POLITECNICO DI TORINO
Repository ISTITUZIONALE

AdapTTA: Adaptive Test-Time Augmentation for Reliable Embedded ConvNets

Original
AdapTTA: Adaptive Test-Time Augmentation for Reliable Embedded ConvNets / Mocerino, Luca; Rizzo, Roberto G.; Peluso, Valentino; Calimera, Andrea; Macii, Enrico. - (2021), pp. 1-6. (Intervento presentato al convegno International Conference on Very Large Scale Integration (VLSI-SoC) nel 4-7 Oct. 2021) [10.1109/VLSI-SoC53125.2021.9606980].

Availability:
This version is available at: 11583/2947632 since: 2021-12-22T18:03:39Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/VLSI-SoC53125.2021.9606980

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

25 April 2024

AdapTTA: Adaptive Test-Time Augmentation for Reliable Embedded ConvNets

Luca Mocerino, Roberto G. Rizzo, Valentino Peluso, Andrea Calimera, Enrico Macii
Department of Control and Computer Engineering,
Politecnico di Torino, 10129 Turin, Italy

***Abstract*—Convolutional Neural Networks (ConvNets) are trained offline using the few available data and may therefore suffer from substantial accuracy loss when ported on the field, where unseen input patterns received under unpredictable external conditions can mislead the model. Test-Time Augmentation (TTA) techniques aim to alleviate such common side effect at inference-time, first running multiple feed-forward passes on a set of altered versions of the same input sample, and then computing the main outcome through a consensus of the aggregated predictions. Unfortunately, the implementation of TTA on embedded CPUs introduces latency penalties that limit its adoption on edge applications. To tackle this issue, we propose *AdapTTA*, an adaptive implementation of TTA that controls the number of feed-forward passes dynamically, depending on the complexity of the input. Experimental results on state-of-the-art ConvNets for image classification deployed on a commercial ARM Cortex-A CPU demonstrate AdapTTA reaches remarkable latency savings, from 1.40× to 2.21×, and hence a higher frame rate compared to static TTA, still preserving the same accuracy gain.**

I. Introduction & Motivations

Learning pattern recognition models with good generalization capability is an extremely challenging task, as the training data often represents a tiny fraction of all the possible patterns. This is a main source of concern in high-dimensional problems, like those in computer vision, e.g., image classification, where covering the large variability across different samples gets unfeasible. In this regard, the advancements in deep learning, Convolutional Neural Networks (ConvNets) in particular, enabled unprecedented results [1]. Nonetheless, state-of-the-art ConvNets still suffer from accuracy drop when ported in real-life scenarios and operated on input patterns that differ substantially from those used at training time. For instance, the most common sources of misprediction are the discrepancy in size and orientation of the objects caught in the image [2], as well as different light conditions or contrast.

Several techniques proposed in the recent literature aim to improve model generalization operating both at training time [3], [4] and inference time [5], [6]. Among them, Test-Time Augmentation (TTA) is a valuable option for ConvNets hosted in the cloud and operated for visual tasks like image classification [7], [8], [9]. It is a simple yet efficient strategy that involves the aggregation of partial predictions over a set of transformed versions of the same input image. When implemented on high-performance architecture, the cost of multiple feed-forward passes is compensated through input batching, that is, the augmented images get processed in parallel with negligible overhead (see Table I). The same does not hold on the edge, where ConvNets are made run on mobile devices powered by low-power CPUs with limited resources [10][11]. Table I collects a quantitative comparison, showing that batch inference raises a prohibitive latency overhead on embedded CPUs, which in turn prevents the portability of TTA. Indeed, batch inference gets 5.5× (batch size=5) and 11.2× (batch size=10) slower than a single inference (batch size=1), which is even less efficient than sequential processing.

Table I: Inference latency (ms) of state-of-the-art ConvNets measured at different batch sizes (1, 5, and 10) on a cloud GPU (NVIDIA Titan Xp with 3840 CUDA cores) and an embedded CPU (ARM Cortex-A53 with 4 cores).

ConvNet	NVIDIA Titan Xp			ARM Cortex-A53		
	1	5	10	1	5	10
MobileNetV1	18.2	18.6	18.7	53.1	290.6	569.9
MobileNetV2	12.1	12.4	12.9	44.2	261.8	513.5
EfficientNet-B0	21.3	22.4	22.6	68.5	358.9	682.3
EfficientNet-B1	31.9	33.4	33.9	103.4	536.4	1290.2

Starting from these observations, this paper introduces *AdapTTA*, an adaptive implementation of TTA suited for embedded systems. Unlike static TTA strategies, where the number of modified samples fed to the ConvNet is fixed, AdapTTA self-regulates the number of transformations and feed-forward passes dynamically. The transformed images are generated and processed sequentially till the model achieves good confidence on the main outcome. Specifically, AdapTTA relies on the fact that different inputs come with different intrinsic complexity and the minimum number of transformations needed to reach an accurate classification changes on a sample basis. This suggests the number of feed-forward passes can be adjusted at run-time depending on the confidence level accumulated. In such a way, the processing gets faster for "easy" images, slower for the most "complex" ones. Leveraging the statistics of the input patterns, AdapTTA provides a substantial average speed-up compared to the original static approach.

AdapTTA was tested on four state-of-the-art ConvNets for image classification, taking into account two common TTA policies, namely, *5-Crops* and *10-Crops* [7], [8], [9], which refer to five and ten consecutive crops on the same image, respectively. To notice that the main objective here is not to find the transformations that reach the highest results, a research problem already addressed in previous works [12], [13], but rather to demonstrate the feasibility of the proposed

dynamic scheme for low-power applications. AdapTTA is orthogonal to the kind of input augmentation applied indeed. The experiments were thereby conducted on a commercial off-the-shelf embedded platform powered by an ARM Cortex-A53 CPU. Collected results show that AdapTTA reaches faster processing than static TTA, from 1.40× to 2.21× on average, still preserving the same accuracy gain. This demonstrates the improved portability and scalability of the method, which can be easily adapted to many edge applications without incurring any modification on the training pipeline.

II. Related Works

Data augmentation is key for training ConvNets. It consists of applying random transformations on the input data to increase the diversity of the training samples, with the final goal of improving the generalization capability of the model. The most simple implementations used in computer vision problems rely on a set of geometric transformations (e.g., translation, rotation, flipping) and graphical transformations (e.g., brightness, contrast, saturation), often hand-tuned by domain experts to match the conditions of real-life scenarios [7], [14]. More advanced strategies aim to automate the design of the augmentation policy, for instance, through a grid search exploration [4], or using faster-searching processes driven by reinforcement learning [3] and gradient-based methods [15]. Some of these strategies have been successfully integrated with the training of state-of-the-art ConvNets [1].

Data augmentation at training time is often not enough to handle the unpredictable changes in the data distribution [12], [16]. Therefore, TTA has been employed to increase the predictive performance of the model. TTA works at inference time, employing the transformations typically used in data augmentation. It aims to generate altered versions of the same input with a similar distribution of the training data, providing the model with more information. In practice, a set of modified samples is fed to the ConvNet, and the partial predictions are aggregated through majority voting or averaging. Similarly to data augmentation, the TTA policy can be hand-crafted [7], [8], [9] or discovered by automatic algorithms [12], [13]. Overall, TTA enables to improve the prediction accuracy [17] and the robustness against adversarial attacks [18].

Regardless of the transformations adopted, the existing TTA policies have been conceived and validated on ConvNets running in the cloud, where even a very large set of input transformations can be efficiently distributed over the extensive parallelism of GPUs. The implementation of TTA on embedded platforms cannot leverage such parallelism, and the efficiency on low-power devices is a less explored problem, which is the target of this work.

III. Adaptive Test-Time Augmentation

All the existing TTA policies share the following limitation: they apply a fixed and predefined number of transformations to each input data, namely, they are static. This represents a major bottleneck for the adoption of TTA on embedded systems. A more detailed view of the execution flow of a generic TTA policy is depicted in Figure 1. It refers to an image classification problem involving C classes. First, a set of N altered versions x' of the input image x is generated through the application of a set of transformation $\mathcal{T} : x \rightarrow x'$. Second, the generated images are processed by the ConvNet in parallel or sequence (more details in Section V). Third, the N partial predictions are aggregated to compute the final outcome. The parameter N is defined at design time by the TTA policy, hence each prediction encompasses the same number of inferences for each input image.

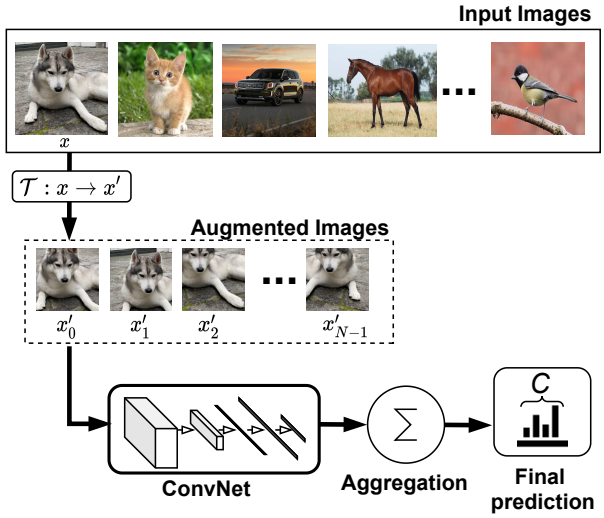


Figure 1: Flow diagram of static TTA policy for an image classification task.

Such a static strategy might be too conservative for most of the samples, especially for certain inputs with key features well exposed and easy to be detected. AdapTTA has been conceived to fulfill a simple objective: implement a more flexible TTA mechanism monitoring intermediate results to apply the lowest number of transformations in $\mathcal{T}$ that allows to infer the correct output.

The schematic flow of AdapTTA is illustrated in Figure 2. Given an input image, modified versions are generated and fed to ConvNet iteratively. After each inference, the partial prediction probabilities p_i are aggregated through a class-wise average P_{avg} to compute a confidence score defined as follows:

$$CS = P_{\text{avg-1}} - P_{\text{avg-2}} \tag{1}$$

where the $P_{\text{avg-1}}$ and $P_{\text{avg-2}}$ denote the probability of the first and second highest scored classes respectively. If the confidence score satisfies a pre-defined confidence threshold τ ($CS > \tau$), the prediction is deemed reliable and the TTA loop stops. The final inference outcome is then returned by taking the highest probability in P_{avg}. The full set of augmented samples in $\mathcal{T}$ are evaluated only in the worst-case, i.e., if CS gets smaller than τ. In this case, AdapTTA returns the same prediction of the static TTA consuming the same computational effort. In other words, AdapTTA implements an adaptive mechanism to adjust the augmentation passes at run-time depending on the confidence level accumulated.

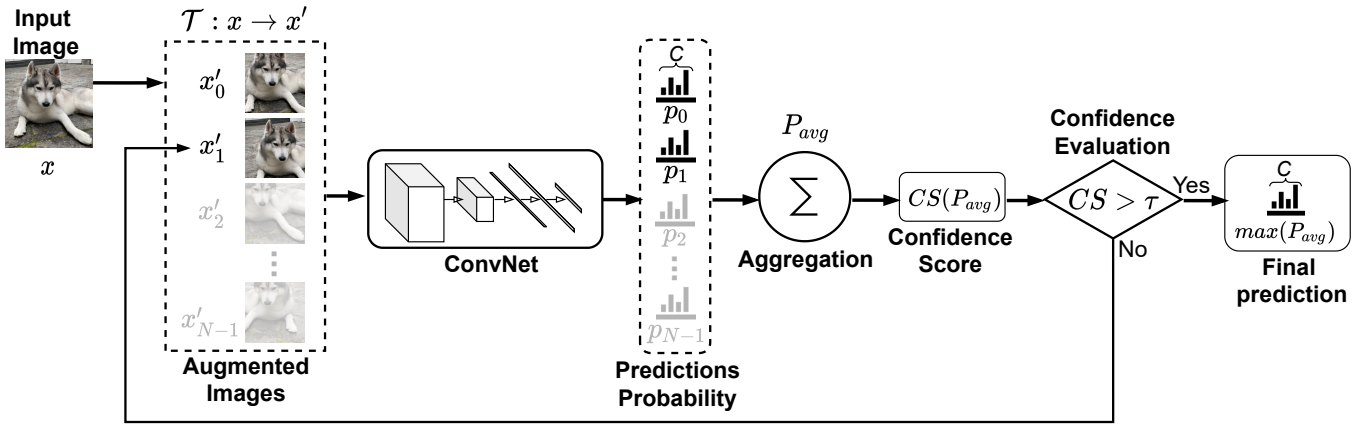


Figure 2: Example of the execution flow of AdapTTA. Augmented images are generated and fed sequentially to the ConvNet. After each iteration, the predictions are aggregated, and the confidence score is computed. Depending on its value, the following transformation is applied and evaluated, or the loop is interrupted. In the example, only x'_0 and x'_1 get processed by the ConvNet to compute the final prediction. The label with the highest probability score in P_{avg} (average between p_0 and p_1) is assigned to the input.

Even though AdapTTA relies on a heuristic method with no guarantee of optimality in identifying the smallest subset of transformations, our experiments validate its efficiency (see Section V). The main source of non-ideality is the metric adopted to evaluate the prediction confidence, i.e., the confidence score, which we borrowed from other optimization techniques operating at run-time [19], [20], [21]. Obviously, different aggregation functions do exists, such as majority voting, stacking [22], or Bayesian methods [23]. However, an average policy is more suited for resource-constrained devices as it requires negligible computational overhead. Further extensions of the proposed AdapTTA technique may integrate any transformation and aggregation function.

IV. Experimental Setup

This section describes the hardware platform used in the experiments along with the software environment adopted for the deployment. Moreover, we introduce the ConvNets taken as benchmarks and the TTA policies adopted in our comparative analysis.

A. Hardware Platform and Software Configurations

The hardware testbench is the Odroid-C2 platform powered with the Amlogic S905 SoC. The CPU is a quad-core ARM Cortex-A53 running @1.5GHz nominal frequency. The board runs Ubuntu Mate 18.04, kernel version 3.16.72-46, released by Hardkernel. The inference engine is TensorFlow Lite 1.14; it offers a collection of neural-network routines optimized to run on the ARM Cortex-A architecture. In our setup, TensorFlow Lite is cross-compiled using the GNU ARM Embedded Toolchain (version 6.5) [24].

B. ConvNet Benchmarks

We adopted pre-trained ConvNets available on TensorFlow Hosted Models [25] and TensorFlow Hub [26] repository. Specifically, we considered two representative families of models for the mobile segment: *MobileNet* [27], [28] and *EfficientNet* [1]. The networks are trained on the ImageNet dataset [29] and quantized to 8-bit, a common choice for edge inference as it ensures lower memory footprint and faster processing speed with negligible accuracy loss with respect to floating-point.

For each model family, we investigated two different versions for a total of four ConvNets listed in Table II. The column **Storage** 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 without TTA, i.e., with a standard pre-processing consisting of resizing the images to 256×256 pixels and extracting the central crop of size 224×224. The column $\boldsymbol{L}_{\textbf{nom}}$ reports the nominal latency of a single inference at maximum performance (4-threads @1.5GHz).

The adopted benchmarks are state-of-the-art for image classification on embedded systems. MobileNets are compact hand-crafted networks optimized for high performance and low memory. EfficientNets have been automatically designed via *neural architectural search* to improve the classification accuracy yet requiring more computational resources. For example, EfficientNet-B1 guarantees 5.1% higher accuracy than MobileNetV2 at the cost of $1.89\times$ larger memory and $2.33\times$ higher latency.

C. Augmentation Policies

We adopted two popular TTA policies [7], [8], [9], which are described as follows:

5-Crops (5C) - starting from a 256×256 image (the leftmost in Figure 3), five crops of size 224×224 are extracted (the center crop and the four corner crops (top-left, top-right, bottom-left and bottom-right).

10-Crops (10C) - is an extension of the 5C policy; it applies

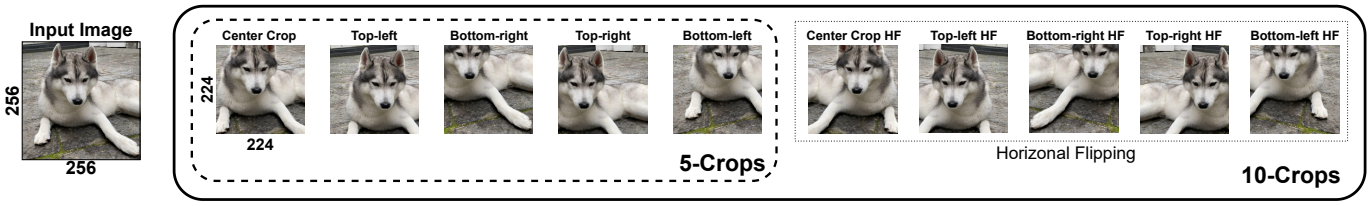


Figure 3: Example of 5-Crops and 10-Crops TTA policies. HF denotes the application of horizontal flipping.

Table II: Memory, nominal top-1 accuracy without TTA (Top-1), and nominal latency (L_{nom}) of the selected benchmarks.

ConvNet	Storage [MB]	Top-1 [%]	L_{nom} [ms]
MobileNetV1	4.3	70.0	53.1
MobileNetV2	3.4	70.8	44.2
EfficientNet-B0	5.4	74.4	68.5
EfficientNet-B1	6.4	75.9	103.4

the left-to-right horizontal flipping to the five crops of 5C for a total of 10 images (Figure 3).

These policies enable substantial improvements in the classification accuracy as reported Table III. Obviously, doubling the number of transformations (from 5C to 10C) increases the accuracy gain. Moreover, the transformations get processed efficiently even on resource-constrained devices: on the Cortex-A53 CPU, cropping requires only $0.8\,\text{ms}$ and horizontal flipping $0.9\,\text{ms}$.

Table III: Accuracy gain (in %) of 5-Crops (5C) and 10-Crops (10C) TTA policies compared to the nominal values for static TTA and AdapTTA ($\tau = 0.8$).

	Static TTA		AdapTTA	
ConvNet	5C	10C	5C	10C
MobileNetV1	2.7	3.1	2.7	3.1
MobileNetV2	2.2	2.9	2.2	2.9
EfficientNet-B0	1.1	1.3	1.1	1.3
EfficientNet-B1	2.2	2.5	2.2	2.5

V. Results

A. Efficiency of AdapTTA

To evaluate the benefits of AdapTTA, we fixed the confidence threshold $\tau = 0.8$ (maximum value is 1.0) for all the adopted ConvNets. Table III reports the accuracy gains achieved with this configuration. In all the benchmarks, AdapTTA ensures the same accuracy levels as static TTA. We point out that the selected value of τ could limit the potential savings of AdapTTA, as even lower values could be enough to achieve the same accuracy. However, we opted for this conservative choice to assess the feasibility of AdapTTA decoupling our analysis from the optimization of τ. For a more in-depth analysis on the confidence threshold, see Section V-B.

Table IV: Average number of inferences in AdapTTA for the 5-Crops (5C) and 10-Crops (10C) policies.

ConvNet	5C	10C
MobileNetV1	2.81	4.57
MobileNetV2	3.37	6.26
EfficientNet-B0	3.57	6.75
EfficientNet-B1	3.24	6.02

We compared the computational efficiency of a standard static TTA and AdapTTA measuring the average prediction rate (in FPS) across the ImageNet validation set (50k images). For the static TTA, we benchmarked two different implementations:

Batch-TTA - the augmented images get processed in parallel through batching (batch size is equal to the number of crops);
Seq-TTA - the augmented images get processed sequentially.

Collected results are summarized in Figure 4. As mentioned in Section I, batching turns out to be inefficient on embedded CPUs due to the low number of parallel cores (4 in the Cortex-A53), hence, Seq-TTA is slightly faster than Batch-TTA. Most importantly, AdapTTA enables substantial acceleration, with much faster prediction rates ranging from $1.40\times$ to $1.78\times$ in 5C and from $1.49\times$ to $2.21\times$ in 10C.

In MobileNetV1, AdapTTA on 10C outperforms Seq-TTA on 5C in both accuracy (+3.1% vs. +2.7%) and speed (4.05 FPS vs. 3.73 FPS). The reason can be inferred from Table IV, which reports the average number of inferences needed to run a prediction with AdapTTA. AdapTTA needs less than than 5 (4.57) inferences on average (row MobileNetV1, column 10C). Despite that, it achieves superior performance than a static 5C implementation, demonstrating that static TTA is too conservative in most cases and unreliable for less frequent complex inputs.

B. On the Optimality of Confidence Threshold

This section aims to assess the sensitivity of AdapTTA efficiency on the hyper-parameter τ. Even though we selected the same value ($\tau = 0.8$) for all the networks in the preliminary analysis of Section V-A, more precise control of τ could enable additional margins of optimization. The search of the optimal value should be conducted on a set of data disjoint from those used in the training and the evaluation of the ConvNets by developing a dedicated algorithm, a problem that we will investigate in future studies. Indeed, a too low value of τ can limit the accuracy gains of TTA, while a

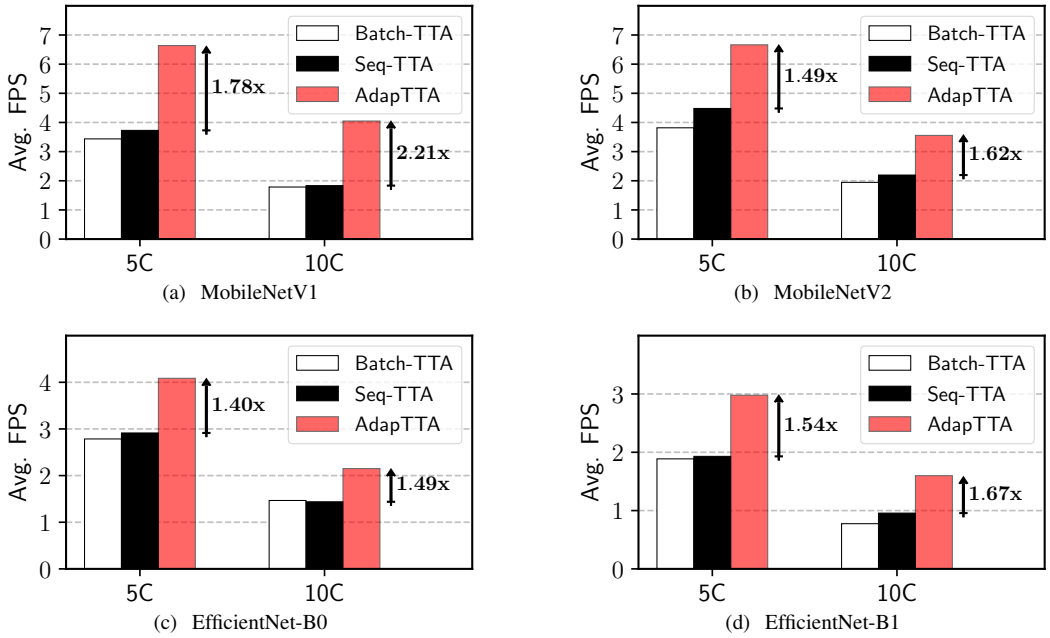


Figure 4: Average prediction rate (Avg. FPS, higher is better) for 5C and 10C policies of the static implementations (Batch-TTA and Seq-TTA) and AdapTTA. The reported data takes into account the execution time needed for both the transformations and the inference. The arrows indicate the relative speed-up of AdapTTA compared to Seq-TTA.

too high value can lower the prediction rate as unneeded transformations get processed. Here, we aim to quantify the maximum speed-ups that can be achieved while keeping the highest level of accuracy. For this purpose, we evaluated a discrete set of values of τ, ranging from 0.1 to 0.9, with a step of 0.1. The experiments were conducted on the ImageNet validation set.

Figure 5 summarizes the results collected on MobileNetV1 (top row) and EfficientNet-B1 (bottom row) with the 5C policy. Similar trends were observed for the other networks and policies, which we omitted for brevity. The main outcome of the analysis is that the minimum value of τ ensuring the highest accuracy gain differs across the selected benchmarks: 0.7 for MobileNetV1 and 0.5 for EfficientNet-B1. This translates to additional acceleration: in MobileNetV1, the prediction rate increases from 6.64 FPS ($\tau = 0.8$) to 7.28 FPS ($\tau = 0.7$) on average; in EfficientNet-B1, from 2.97 FPS ($\tau = 0.8$) to 4.16 FPS ($\tau = 0.5$). Besides a different topology, these networks followed a different training protocol, e.g., integrating different data augmentation pipelines [27], [1]. Future studies should therefore investigate the impact of the training hyper-parameters to understand if corrective actions applied at training time could reduce the number of transformations needed at test time, hence, improve the efficiency of AdapTTA.

Finally, τ could be easily tuned at run-time to enable a fine-grain trade-off between accuracy and speed. For example, with $\tau = 0.5$ MobileNetV1 reaches 8.7 FPS, yet with a marginal accuracy loss with respect to Seq-TTA ($< 0.5\%$). This can be helpful when the application has to rescale its energy footprint (e.g., if the mobile system is running out of battery) or when the classification task is not that critical to being solved (in which case some accuracy loss is tolerable).

VI. Conclusions

This work presented AdapTTA, an adaptive implementation of TTA suited for embedded systems powered by off-the-shelf CPUs. Differently from static strategies, AdapTTA aims to minimize number of feed-forward passes to output a correct prediction depending on the input complexity. Experimental results proved its efficiency across different families of ConvNets for the mobile segment and different TTA policies. Overall, AdapTTA reaches substantial acceleration, from to $1.40\times$ to $2.21\times$ compared to static TTA policies, with no loss of prediction accuracy.

References

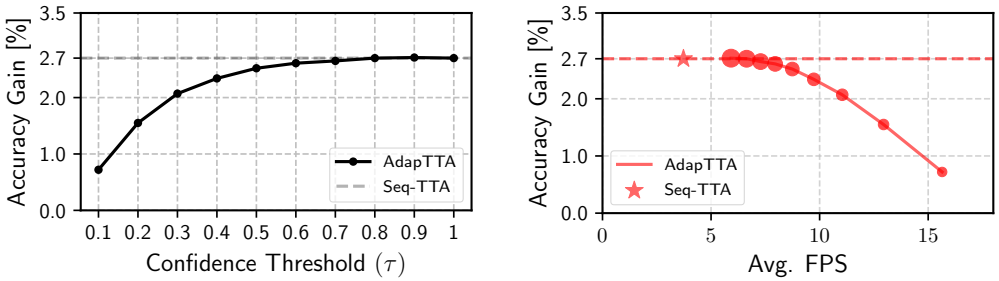
[1] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[2] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations*, 2019.

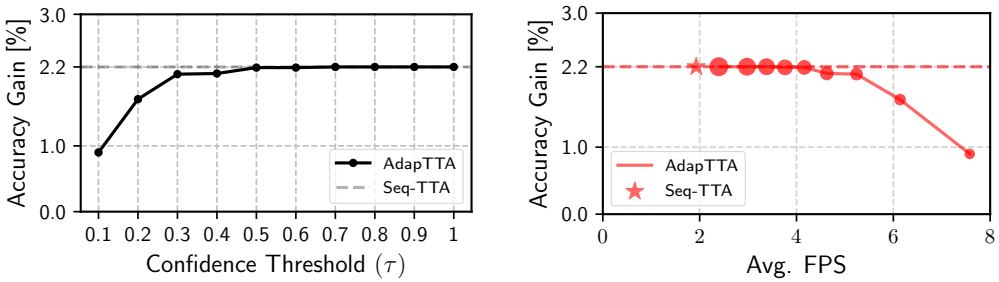
[3] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 113–123.

[4] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[5] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119. PMLR, 13–18 Jul 2020, pp. 9229–9248.



(a) MobileNetV1



(b) EfficientNetB1

Figure 5: Impact of the confidence threshold τ on AdapTTA for the 5C policy on MobileNetV1 (top) and EfficientNet-B1 (bottom). On the left, the accuracy gains (in %) at different values of τ. On the right, the accuracy gain vs. average prediction rate (Avg. FPS) at the same values of τ. The size of the red points is proportional to τ (larger size means higher τ).

[6] J. Kim, H. Kim, and G. Kim, "Model-agnostic boundary-adversarial sampling for test-time generalization in few-shot learning," *Computer Vision–ECCV*, pp. 599–617, 2020.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[8] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014*, 2014.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[10] 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.

[11] 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.

[12] A. Lyzhov, Y. Molchanova, A. Ashukha, D. Molchanov, and D. Vetrov, "Greedy policy search: A simple baseline for learnable test-time augmentation," in *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 124. PMLR, 03–06 Aug 2020, pp. 1308–1317.

[13] I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 4163–4174.

[14] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *Icdar*, vol. 3, no. 2003. Citeseer, 2003.

[15] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Faster autoaugment: Learning augmentation strategies using backpropagation," in *European Conference on Computer Vision*. Springer, 2020, pp. 1–16.

[16] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *International Conference on Machine Learning*. PMLR, 2019, pp. 5389–5400.

[17] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, "Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation," in *International MICCAI Brainlesion Workshop*. Springer, 2018, pp. 61–72.

[18] R. Volpi, H. Namkoong, O. Sener, J. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 5339–5349.

[19] E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, and S. Yoo, "Big/little deep neural network for ultra low power inference," in *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*. IEEE, 2015, pp. 124–132.

[20] L. Mocerino and A. Calimera, "Coopnet: Cooperative convolutional neural network for low-power mcus," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 414–417.

[21] ——, "Fast and accurate inference on microcontrollers with boosted cooperative convolutional neural networks (bc-net)," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 77–88, 2020.

[22] L. Breiman, "Stacked regressions," *Machine learning*, vol. 24, no. 1, pp. 49–64, 1996.

[23] K. Monteith, J. L. Carroll, K. Seppi, and T. Martinez, "Turning bayesian model averaging into bayesian model combination," in *The 2011 International Joint Conference on Neural Networks*. IEEE, 2011, pp. 2657–2663.

[24] Linaro toolchain. [Online]. Available: https://www.linaro.org/downloads/

[25] Tensorflow lite hosted models. [Online]. Available: https://www.tensorflow.org/lite/guide/hosted_models

[26] Tensorflow hub. [Online]. Available: https://tfhub.dev

[27] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.