POLITECNICO DI TORINO
Repository ISTITUZIONALE

On the Efficiency of AdapTTA: An Adaptive Test-Time Augmentation Strategy for Reliable Embedded ConvNets

Original
On the Efficiency of AdapTTA: An Adaptive Test-Time Augmentation Strategy for Reliable Embedded ConvNets / Mocerino, L; Rizzo, Rg; Peluso, V; Calimera, A; Macii, E (IFIP ADVANCES IN INFORMATION AND COMMUNICATION TECHNOLOGY). - In: VLSI-SoC: Technology Advancement on SoC Design / Grimblatt V., Hong Chang C., Reis R., Chattopadhyay A., Calimera A.. - Cham, Switzerland : SPRINGER INTERNATIONAL PUBLISHING AG, 2022. - ISBN 978-3-031-16817-8. - pp. 1-23 [10.1007/978-3-031-16818-5_1]

Availability:
This version is available at: 11583/2972827 since: 2022-11-11T09:03:06Z

Publisher:
SPRINGER INTERNATIONAL PUBLISHING AG

Published
DOI:10.1007/978-3-031-16818-5_1

Terms of use:

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

Publisher copyright
Springer postprint/Author's Accepted Manuscript

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-031-16818-5_1

(Article begins on next page)

06 May 2026

On the Efficiency of AdapTTA: an Adaptive Test-Time Augmentation Strategy for Reliable Embedded ConvNets

Luca Mocerino[1], Roberto G. Rizzo[1], Valentino Peluso[2], Andrea Calimera[1], and Enrico Macii[2]

[1] Department of Control and Computer Engineering
Politecnico di Torino, 10129 Turin, Italy
[2] Interuniversity Department of Regional and Urban Studies and Planning
Politecnico di Torino, 10129 Turin, Italy
{luca.mocerino,robertogiorgio.rizzo,valentino.peluso,
andrea.calimera,enrico.macii}@polito.it

Abstract. Test-Time Augmentation (TTA) is a popular technique that aims to improve the accuracy of Convolutional Neural Networks (ConvNets) at inference-time. TTA addresses a limitation inherent to any deep learning pipeline, that is, training datasets cover only a tiny portion of the possible inputs. For this reason, when ported to real-life scenarios, ConvNets may suffer from substantial accuracy loss due to unseen input patterns received under unpredictable external conditions that can mislead the model. TTA tackles this problem directly on the field, first running multiple inferences on a set of altered versions of the same input sample and then computing the final outcome through a consensus of the aggregated predictions. TTA has been conceived to run on cloud systems powered with high-performance GPUs, where the altered inputs get processed in parallel with no (or negligible) performance overhead. Unfortunately, when shifted on embedded CPUs, TTA introduces latency penalties that limit its adoption for edge applications. For a more efficient resource usage, we can rely on an adaptive implementation of TTA, *AdapTTA*, that adjusts the number of inferences dynamically, depending on the input complexity. In this work, we assess the figures of merit of the AdapTTA framework, exploring different configurations of its basic blocks, i.e., the augmentation policy, the predictions aggregation function, and the model confidence score estimator, suitable for the integration with the proposed adaptive system. We conducted an extensive experimental evaluation, considering state-of-the-art ConvNets for image classification, MobileNets and EfficientNets, deployed onto a commercial embedded device, the ARM Cortex-A CPU. The collected results reveal that thanks to optimal design choices, AdapTTA ensures substantial acceleration compared to a static TTA, with up to 2.21× faster processing preserving the same accuracy level. This comprehensive analysis helps designers identify the most efficient AdapTTA configuration for custom inference engines running on the edge.

Keywords: Test-Time Augmentation, Deep Learning, Embedded Systems

1 Introduction

1.1 Context

Convolutional Neural Networks (ConvNets) are the backbone of many computer vision applications, thanks to their ability to recognize complex data structures with good generalization capability. However, state-of-the-art ConvNets are far from the robustness of the human vision systems, which can deal with abstract changes in structure and style and are rarely misled by spatial changes in images or forms of corruption such as blur, snow, noise, and a combination of them. Achieving this level of generalization is an essential target for intelligent systems, especially in safety-critical applications. Still, current ConvNets suffer from accuracy drop when ported to real-life scenarios and operated on input patterns that differ substantially from those used at training time, which often represents only a limited subset of all the possible patterns. This issue gets critical in high-dimensional problems like image classification, for which covering the large variability across different data samples is unfeasible. For example, the most common sources of misprediction are the discrepancy in size and orientation of the objects caught in the image [1], as well as different light conditions or contrast.

The first actions to address this problem can be taken at training time. Among the possible options, data augmentation is one of the most common techniques, thanks to its straightforward integration in standard training pipelines. 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. The most simple implementations used in computer vision problems rely on a set of geometric and graphical transformations, often hand-tuned by domain experts to match the conditions of real-life scenarios [2,3]. More advanced strategies aim to automate the design of the augmentation policy, for instance, through a grid search [4], reinforcement learning [5], or gradient-based optimization [6]. Some of these strategies have been successfully integrated with the training of state-of-the-art ConvNets [7].

Despite these efforts, ConvNets may still fail to handle unpredictable changes in the data distribution [8, 9] in real-life scenarios. For a more robust generalization, recent works proposed complementary strategies operating at inference time [10, 11]. Among them, Test-Time Augmentation (TTA) is a valuable option for ConvNets hosted in the cloud and operated for visual tasks like image classification [2, 12, 13]. It is a simple yet efficient strategy that leverages multiple predictions to increase the model's confidence. Specifically, it involves the aggregation of partial predictions over a set of transformed versions of the same input image. In practice, the transformations applied are inspired by the data augmentation techniques typically adopted during training.

Different implementations of TTA exist, yet all of them have been validated only on high-performance platforms for cloud applications. In this work, we focus instead on the portability of TTA to inference engines running on embedded systems integrating low-power CPUs. This shift raises several challenges due to

Table 1: 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).

	NVIDIA Titan Xp			**ARM Cortex-A53**		
ConvNet	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
MobileNetV3	19.0	20.1	21.3	46.2	221.3	470.6
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
EfficientNet-B2	33.2	35.7	38.4	122.6	591.9	1360.4

the limited computational resources of embedded systems, as detailed in the following sub-section.

1.2 Motivations

Conventional TTA policies have been conceived for high-performance architectures like GPUs, which offer thousands of parallel processing cores. For example, a commercial device like the NVIDIA Titan XP hosts 3840 CUDA cores. These architectures enable to process multiple inputs in parallel with a single feed-forward pass, a procedure commonly called *batch inference*. When implemented on cloud GPUs, TTA relies on batch inference to process the augmented images with negligible performance overhead (see Table 1). The same does not hold on the edge, where ConvNets are made run on mobile devices powered by low-power CPUs with limited resources [14–16] (e.g., 4 cores in the ARM Cortex-A53). On low-power CPUs, a single image is enough to saturate all the available computing units. Table 1 demonstrates this observation with a quantitative comparison, showing that batch inference raises a prohibitive latency overhead on embedded CPUs, which in turn prevents the portability of TTA. Specifically, batch inference gets 5.5× (batch size=5) and 11.2× (batch size=10) slower than a single inference (batch size=1), therefore it is even less efficient than sequential processing.

In our recent work [17], we introduced *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 in the main outcome. In other words, it only runs those inferences that make the model confident enough about the prediction. Specifically, AdapTTA relies on the fact that different inputs have different intrinsic complexity and the minimum number of transformations needed to reach an accurate classification changes on a sample basis. This suggests that the number of feed-forward passes can be adjusted at run-time depending on the confidence

level accumulated. The processing gets faster for "easy" images and slower for the most "complex" ones. Leveraging the statistics of the input patterns, AdapTTA allows a substantial average speed-up compared to the original static approach.

1.3 Contributions

Starting from the findings of AdapTTA, we further investigate the design of a TTA framework for embedded ConvNets, exploring different implementations of its basic components and quantifying their impact on accuracy gain and performance. Specifically, the design and the optimization of AdapTTA involve three main choices: (i) the augmentation policy, i.e., the set of transformations to apply to the input image; (ii) the aggregation function, i.e., the method to combine the partial predictions; (iii) the confidence score estimator, i.e., a proxy to control the number of transformations needed for each input. For all three blocks, we consider different options borrowed from the literature, focusing on those configurations that fit the target of our adaptive strategy, i.e., systems with limited computing resources.

The remainder of this paper is organized as follows. After a brief description of data augmentation and TTA, we report the most recent advancements in cloud-based TTA policies (Section 2). We then introduce the architecture of AdapTTA, discussing the viable options for the implementation of augmentation policy, aggregation function, and confidence score (Section 3). To assess the figures of merit of AdapTTA, we considered two families of ConvNets for image classification, MobileNets and EfficientNets, running on a commercial off-the-shelf embedded platform powered by an ARM Cortex-A53 CPU (Section 4). The results collected from the comprehensive analysis of different AdapTTA configurations guide designers towards the understanding of the best practices for an efficient porting of AdapTTA to embedded platforms (Section 5). Finally, a summary of the main achievements concludes the work (Section 6)

2 Background & Related Work

2.1 Data Augmentation for Training

The main bottleneck for training reliable ConvNets lies in imperfections in the data. The most critical aspects to consider include (i) domain mismatch when the data used for training differs from that processed on the field, (ii) data bias when the data is imbalanced towards specific classes or categories, (iii) data noise when the data is cluttered or corrupted.

Data augmentation is one of the simplest solutions to deal with these problems. It consists of adding additional training data through the application of random transformations on the available training samples. In computer vision tasks, the most popular augmentation procedures involve a set of geometric transformations (e.g., translation, rotation, flipping) and color transformations (e.g., brightness, contrast, saturation) that try to reproduce the conditions of

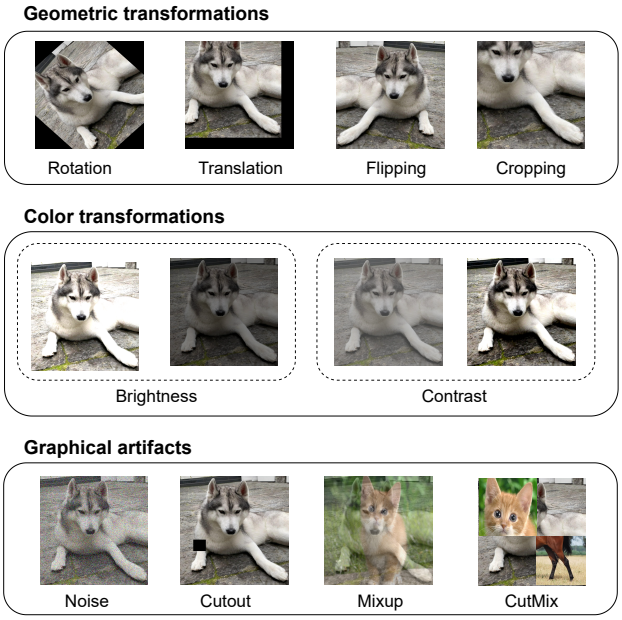


Fig. 1: Example of data augmentations for an image classification task.

the application domain [2, 3] (Figure 1). More sophisticated techniques introduce graphical artifacts injecting random noise, masking random regions on the input (*Cutout* [18]), or mixing multiple samples in a single image (*Mixup* [19] and *CutMix* [20]). The generated samples help the model learn features that make the classification more robust to changes in objects' position, lighting conditions, and scales.

Common training pipelines combine multiple transformations to further increase the diversity of data. The set of the transformations selected defines the augmentation policy. At each training iteration, a random subset of these transformations are applied sequentially to the original data. Augmentation policies can be hand-crafted or built with automatic techniques. For example, the optimal selection can be driven by a random search engine to adapt the augmentation policy to different contexts [4]. In general, automatic solutions outperform manual designs, motivating their integration in the training flow of state-of-the-art ConvNets [7, 21].

Rather than transforming the original input, alternative solutions are proposed to extend the training dataset with synthetic images that preserve the features of the original data. These solutions rely on generative models, like Variational Autoencoders [22] or Generative Adversarial Networks [23], that are trained on the available samples together with the classification model. Despite the potential benefits, the additional training operations generate a substantial computational overhead, which hinders the adoption of these methods.

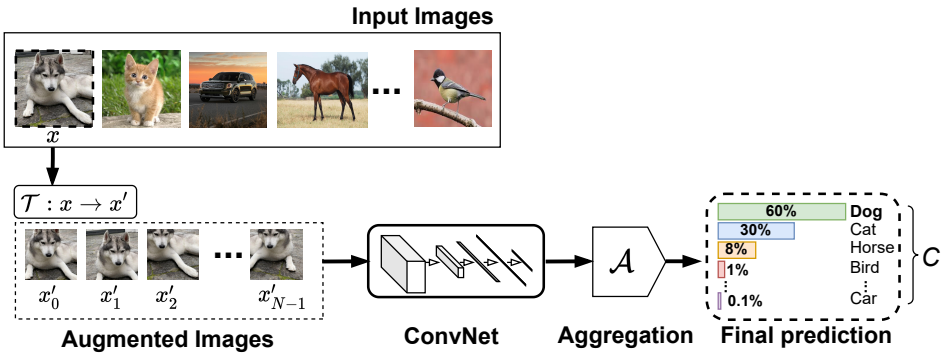


Fig. 2: Flow diagram of static TTA policy for an image classification task.

2.2 Test-Time Augmentation

Even if trained with complex augmentation policies, ConvNets still remain susceptible to unpredictable changes in the data distribution due to the shift from laboratory conditions to real-life scenarios [9]. TTA has emerged as a common strategy integrated with prediction services hosted in the cloud to increase the model robustness. In practice, TTA employs the same transformations for data augmentation to generate altered versions of an input sample. The generated instances are fed to the ConvNet, and the partial predictions are aggregated to compute the outcome. The rationale behind this process is that an altered version of the same input data increases the information contents provided to the model, improving the decision-making process.

Like data augmentation, most research efforts to optimize TTA focused on the search for the transformations that maximize the accuracy gains. Early works in the literature adopted hand-crafted policies based on basic spatial transformations such as image cropping & flipping and input resolution rescaling [2, 12, 13, 24]. More recent studies investigated algorithms for the automatic design of the TTA policy. For example, the selection of the transformations can be driven by a greedy exploration [8] or even tailored to each input sample [25]. However, automatic methods share an import shortcoming, that is, they require the training of additional modules or the re-training of the entire ConvNets.

Regardless of the transformations adopted, the major limitation of all the TTA implementations lies in their *static* behavior: they apply a predefined number of transformations to each input data without discriminating their features and complexity. Figure 2 shows a more detailed view of the execution flow of a generic TTA strategy. It depicts an image classification problem involving C classes. First, a set of N augmented versions x' of the input image x is generated through the application of a set of transformations included in the augmentation policy $\mathcal{T} : x \to x'$. Second, the generated images are fed to the ConvNet

in parallel or sequentially (more details in Section 5). Third, the N outputs are processed by a softmax layer to score the available labels. Finally, the resulting partial predictions are aggregated through a function $\mathcal{A}$ that returns the final outcome. The parameter N is fixed at design time by the TTA policy, therefore each prediction encompasses the same number of inferences for each input image.

Existing TTA policies address a single optimization goal that is the accuracy gain. They have been conceived and integrated with ConvNets running on cloud systems, which can process a high number of transformations still without saturating the available processing units thanks to the extensive parallelism of GPUs. On the contrary, embedded systems cannot offer comparable levels of parallelism, and even the inference of a single image requires the full utilization of resources. Besides accuracy, latency is an important variable to consider for the efficiency of TTA on low-power devices. This is a less explored problem, which motivated the design of AdapTTA.

3 Adaptive Test-Time Augmentation

Static TTA policies might be too conservative for most input samples, especially for specific inputs with well-exposed features that ConvNets can spot with a single or few feed-forward passes. Therefore, we conceived AdapTTA with a specific goal: provide a more flexible TTA mechanism that monitors intermediate predictions to minimize the number of transformations needed to return a reliable classification.

The schematic flow of Figure 3 illustrates the working principle of AdapTTA. The flow is iterative, and the number of iterations changes on a sample basis depending on the level of *confidence* of the classification. In each iteration, the ConvNet takes as input an altered version of the original data x'_i, which is generated with a transformation defined in the augmentation policy $\mathcal{T}$. Then, the ConvNet returns a partial prediction p_i, which contains the probabilities over the C classes. The partial predictions are aggregated class-wise after each inference using the aggregation function $\mathcal{A}(p_i)$. The resulting probability distribution P_A is evaluated with the confidence score CS to decide whether to process to the next transformation or return P_A to infer the final output. Specifically, if the confidence score satisfies a user-defined threshold τ, i.e., $CS > \tau$, the prediction is deemed reliable, and the TTA loop ends. The class with the largest probability in P_A is then selected as the label of the input image. In other words, AdapTTA implements an adaptive mechanism to control the augmentation passes at runtime based on the confidence level accumulated across repeated inferences. In the worst-case scenario, namely, if CS falls below the threshold τ for each iteration, the entire set of augmented samples extracted from the policy $\mathcal{T}$ is evaluated. In this case, AdapTTA delivers the same predictions as the static TTA, with the same computing effort and accuracy gain.

The flow depicted in Figure 3 is kept general to underline that, in principle, AdapTTA is compatible with different augmentation policies, aggregation functions, and confidence scores. However, the design and optimization of these

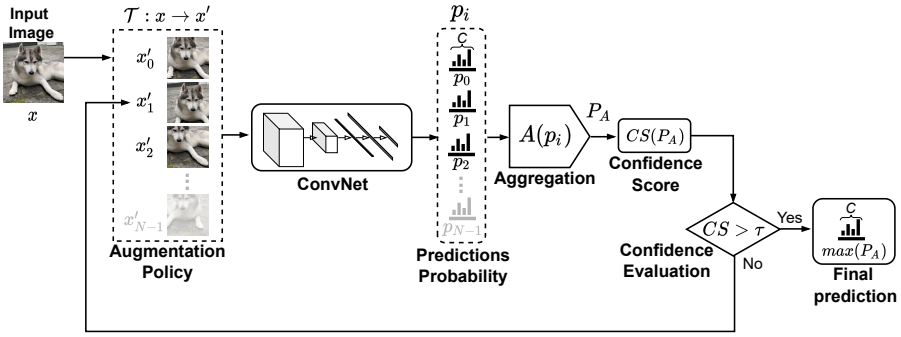


Fig. 3: AdapTTA schematic flow. Augmented images are generated and fed sequentially to the ConvNet. After each iteration, the predictions are aggregated, and the confidence score is computed. Depending on its value, the following transformation is applied and evaluated, or the loop is interrupted. In the example, only x'_0, x'_1, and x'_2 get processed by the ConvNet to compute the final prediction. The label with the largest probability in P_{A} is assigned to the input.

components are paramount to build an efficient adaptive scheme that maximizes the accuracy gain with minimum computational effort. Regarding the augmentation policies and aggregation functions, we considered solutions already adopted in static TTA strategies. The confidence score, on the contrary, is the fundamental component that distinguishes AdapTTA from the static approach, as it regulates the dynamic behavior of the proposed flow. For such reason, it is critical to identify a good proxy to evaluate the confidence of a model prediction. For such purpose, we considered different metrics that investigated the level of correctness of a classification taken from the recent literature [26–30]. Compared to these works, the novelty of our contribution lies in the application of the confidence score for the optimization of TTA.

Among the possible design choices for the above mentioned blocks, only a subset of them is compliant with the systems having limited computing resources. Therefore, we conducted our analysis considering the portability ops such blocks to embedded systems as a primary constraint (more details in the following subsections).

3.1 Augmentation Policy

The augmentation policy $\mathcal{T}$ defines the set of transformations that generate N different versions of the input image. In resource-constrained environments, the design of the augmentation policies should follow two important considerations. First, the augmentation policy should keep N as small as possible, as larger values of N imply more network feed-forward passes, which can affect both latency and power dissipation [31]. Second, the execution time needed to process a transformation should be negligible compared to that needed for inference.

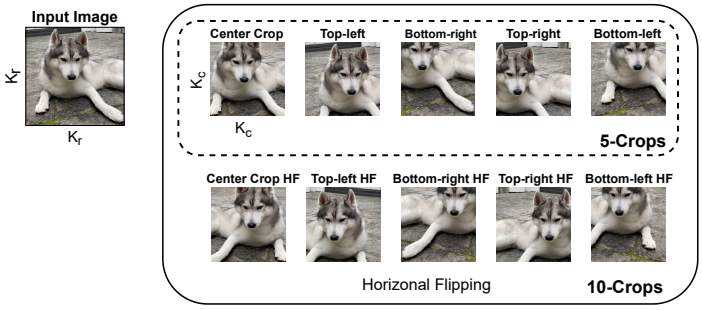


Fig. 4: Example of 5-Crops and 10-Crops TTA policies. HF denotes the application of horizontal flipping.

Following these observations, we considered only simple spatial manipulations, i.e., cropping & flipping. We integrated them into two TTA policies, *5-Crops* and *10-Crops*, inspired by the early implementations of TTA [2,12,13,24]. These two policies fit our design target. On the ARM Cortex-A53 CPU, cropping requires only 0.8 ms and horizontal flipping 0.9 ms, which is negligible compared to the tens of ms needed for network inference. Specifically, the two policies can be described as follows:

5-Crops (5C) - This policy takes as input a $K_\mathrm{r} \times K_\mathrm{r}$ image (the leftmost in Figure 4) and extracts consecutively a set of five crops of size $K_\mathrm{c} \times K_\mathrm{c}$, with $K_\mathrm{c} < K_\mathrm{r}$, from different areas of the input image. Specifically, it returns the center crop and the four corner crops (top-left, top-right, bottom-left and bottom-right).

10-Crops (10C) - This policy is an extension of the 5C policy; it applies the left-to-right horizontal flipping to the five crops of 5C for a total of 10 images (Figure 4). Doubling the number of transformations (from 5C to 10C) should increase the accuracy gain at the cost of a higher overall inference latency.

3.2 Aggregation Function

The aggregation function $\mathcal{A}(p_i)$ defines how to combine the partial predictions p_i generated at the different iterations of the flow in Figure 3. The study in [26] reported the most common implementations adopted in cloud-based TTA, *Max* and *Mean* aggregation. Although introduced for the cloud, we imported these two functions to our design target. Their execution, consisting of simple arithmetic operations, is negligible compared to the intensive workloads of ConvNets, and makes these functions a good fit for systems with limited computing resources. We summarize them as follows:

Max Aggregation - The *Max* function selects the distribution p_i that contains the class with the largest score among all the partial predictions. Therefore, the outcome of this function rewards only a single prediction and discards the contribution of the other ones.

Mean Aggregation - The *Mean* function performs the class-wise average of the partial predictions p_i. Different from *Max*, this aggregation function gives the same importance to all the partial predictions. For higher accuracy, it is possible to apply a weighted average, where the weights are trained with a dedicated procedure [26]. However, this procedure requires additional calibration data, which might be unavailable in the deployment phase. Therefore, our analysis considers only the arithmetic average.

In static TTA, the aggregation is performed after processing all the transformations defined in the augmentation policy. In AdapTTA, the aggregated probability P_A is instead updated after every inference to evaluate the classification confidence at the end of each iteration.

3.3 Confidence Score Estimator

The confidence score estimates the correctness of the classification and regulates the dynamic behavior of AdapTTA. A proper definition of the confidence score should guarantee two essential properties: (i) an high confidence value should correspond to a correct prediction, avoiding early stops of AdapTTA; (ii) low confidence should be assigned only to those inputs that need more transformations for committing the correct prediction, avoiding waste of computing resources.

Estimating the classification confidence of a ConvNet is not a new research problem; still, it remains an open issue. Among the viable options, we considered the following approaches:

MaxP - This function selects as confidence score the largest probability in P_A [26], over the C classes. This function considers only the top-1 probability, thus it may be misleading if all class probabilities have similar values.

Score Margin (SM) - It denotes the difference between the first and second largest probabilities over the C classes contained in P_A [27, 28].

$$SM = P_{A_{top1}} - P_{A_{top2}} \tag{1}$$

Intuitively, low values of SM denote that the model is uncertain between the two most likely classes.

Entropy - In information theory and statistics, entropy represents the informative content of a random variable [32]. For this reason, it has been adopted as an uncertainty metric in many deep learning problems like active learning [29]

and unsupervised learning [33]. For the AdapTTA design, we computed the normalized entropy H_n [34] of P_A (over the C classes).

$$H = -\sum_{c=0}^{C-1} P_{Ac} \cdot log(P_{Ac}) \tag{2}$$

$$H_n = 1 - \frac{H}{log(C)} \tag{3}$$

In this way, the *Entropy* score ranges in $[0, 1]$ like the above mentioned confidence metrics, where lower values imply higher uncertainty and larger values indicate stronger confidence.

Similar to the aggregation functions, these confidence scores are suitable for low-power systems as they only require simple arithmetic operations with negligible computational overhead compared to the execution of a ConvNet.

4 Experimental Setup

This section describes the hardware platforms and the software environment for AdapTTA deployment. In addition, we discuss the ConvNets families used as benchmarks for the experiments.

4.1 Hardware Platform and Software Setup

The Odroid-C2 platform, powered by the Amlogic S905 SoC, serves as the hardware testbench. The CPU is a quad-core ARM Cortex-A53 with a nominal frequency of 1.5GHz. The board runs Ubuntu Mate 18.04 with Hardkernel's version 3.16.72-46. TensorFlow Lite 1.14 is the inference engine, and it includes a collection of neural-network procedures tailored for the ARM Cortex-A architecture. TensorFlow Lite is cross-compiled in our environment using the GNU ARM Embedded Toolchain (version 6.5) [35].

4.2 ConvNet Benchmarks

The adopted benchmarks are pre-trained models from TensorFlow Hub [36] and TensorFlow Hosted Models [37] repository. Specifically, they belong to two families of ConvNets that represent the state-of-the-art for image classification tasks for the mobile segment: *MobileNets* [38–40] and *EfficientNets* [7]. All the models were trained on the ImageNet [41] dataset and quantized to 8 bits, which is a standard solution for edge inference as it provides a smaller memory footprint and faster processing with negligible accuracy loss when compared to floating-point.

Table 2 reports structural properties (memory footprint and latency) and functional (the classification accuracy) of the ConvNets under test. Specifically,

Table 2: Storage requirements, input resolution (K_c), top-1 accuracy without TTA (Top-1), and inference latency (L_{nom}) of the selected benchmarks.

ConvNet	**Storage [MB]**	$\mathbf{K_c}$	**Top-1 [%]**	$\boldsymbol{L_{\text{nom}}}$ **[ms]**
MobileNetV1	4.3	224	70.0	53.1
MobileNetV2	3.4	224	70.8	44.2
MobileNetV3	4.2	224	72.2	46.2
EfficientNet-B0	5.4	224	74.4	68.5
EfficientNet-B1	6.4	240	75.9	103.4
EfficientNet-B2	6.9	260	77.0	122.6

the column **Storage** collects the size (in MB) of the ConvNet in `.tflite` format, which includes the model weights and additional metadata (i.e., the topology description) to deploy the model on the target device. The metric **Top-1** refers to the top-1 classification accuracy measured on the ImageNet validation set, which consists of 50k images split into 1k different classes. The accuracy is evaluated without TTA, i.e., with a standard pre-processing pipeline consisting of resizing the images to a fixed resolution of $K_{\text{r}} \times K_{\text{r}}$ pixels and extracting the central crop of shape $K_{\text{c}} \times K_{\text{c}}$ ($K_{\text{r}} = K_{\text{c}} + 32$). Finally, the column $\boldsymbol{L_{\text{nom}}}$ reports the nominal latency of a single inference running at the maximum available resources (4 threads @1.5GHz).

5 Results

5.1 Design & Optimization of AdapTTA

The primary goal of any TTA strategy, static or adaptive, is to improve classification accuracy. For this reason, our first analysis aims to identify the most accurate static configurations that will serve as baselines to assess the quality of AdapTTA. In the static TTA, the design choices that impact the accuracy are the augmentation policy and the aggregation function. Therefore, we conducted an exhaustive exploration that considers all the possible combinations of the augmentation policies and aggregation functions under investigation.

Specifically, the results in Table 3 report the accuracy gain achieved by *Max* and *Mean* aggregation functions with the 5C and 10C policies for the entire benchmark suite. Regardless of the design choices, TTA improves the classification quality, yet with different benefits depending on the configuration. The accuracy gain ranges from 0.5% (MobileNetV3) to 2.70% (MobileNetV1) for 5C policy and from 0.9% (EfficientNet-B2) to 3.1% (MobileNetV1) for 10C policy. In general, a larger number of transformations brings higher accuracy. Moreover, the *Mean* aggregation function always outperforms *Max*, with relative improvements up to 1.2% (MobileNetV2 with 5C policy). The two functions only reach

Table 3: Accuracy gain (in %) of 5-Crops (**5C**) and 10-Crops (**10C**) TTA policies with **Max** and **Mean** aggregation functions.

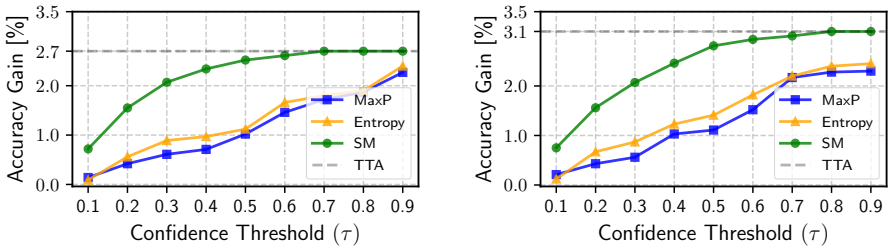
	5C		**10C**	
ConvNet	**Max**	**Mean**	**Max**	**Mean**
MobileNetV1	2.4%	**2.7%**	2.8%	**3.1%**
MobileNetV2	1.0%	**2.2%**	1.8%	**2.9%**
MobileNetV3	**0.5%**	**0.5%**	1.0%	**1.2%**
EfficientNet-B0	0.7%	**1.1%**	0.9%	**1.3%**
EfficientNet-B1	1.9%	**2.2%**	2.2%	**2.5%**
EfficientNet-B2	0.7%	**0.8%**	0.9%	**1.1%**

the same accuracy level for MobileNetV3 with the 5C policy. For two ConvNets (MobileNetV2 and EfficientNet-B0), *Mean* with 5C shows even a larger gain than *Max* with 10C, suggesting that the proper selection of the aggregation function enables a smaller number of transformations, retaining the same accuracy.

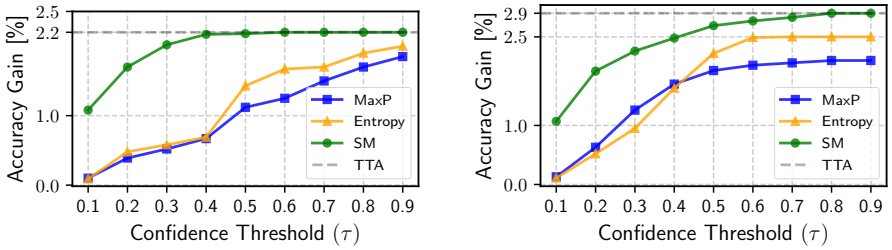
Our findings confirm the analyses conducted in previous studies like [26], which reported similar trends on a different set of networks and datasets. We believe that the *Max* function is susceptible to wrong classifications due to a partial prediction that erroneously overestimates a class probability. In these cases, the other predictions would be simply discarded. On the contrary, the *Mean* function mitigate this effect, as averaging over all the predictions can distribute the influence of outliers over the final decision. Motivated by these observations, we selected *Mean* as the aggregation function for both the static TTA used as a reference and the implementation of AdapTTA (more details about their comparison in Section 5.2).

As described in Section 3, the dynamic behavior of AdapTTA is controlled by the confidence score and the corresponding value of the confidence threshold τ (set as a hyper-parameter). We then focus on understanding which confidence estimator provides the most reliable evaluation of the classification correctness. For this purpose, we validated the three candidate functions (*MaxP*, *Entropy*, *SM*) in AdapTTA and we measured the accuracy gain for different values of $\tau \in [0.1, 0.9]$, with a step of 0.1. Notice that $\tau = 0$ is equivalent to classification without TTA, and $\tau = 1$ corresponds to the static TTA (all the transformations get processed).

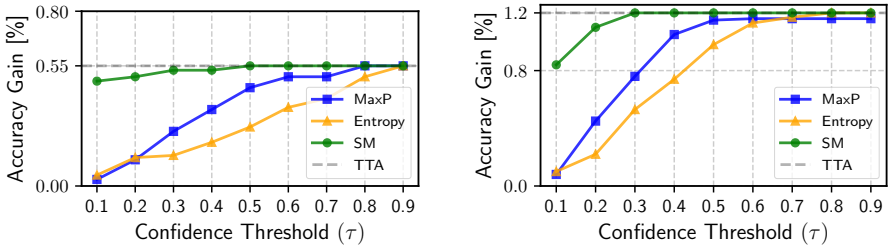
The results are reported in Figures 5 and 6 for the MobileNet and EfficientNet families, respectively. *SM* is the only metric that, with appropriate values of τ, ensures the same accuracy as the static TTA (dashed grey line in the plots). In general, *SM* always outperforms the other metrics, even at lower values of τ. The same trend holds for all the ConvNets and augmentation policies (5C and 10C). The most representative example is MobileNetV2 with the 5C policy (Figure 5-b left). In this case, *SM* keeps almost the maximum level of accuracy even with $\tau = 0.4$, while *MaxP* and *Entropy* reduce accuracy by 1.48%. In summary,



(a) MobileNetV1



(b) MobileNetV2



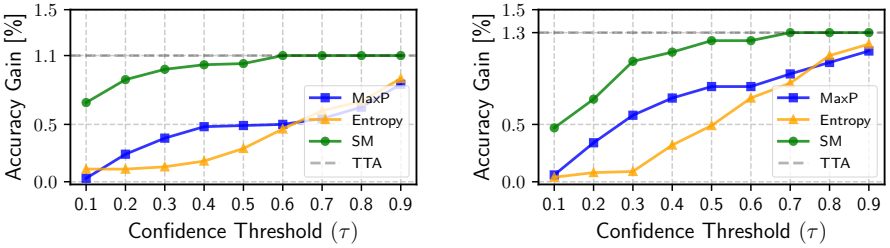
(c) MobileNetV3

Fig. 5: Accuracy gain of AdapTTA for different confidence scores (MaxP, Entropy, SM) using 5C policy (left) and 10C policy (right). Results on the MobileNets family.

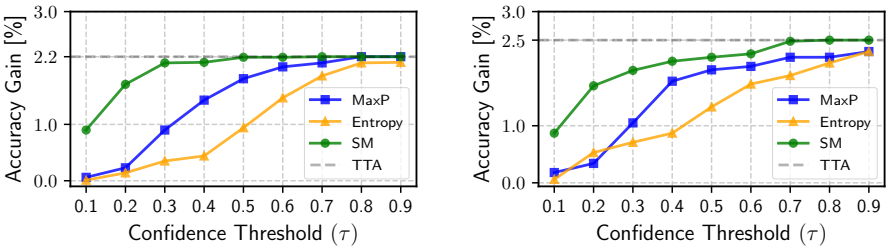
with *SM* the accuracy shows a lower sensitivity to the variations of τ. This is a desirable property, as lowering the value of τ could enable higher acceleration. Intuitively, if the classification is deemed correct even at "low" confidence levels, fewer transformations must be processed to return the final prediction (see Section 5.3 for more details). In MobileNetV3 with 10C policy (Figure 5-c right) and EfficientNet-B2 with 5C policy (Figure 6-c left), *SM* shows the lowest sensitivity to τ: the accuracy gain quickly saturates to the maximum level of accuracy starting from $\tau \geq 0.3$.

5.2 Comparing Static TTA and AdapTTA

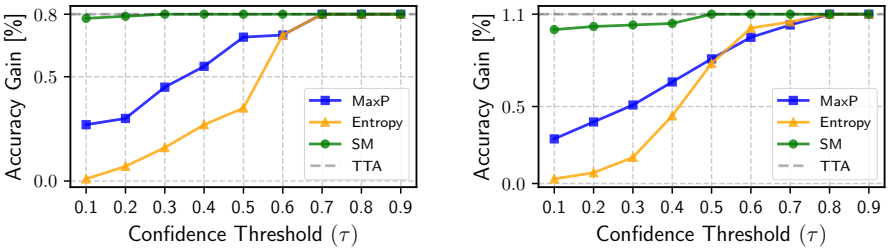
We compared the computational efficiency of a standard static TTA and AdapTTA, measuring the average prediction rate (in FPS) across the ImageNet validation



(a) EfficientNet-B0



(b) EfficientNet-B1



(c) EfficientNet-B2

Fig. 6: Accuracy gain of AdapTTA for different confidence scores (MaxP, Entropy, SM) using 5C policy (left) and 10C policy (right). Results on the EfficientNets family.

set (50k images). For the static TTA, we benchmarked two different implementations:
Batch-TTA - the augmented images get processed in parallel through batching (the batch size is equal to the number of transformations);
Seq-TTA - the augmented images get processed sequentially.
The overall inference time includes the data augmentations latency measured on the target device (0.8 ms for cropping and 0.9 ms for horizontal flipping).

In all cases, we considered the *Mean* aggregation, and we studied both the 5C and 10C policies. For AdapTTA, we fixed $\tau = 0.8$ to ensure the same accuracy gain of the static TTA, although this high value could limit the potential accel-

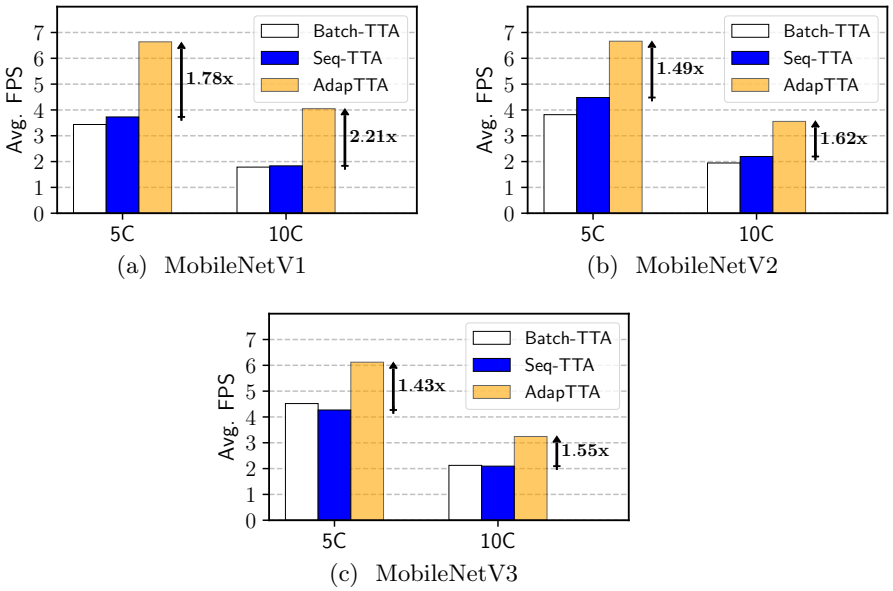


Fig. 7: Average prediction rate (Avg. FPS - the higher, the better) for 5C and 10C policies of the static implementations (Batch-TTA and Seq-TTA) and AdapTTA. The arrows indicate the relative speed-up of AdapTTA compared to Seq-TTA. Results on the MobileNets family.

eration of AdapTTA. However, we opted for this conservative choice to assess the feasibility of AdapTTA decoupling our analysis from the optimization of τ.

Figures 7 and 8 summarize the collected results for the two families of benchmarks. As mentioned in Section 1, 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. Also, AdapTTA enables substantial acceleration, with much faster prediction rates ranging from 1.16× to 1.78× in 5C and from 1.19× to 2.21× 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 4, which reports the average number of inferences needed to run a prediction with AdapTTA. AdapTTA needs less than 5 (4.57) inferences on average (row MobileNetV1, column 10C), achieving superior performance than a static 5C implementation. A comprehensive analysis on all the benchmarks shows that the average number of images ranges from 2.81 to 4.32 for 5C and from 4.57 to 8.41 for 10C at the same accuracy level, demonstrating that static TTA is too conservative in most cases and unreliable for less frequent complex inputs.

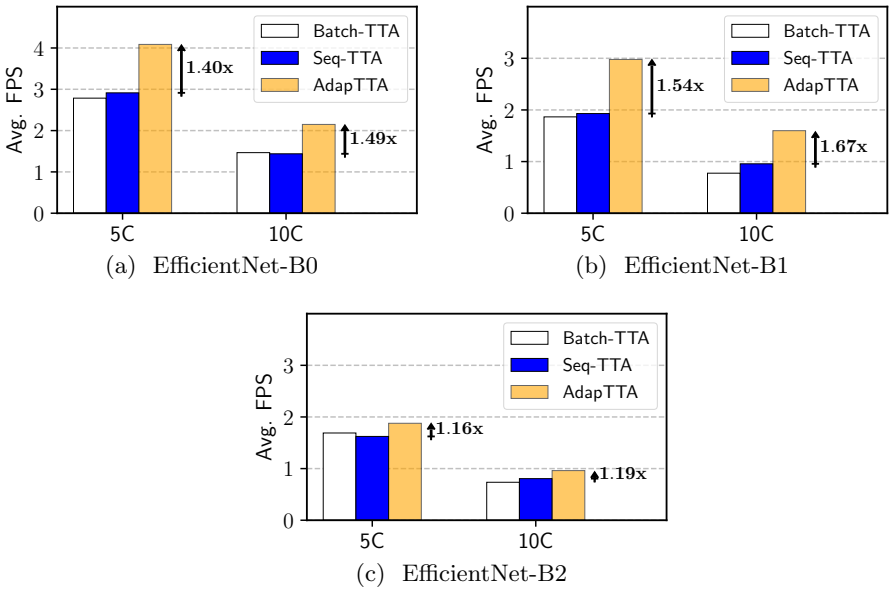


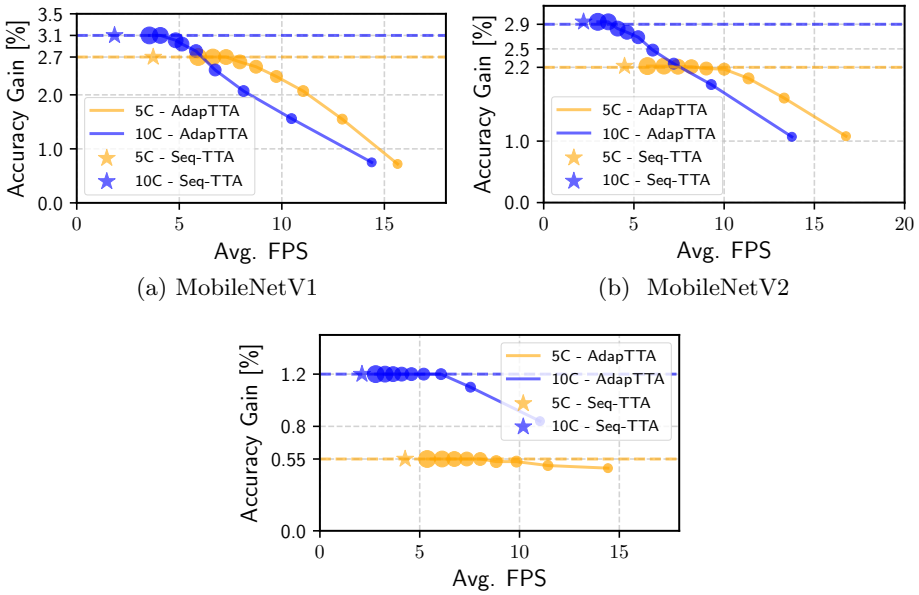
Fig. 8: Average prediction rate (Avg. FPS - the higher, the better) for 5C and 10C policies of the static implementations (Batch-TTA and Seq-TTA) and AdapTTA. The arrows indicate the relative speed-up of AdapTTA compared to Seq-TTA. Results on the EfficientNets family.

Table 4: 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
MobileNetV3	3.48	6.54
EfficientNet-B0	3.57	6.75
EfficientNet-B1	3.24	6.02
EfficientNet-B2	4.32	8.41

5.3 Accuracy vs. Performance Trade-offs

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 5.2, more precise control of τ could enable additional margins of optimization. Indeed, a too low value of τ can limit the accuracy gains of AdapTTA, while a 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 retaining the



(a) MobileNetV1 (b) MobileNetV2

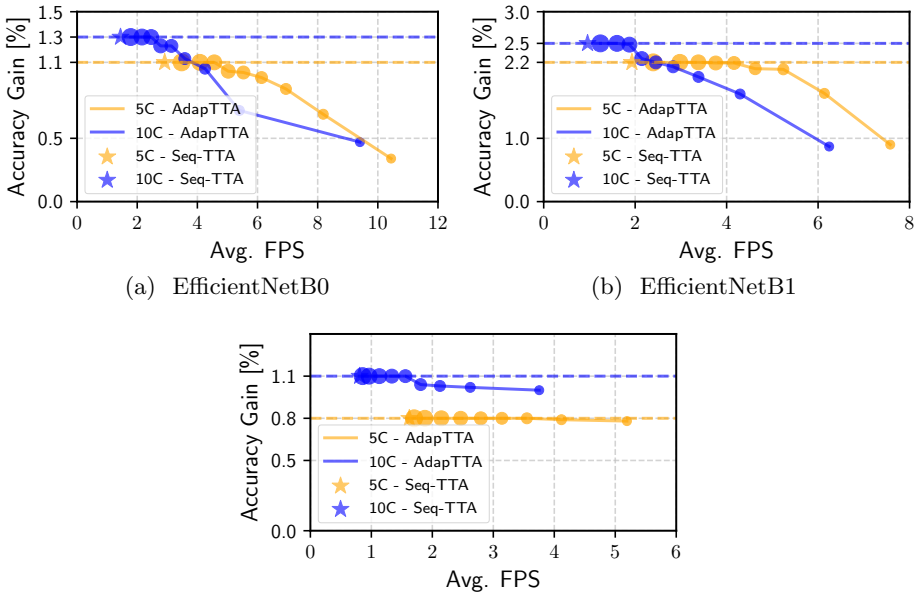
(c) MobileNetV3

Fig. 9: Accuracy gain (in %) vs. average prediction rate (Avg. FPS) at different values of τ for 5C and 10C policy. The size of the circles is proportional to τ (a larger size indicates a higher τ). Results on the MobileNets family.

maximum 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.

The main outcome of the analysis is that the minimum value of τ ensuring the highest accuracy gain differs across the selected benchmarks: from 0.7 for MobileNetV1 in 5C policy to 0.3 for MobileNetV3 in 10C policy (Figure 9). This translates to additional acceleration: in MobileNetV1 5C policy, the prediction rate increases from 6.64 FPS ($\tau = 0.8$) to 7.28 FPS ($\tau = 0.7$) on average; in MobileNetV3 10C policy, from 3.26 FPS ($\tau = 0.8$) to 6.07 FPS ($\tau = 0.3$). Similar trends do hold for the EfficientNet family (Figure 10). Besides a different topology, these networks followed a different training protocol, e.g., integrating different data augmentation pipelines [7, 38]. This observation suggests that training hyper-parameters can affect the efficiency of AdapTTA, and potentially corrective actions applied at training time could reduce the number of transformations needed at test time.

Moreover, the circles in the plots represent different operating points that can be selected at run-time to enable a fine-grain trade-off between accuracy and speed. This can be helpful when the application has to rescale its energy footprint (e.g., together with DVFS [42], if the mobile system is running out of



(a) EfficientNetB0 (b) EfficientNetB1

(c) EfficientNetB2

Fig. 10: Accuracy gain (in %) vs. average prediction rate (Avg. FPS) at different values of τ for 5C and 10C policy. The size of the circles is proportional to τ (a larger size indicates a higher τ). Results on the EfficientNets family.

battery) or when the classification task is not a critical application (some accuracy loss is tolerable). For example, with $\tau = 0.5$, MobileNetV1 reaches 8.7 FPS, yet with a marginal accuracy loss with respect to Seq-TTA ($< 0.5\%$). Also, a more in-depth analysis of the collected results reveals an interesting relationship between the value of τ and the policy selection. In all the networks except MobileNetV3 and EfficientNet-B2, the 10C (blue) and 5C (yellow) curves show a point of intersection. This point qualitatively delimits the boundary of two working conditions, *high-accuracy* on the left and *high-performance* on the right. For high-accuracy, 10C always outperforms 5C, while, for high-performance, the opposite consideration holds. This is due to the rapid drop in the accuracy observed in the 10C policy at lower values of τ. In summary, the maximum efficiency can be reached only with the joint optimization of τ and the augmentation policy. However, MobileNetV3 (Figure 9-c) and EfficientNet-B2 (Figure 10-c) show different trends. As mentioned, the accuracy of these networks is less sensitive to the variations of τ, resulting in almost flat curves in the accuracy vs. performance space. Specifically, the 10C curve never intersects the 5C curve, indicating that the 10C policy is the most practical choice for these networks. The 5C policy could be taken into account only for smaller values of τ (the two rightmost points in this case), which can still guarantee high-performance

operating conditions with limited accuracy loss ($< 0.4\%$ for MobileNetV3 and $< 0.7\%$ for EfficientNet-B2).

6 Conclusions

AdapTTA introduced a dynamic implementation of TTA targeting low-power applications deployed on embedded systems. Specifically, AdapTTA minimizes the number of augmented samples to process, with the final goal of reducing the computational effort while preserving the same accuracy benefits. To validate the efficiency of AdapTTA, we conducted a comprehensive analysis of different components and configurations of the proposed framework. We explored different TTA policies, aggregation functions, and confidence scores, assessing their impact on accuracy and performance. Our analyses serve as practical guidelines for designers and end-users to identify the most efficient configuration. Moreover, extensive experiments on a large variety of benchmarks reveal that AdapTTA reaches substantial acceleration, from $1.16\times$ to $2.21\times$ compared to static TTA policies, with no loss of prediction accuracy.

References

1. D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.
2. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 2012, pp. 1106–1114.
3. P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*. IEEE Computer Society, 2003, pp. 958–962.
4. E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 3008–3017.
5. E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 113–123.
6. R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Faster autoaugment: Learning augmentation strategies using backpropagation," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, ser. Lecture Notes in Computer Science, vol. 12370. Springer, 2020, pp. 1–16.

7. M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 6105–6114.
8. A. Lyzhov, Y. Molchanova, A. Ashukha, D. Molchanov, and D. P. Vetrov, "Greedy policy search: A simple baseline for learnable test-time augmentation," in *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, Virtual, August 3-6, 2020*, ser. Proceedings of Machine Learning Research, vol. 124. AUAI Press, 2020, pp. 1308–1317.
9. B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 5389–5400.
10. Y. Sun *et al.*, "Test-time training with self-supervision for generalization under distribution shifts," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual, 13-18 July 2020*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 9229–9248.
11. J. Kim, H. Kim, and G. Kim, "Model-agnostic boundary-adversarial sampling for test-time generalization in few-shot learning," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 12346. Springer, 2020, pp. 599–617.
12. 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, Conference Track Proceedings*, 2014.
13. C. Szegedy *et al.*, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 1–9.
14. 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 *Sixth International Conference on Social Networks Analysis, Management and Security, SNAMS 2019, Granada, Spain, October 22-25, 2019*. IEEE, 2019, pp. 454–459.
15. 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.
16. M. Grimaldi, V. Peluso, and A. Calimera, "Optimality assessment of memory-bounded convnets deployed on resource-constrained risc cores," *IEEE Access*, vol. 7, pp. 152 599–152 611, 2019.
17. L. Mocerino, R. G. Rizzo, V. Peluso, A. Calimera, and E. Macii, "Adaptta: Adaptive test-time augmentation for reliable embedded convnets," in *29th IFIP/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2021, Singapore, Singapore, October 4-7, 2021*. IEEE, 2021, pp. 1–6.
18. T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.
19. H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
20. S. Yun *et al.*, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *2019 IEEE/CVF International Conference on Computer*

Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. IEEE, 2019, pp. 6022–6031.
21. M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual, 18-24 July 2021*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 10 096–10 106.
22. J. Jorge, J. Vieco, R. Paredes, J. Sánchez, and J. Benedí, "Empirical evaluation of variational autoencoders for data augmentation," in *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 5: VISAPP, Funchal, Madeira, Portugal, January 27-29, 2018*. SciTePress, 2018, pp. 96–104.
23. A. Antoniou, A. J. Storkey, and H. Edwards, "Augmenting image classifiers using data augmentation generative adversarial networks," in *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III*, ser. Lecture Notes in Computer Science, vol. 11141. Springer, 2018, pp. 594–603.
24. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
25. I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, December 6-12, 2020*, 2020.
26. D. Shanmugam, D. W. Blalock, G. Balakrishnan, and J. V. Guttag, "Better aggregation in test-time augmentation," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 1194–1203.
27. E. Park *et al.*, "Big/little deep neural network for ultra low power inference," in *2015 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2015, Amsterdam, Netherlands, October 4-9, 2015*. IEEE, 2015, pp. 124–132.
28. L. Mocerino and A. Calimera, "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.
29. A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 2372–2379.
30. K. Wang, X. Yan, D. Zhang, L. Zhang, and L. Lin, "Towards human-machine cooperation: Self-supervised sample mining for object detection," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 1605–1613.
31. R. G. Rizzo, V. Peluso, and A. Calimera, "Tvfs: Topology voltage frequency scaling for reliable embedded convnets," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 2, pp. 672–676, 2020.
32. C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

33. S. Melacci and M. Gori, "Unsupervised learning by minimal entropy encoding," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 12, pp. 1849–1861, 2012.
34. B. Hassibi and S. Shadbakht, "Normalized entropy vectors, network information theory and convex optimization," in *Proceedings of the IEEE Information Theory Workshop on Information Theory for Wireless Networks, July 1-6, 2007, Solstrand, Norway*. IEEE, 2007, pp. 1–5.
35. Linaro toolchain. [Online]. Available: https://www.linaro.org/downloads/
36. Tensorflow hub. [Online]. Available: https://tfhub.dev
37. Tensorflow lite hosted models. [Online]. Available: https://www.tensorflow.org/lite/guide/hosted_models
38. A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.
39. M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4510–4520.
40. A. Howard *et al.*, "Searching for mobilenetv3," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1314–1324.
41. J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255.
42. V. Peluso, R. G. Rizzo, A. Calimera, E. Macii, and M. Alioto, "Beyond ideal dvfs through ultra-fine grain vdd-hopping," in *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*. Springer, 2016, pp. 152–172.