**	**82.1**
2000	NS	1.0	224	2.0	70.0	58.6	82.1	89.5
	VFS	1.0	224	1.2	70.0	52.5	0.0	81.4
	TS	1.0	160	2.0	66.9	28.6	73.7	89.3
	TVFS	**1.0**	**160**	**1.1**	**66.9**	**29.8**	**0.0**	**73.6**

The VFS approach improves performance preventing CPUs to enter the low-power state for cooling down the silicon. This can be verified through the numbers reported in column (*Th*) of Table I which exactly report 0% of throttling. However, VFS still fails to reach the latency constraint, with an overhead ranging from 11.6% for N=10, to 64.0% for N=2000. The gap gets larger with N, as more aggressive voltage scaling is needed to extend the thermal headroom over longer timing intervals (down to 1.2 GHz for N=2000). Fig. 5 confirms these observations showing that lower VF levels keep down the thermal gradient, ensuring flat latency profiles. On the other hand, TS does matches the latency constraint ($L_{\text{avg}} < L_{\text{s}}$), yet incurring some accuracy loss due to the smaller topologies adopted (66.9% accuracy vs. 70% of the baseline model). To notice that TS does not alleviate the percentage of thermal throttling, that is lower than NS but still large (*Th* ranges from 38.5% for N=10, up to 73.7% for N=2000). The VF level is the highest one (2 GHz), and so the temperature, which keeps around the critical value indeed ($T_{\text{avg}} \geq 88.8$ °C), raising reliability concerns and user experience issues. Fig. 5 (top)

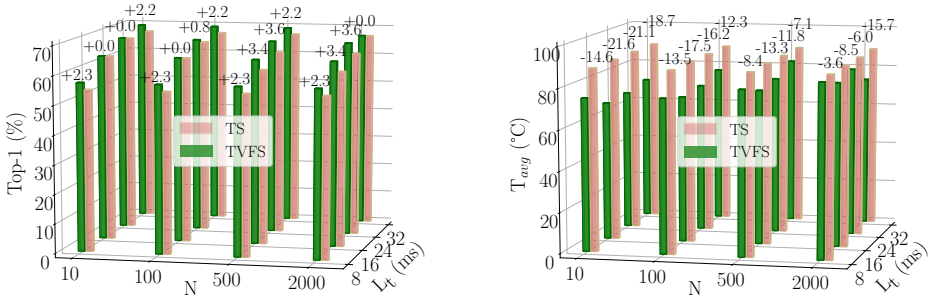

Figure 6: TVFS accuracy gain (left) and temperature drop (right) for tighter latency constraints.

shows that TS reaches the safety threshold of 90 °C after 2 runs and raises sustained throttling events. The continuous switching between VF_{max} and VF_{low} reflects on the inference latency and its ripple as highlighted in Fig. 5 (bottom).

TVFS combines the advantages of both VFS and TS. The most interesting aspect is that TVFS enables the task to run at a lower VF, still meeting the requirements. This condition allows a substantial reduction in power consumption, keeping the temperature far from that of VFS and TS, $-6.6\,^{\circ}\mathrm{C}$ and $-12.1\,^{\circ}\mathrm{C}$ on average over the N values. Moreover, even when TVFS loses accuracy getting closer to TS (for instance, N=2000), it still ensures lower temperatures ($-15.7\,^{\circ}\mathrm{C}$). In other words, TVFS plays with the two hardware and software knobs finding the best thermal-accuracy trade-off. The plots in Fig. 5 help visualize the benefits of TVFS: (i) temperature is far from the critical threshold, (ii) 0% of thermal throttling, (iii) no latency variations.

As a final remark, the barplots in Fig. 6 emphasize the savings brought by TS in terms of accuracy (left) and average temperature (right) under tighter latency constraints $L \leq N \cdot L_t$, with L_t={32, 24, 16, 8} ms, and different inference runs N. We omitted VFS as it violates the constraints. As reported by the top labels, TVFS achieves higher or equal accuracy (+3.6% as the best-case) with lower on-chip temperature from $-21.6\,^{\circ}\mathrm{C}$ (best-case) to $-3.6\,^{\circ}\mathrm{C}$ (worst-case). This validates TVFS and its important role in pursuing reliable ConvNet processing.

V. Conclusion

This work presented TVFS, a novel performance management strategy for embedded ConvNets on off-the-shelf low-power CPUs. Via the cooperation of power knob (VFS) and algorithmic optimization (TS), TVFS enables efficient multi-inference tasks under latency constraints ensuring thermal stability. The experimental results revealed that TVFS can sustain up to 2000 runs of MobileNet-v1 on the ARM A15 CPU under tight latency targets, keeping the average on-chip temperature below the critical threshold ($-16.4\,^{\circ}\mathrm{C}$) and a minor accuracy loss (3.1% as a worst-case). We believe that the integration of TVFS into standard inference engines for embedded systems would open to many practical applications. Moreover, smarter algorithms might speed-up the search phase to find the best setting, useful in the context of neural architecture search.

References

[1] A.-C. Cheng *et al.*, "Searching toward pareto-optimal device-aware neural architectures," in *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2018, p. 136.

[2] T.-J. Yang *et al.*, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 285–300.

[3] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[4] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[5] M. Tan *et al.*, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[6] V. Peluso, R. G. Rizzo, A. Cipolletta, and A. Calimera, "Inference on the edge: Performance analysis of an image classification task using off-the-shelf cpus and open-source convnets," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2019, pp. 454–459.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[9] T. Nilanon, J. Yao, J. Hao, S. Purushotham, and Y. Liu, "Normal/abnormal heart sound recordings classification using convolutional neural network," in *2016 Computing in Cardiology Conference (CinC)*. IEEE, 2016, pp. 585–588.

[10] D. Brooks, R. P. Dick, R. Joseph, and L. Shang, "Power, thermal, and reliability modeling in nanometer-scale microprocessors," *Ieee Micro*, vol. 27, no. 3, pp. 49–62, 2007.

[11] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-specific skin temperature-aware dvfs for smartphones," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1217–1220.

[12] Y. G. Kim, J. Kong, and S. W. Chung, "A survey on recent os-level energy management techniques for mobile processing units," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 10, pp. 2388–2401, 2018.

[13] S. M. Nabavinejad, H. Hafez-Kolahi, and S. Reda, "Coordinated dvfs and precision control for deep neural networks," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 136–140, 2019.

[14] W. Kang, D. Kim, and J. Park, "Dms: Dynamic model scaling for quality-aware deep learning inference in mobile and embedded devices," *IEEE Access*, vol. 7, pp. 168 048–168 059, 2019.

[15] V. Peluso, R. G. Rizzo, and A. Calimera, "Efficacy of topology scaling for temperature and latency constrained embedded convnets," *Journal of Low Power Electronics and Applications*, vol. 10, no. 1, p. 10, 2020.

[16] Y. Chen, D. Jahier Pagliari, E. Macii, and M. Poncino, "Battery-aware design exploration of scheduling policies for multi-sensor devices," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 201–206.

[17] S. Isuwa, S. Dey, A. K. Singh, and K. McDonald-Maier, "Teem: Online thermal-and energy-efficiency management on cpu-gpu mpsocs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 438–443.

[18] V. Peluso, R. G. Rizzo, and A. Calimera, "Performance profiling of embedded convnets under thermal-aware dvfs," *Electronics*, vol. 8, no. 12, p. 1423, 2019.