

Benchmarking the Cost of Adaptation: A System-Level Analysis of Test-Time Adaptation on Edge Devices

Mateusz Piechocki

Poznan University of Technology
Poznań, Poland
mateusz.piechocki@put.poznan.pl

Alessandro Capotondi

University of Modena and Reggio Emilia
Modena, Italy
a.capotondi@unimore.it

Abstract

The widespread use of deep neural networks on edge devices has revealed a significant vulnerability related to performance degradation caused by distribution shifts. Static models fail when the statistical properties of the test data differ from those of the training data due to environmental changes, sensor noise, or domain shift. Test-time adaptation (TTA) is an effective algorithmic solution that allows models to continuously adjust their parameters during inference using unlabeled data streams. However, the existing literature on TTA primarily focuses on improving functional metrics like accuracy and often overlooks the severe resource constraints present in edge computing. This study addresses these gaps by providing a thorough, system-level analysis of TTA that measures hidden costs in terms of memory footprint, computational latency, and energy consumption.

CCS Concepts

• **Information systems** → Computing platforms; • **Hardware** → **Hardware test**; • **Computing methodologies** → *Computer vision*.

Keywords

Test-time Adaptation, Computer Vision, Edge Devices, Benchmarking

ACM Reference Format:

Mateusz Piechocki and Alessandro Capotondi. 2026. Benchmarking the Cost of Adaptation: A System-Level Analysis of Test-Time Adaptation on Edge Devices. In *The 24th Annual International Conference on Mobile Systems, Applications and Services (MobiSys Workshop '26)*, June 21–25, 2026, Cambridge, United Kingdom. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3812836.3814791>

1 Introduction

Conventionally, the deployment of deep neural networks on edge devices has relied on a train-and-freeze paradigm, assuming the inference data matches the training distribution. This assumption often fails in real-world settings due to unpredictable domain shifts. As a result, Test-Time Adaptation (TTA) has emerged as a potential solution that enables models to recalibrate their parameters online using only the incoming data stream (Fig. 1). While cloud environments offer virtually unlimited compute capabilities, applying

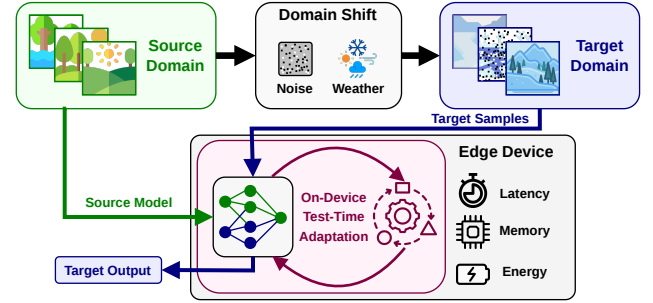


Figure 1: High-level conceptual overview of the on-device test-time adaptation paradigm.

TTA at the edge is challenging because continuous optimization introduces latency and power overheads that can disrupt the device's primary function. Nonetheless, the majority of existing literature omits the operational costs and focuses mainly on functional metrics. To address this issue, this paper presents a system-level benchmark of TTA methods on a heterogeneous hardware testbed to quantify the true cost of algorithmic robustness. The evaluation targets versatile MPU-class platforms and dedicated edge AI computers, but excludes MCU-class devices because the backpropagation, optimizer state tracking, and continuous gradient updates required by most TTA methods are incompatible with kilobyte-scale memory and milliwatt-level power budgets. Within this hardware scope, the contributions are twofold. First, the study provides a detailed on-device profiling of energy consumption, inference latency, and memory overheads. Second, it evaluates methods using both functional metrics and operational costs, utilizing the concept of adaptation efficiency, which quantifies the return on investment of computational updates in resource-constrained settings. To support further research and reproducibility, the source code, including TTA methods implementation, and hardware profiling scripts, is publicly available at <https://github.com/MatPiech/edge-tta>.

The paper is organized as follows. Section 2 reviews related work on model adaptation and prior TTA benchmarks, providing context for positioning this study within on-device test-time adaptation. Section 3 describes the experimental methodology, including datasets, model architectures, TTA methods, hardware testbed, metrics, and evaluation protocol. Section 4 presents and discusses system-level profiling results, emphasizing trade-offs between model performance and operational costs. Finally, Section 5 summarizes key insights and offers hardware-aware recommendations for future edge AI deployments.



This work is licensed under a Creative Commons Attribution 4.0 International License. *MobiSys Workshop '26, Cambridge, United Kingdom*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2712-2/2026/06
<https://doi.org/10.1145/3812836.3814791>

2 Background and Related Work

2.1 Model Adaptation

In recent years, test-time adaptation has become the dominant approach for improving post-deployment robustness. TTA methods update model parameters or recalibrate feature representations during inference using only unlabeled target-domain data, without the source training set. Early work like Pseudo Label [11] updates parameters by treating high-confidence test predictions as surrogate ground-truth labels. In turn, AdaBN [12], an optimization-free method, completely bypasses backpropagation by replacing source-domain batch normalization (BN) statistics learned during source-domain training with local statistics computed directly from the target test stream. Later methods widely adopted entropy minimization techniques to refine predictions without source data. SHOT-IM [13] adapts the feature extractor by minimizing per-sample prediction entropy while enforcing batch diversity, and TENT [18] simplifies this into a popular gradient-based scheme that updates only BN scale and shift parameters. Iwasawa and Matsuo [10] introduced a computationally cheaper T3A method that adjusts the linear classifier prototypes via pseudo-labels and eliminating gradient-based updates, making it suitable for edge devices. In contrast, more recent methods have focused more on addressing more complex deployment scenarios involving continual distribution shifts, small batches, and noisy data. CoTTA [19] mitigates catastrophic forgetting under continuous distribution shifts by employing a teacher–student framework with weight averaging; LAME [4] corrects predictions through Laplacian regularization on classifier outputs without changing parameters. EATA [14] improves gradient-based efficiency by filtering unreliable or redundant samples, and SAR [15] addresses the volatility of edge environments using sharpness-aware minimization to selectively update parameters under noisy or imbalanced streams.

2.2 Prior TTA Benchmarking

The rapid development of test-time adaptation algorithms has prompted several recent benchmarking efforts from complementary perspectives. Wang et al. [20] presented a systematic review and empirical benchmark of TTA methods in online scenarios. To characterize efficiency, the authors introduced novel evaluation metrics and reporting GFLOPs, wall-clock time, and GPU memory usage to clarify the trade-offs between model accuracy and computational overhead. Similarly, Yu et al. [23] benchmarked 13 TTA methods across 5 datasets and multiple regimes (online, offline, batch, instance-level), focusing on accuracy, robustness across different model architectures, and resilience to various distribution shifts. In contrast, Bhardwaj et al. [3] conducted a measurement study of two BN adaptation techniques that specifically target resource-constrained settings, evaluating performance, latency, memory footprint, and energy consumption on FPGA, Raspberry Pi 4, and Nvidia Xavier NX. The authors of [1] analyzed the computational cost of TTA methods, proposing a realistic online evaluation protocol that approximates hardware constraints by emphasizing adaptation speed. In this setting, slower algorithms are explicitly penalized by limiting the number of samples they can process from a constant-rate data stream, showing that fast, simple methods can outperform slower, complex ones under strict real-time constraints.

Finally, Danilowski et al. [5] introduced a benchmark for TTA on edge devices with periodic adaptation, reporting system-level metrics from a real testbed and highlighting challenges posed by limited computational resources, restricted availability of test samples, and diverse distribution shifts.

Overall, TTA methods have become prominent and effective techniques for mitigating concept shift in non-stationary environments. However, although these approaches can be highly effective in controlled experimental scenarios, their deployment on resource-constrained edge platforms faces major practical limits. MPU-class edge hardware, such as low-power systems-on-chip and unified-memory edge GPUs, operates under strict power, thermal, and memory budgets. Therefore, assessing TTA only by its accuracy overlooks these constraints and raises serious concerns about the real-world feasibility of backpropagation-heavy or augmentation-intensive TTA approaches on edge devices.

3 Experimental Methodology

3.1 Datasets and Models

To evaluate robustness and adaptation capabilities, this study uses an adversarially perturbed version of ImageNet [6] - ImageNet-C [9], which consists of 15 types of algorithmic corruption ranging from environmental effects, such as fog and snow, to digital artifacts, such as Gaussian noise and JPEG compression. For all corruptions, severity level 5 was selected to evaluate the models and methods under extreme domain shift. As the highest severity level defined in ImageNet-C, this setting corresponds to applying the maximal degree of visual distortion to the images.

From a model perspective, the benchmark includes the widely used ResNet-18 and ResNet-50 [8] residual networks, alongside the more computationally optimized ResNet-26t variant [22]. Additionally, the Vision Transformer (ViT-Base) [7] is included to assess the performance of transformer-based architectures. The pre-trained weights were obtained from the *PyTorch Image Models (timm)* [21] library, ensuring state-of-the-art baseline performance.

3.2 TTA Methods

Assessing the hardware impact of on-device adaptation requires examining a broad spectrum of learning algorithms, from lightweight statistical updates to compute-heavy gradient-based optimization methods. The first group consists of optimization-free methods that avoid the costly backpropagation step, making them well suited to CPU-limited devices. This group includes AdaBN, T3A, and LAME. AdaBN sets the model to evaluation mode but keeps batch normalization layers in training mode, updating their running statistics across batches using momentum. T3A adapts the model by removing the classification head and refining predictions using an online, entropy-filtered support set of feature prototypes. LAME jointly adapts predictions across batches by constructing an affinity matrix and applying iterative Laplacian optimization to enforce smoothness, all without modifying the underlying network parameters.

The second, and most widely adopted, category relies on entropy minimization, employing gradient-based optimization to iteratively refine model predictions. This group includes methods such as SHOT-IM, TENT, EATA, and SAR. SHOT-IM operates in training mode and enforces the use of current batch statistics to update

Table 1: Overview of state-of-the-art TTA methods selected for evaluation, including brief algorithmic descriptions and the corresponding experimental configurations. BP stands for backpropagation.

Method	BP	Algorithmic Approach	Config
No Adaptation	×	Frozen model at test time	—
AdaBN [12]	×	Update BN running statistics	BN momentum $m = 0.1$
T3A [10]	×	Prototype/feature-based adaptation	Support set size per class $K = 20$
LAME [4]	×	Label assignment via energy minimization	kNN graph affinity with $k = 5$
Pseudo Label [11]	✓	Self-training on pseudo-labels	$LR = 0.001$; confidence threshold $T = 0.9$
SHOT-IM [13]	✓	Information maximization + self-supervision	$LR = 0.001$
TENT [18]	✓	Entropy minimization (update BN affine)	$LR = 0.001$
CoTTA [19]	✓	Continual TTA w/ teacher-student	$LR = 0.001$; EMA rate $\alpha_{C10} = 0.99$; $\alpha_{IN} = 0.999$; restore probability $p_{C10} = 0.1$; $p_{IN} = 0.001$; anchor prob. threshold $ap_{C10} = 0.9$; $ap_{IN} = 0.1$
EATA [14]	✓	Entropy minimization w/ sample filtering	Fisher samples $N = 2000$; Entropy margin $e = 0.4 \ln(N_{\text{classes}})$; Cosine similarity threshold $d = 0.05$
SAR [15]	✓	Sharpness-aware entropy minimization	$LR = 0.001$; SAM $\rho = 0.05$; Entropy margin $e = 0.4 \ln(N_{\text{classes}})$

normalization-layer parameters. Similarly, TENT uses an optimizer to update exclusively the affine parameters of normalization layers, minimizing prediction entropy to promote confident model outputs on test data. EATA performs entropy minimization on selected, reliable, non-redundant samples, while limiting updates to normalization layers in training mode and adding Fisher-information-based regularization to mitigate catastrophic forgetting of source-domain knowledge. SAR, in turn, adopts sharpness-aware minimization and reliability-based filtering to update a subset of normalization-layer parameters in training mode, enforcing batch-specific statistics while preserving high-level features by freezing top layers.

The final category covers methods that use pseudo-labeling and teacher-student architectures to guide adaptation, notably Pseudo Label and CoTTA methods. Pseudo Label performs adaptation by updating both the normalization layers and classifier parameters in training mode, optimizing a cross-entropy loss computed on high-confidence predictions while enforcing batch statistics. CoTTA instead updates all trainable parameters in a teacher-student framework, using augmentation-averaged predictions and periodically restoring a subset of the model weights via stochastic restoration to maintain long-term stability.

A complete overview of these algorithms, including their hyperparameter configurations, is presented in Table 1.

3.3 Hardware Testbeds

Unlike cloud environments, edge devices operate under stringent physical constraints, including limited power, thermal constraints, and restricted computational resources. While the broader edge ecosystem includes ultra-low-power microcontrollers, evaluating TTA on such devices is beyond this benchmark’s scope. Instead, this study focuses on representative, widely available edge AI platforms, specifically the Raspberry Pi 5 [17] and the NVIDIA Jetson Orin

Nano [16]. The Raspberry Pi 5 is a low-cost, compact single-board computer with a Broadcom BCM2712 quad-core Arm Cortex-A76 processor operating at 2.4 GHz and 4 GB RAM. In contrast, the NVIDIA Jetson Orin Nano is a higher-performance edge AI computer with a 6-core Arm Cortex-A78AE processor running at 1.7 GHz, a 1024-core NVIDIA Ampere GPU, and 8 GB of RAM.

3.4 Evaluation Metrics

In this study, on-device evaluation of test-time adaptation methods is performed by measuring the model’s functional metrics under severe distribution shifts and by assessing its operational efficiency on selected hardware. The principal functional metric employed to characterize the model’s performance is the top-1 accuracy. In addition, expected calibration error (ECE) is utilized as a complementary metric to measure the consistency between the predictive confidence of the model and its empirical accuracy. Together, these metrics ensure that continuous adaptation does not produce overconfident, poorly calibrated predictions.

Beyond these indicators, the analysis incorporates an additional operational measure to quantify the effective cost of on-device updates, focusing on adaptation efficiency (η_{AE}). Within the energy-aware systems literature, this trade-off is typically expressed as accuracy gain per unit of energy. It assesses whether the computational overhead introduced by a given TTA method is justified by its functional benefits on resource-constrained devices by normalizing the achieved accuracy improvement by the additional energy consumed over the corresponding update cycles:

$$\eta_{AE} = \frac{\text{Acc}_{TTA} - \text{Acc}_{base}}{E_{TTA} - E_{base}}, \quad (1)$$

where Acc_{base} and E_{base} denote the accuracy and energy consumed under standard model inference without adaptation, and Acc_{TTA} and E_{TTA} denote the corresponding values when performing TTA over the same evaluation stream.

3.5 Evaluation Protocol

Assessing the on-device performance of test-time adaptation requires a multidimensional methodological framework that considers both algorithmic robustness and edge hardware resource constraints. Accordingly, to rigorously evaluate the practical viability of TTA methods for on-device deployment and to quantitatively characterize their associated system-level overheads, the following evaluation protocol was employed.

Adaptation Protocol. The evaluation follows a batch-based on-line adaptation paradigm, wherein each method triggers a single parameter update for every incoming batch. To examine performance across varying throughput requirements, six batch sizes are employed, $B \in \{2, 4, 8, 16, 32, 64\}$. To isolate the effects of individual distribution shifts in ImageNet-C, the model parameters are reinitialized to their original source-trained configuration prior to exposure to each new corruption type. This reset ensures that adaptation-related updates do not accumulate across distinct shifts. **Decoupled Evaluation Strategy.** To perform this large-scale benchmark without compromising the fidelity of the physical hardware behavior, the protocol separates task-level algorithmic evaluation from on-device profiling. This separation is primarily motivated by computational efficiency, since computing functional metrics

Table 2: Summary of system-level results for TTA using a ResNet-50 model, evaluated on selected MPU-class edge devices with the ImageNet-C benchmark. The change in accuracy, ΔAcc , and in ECE, ΔECE , are reported relative to the *No Adaptation* baseline. The metric η_{AE} denotes the adaptation efficiency. Both latency and energy are reported on a per-batch basis.

Method	Raspberry Pi 5 (Batch Size 16)						NVIDIA Jetson Orin Nano (Batch Size 32)					
	ΔAcc (%) $\uparrow$	ΔECE (%) $\uparrow$	Lat. (ms) $\downarrow$	Energy (J) $\downarrow$	Peak RAM (MB) $\downarrow$	η_{AE} $\uparrow$	ΔAcc (%) $\uparrow$	ΔECE (%) $\uparrow$	Lat. (ms) $\downarrow$	Energy (J) $\downarrow$	Peak RAM (MB) $\downarrow$	η_{AE} $\uparrow$
No Adaptation	0.00	0.00	3660.74	25.59	991.92	0.0	0.00	0.00	303.61	5.41	2775.62	0.0
AdaBN	-6.23	74.10	3859.45	26.96	1030.55	-72.76	2.24	72.94	408.66	6.59	2782.83	60.75
T3A	-2.88	-48.03	3958.79	27.68	1367.27	-22.05	-3.18	-47.91	930.05	14.95	3338.02	-10.67
LAME	-0.97	-9.05	3795.25	26.51	1121.89	-16.87	-1.30	-12.58	404.57	6.55	3011.36	-36.49
Pseudo Label	-0.56	75.11	4217.33	29.47	2507.98	-2.31	6.92	69.21	514.80	8.40	5526.63	74.06
SHOT-IM	8.73	-24.05	8654.84	60.53	2646.09	4.00	18.25	-16.49	991.27	15.05	5493.34	60.58
TENT	9.41	-116.65	8616.83	60.12	2985.81	4.36	23.56	-62.86	843.79	13.04	5470.41	98.81
CoTTA			Out-of-Memory						Out-of-Memory			
EATA	15.41	-137.79	4602.03	32.18	2495.84	37.41	33.10	-105.78	1147.74	17.56	5597.54	87.18
SAR	10.50	38.52	6904.60	48.30	2767.84	7.40	22.22	40.83	1780.91	10.28	5528.69	146.00

for all TTA methods and batch sizes on the full dataset is highly compute-intensive. Thus, classification accuracy and expected calibration error were computed on the full dataset on a cloud server with an NVIDIA GeForce RTX 4090 GPU (24 GB VRAM) running CUDA 12.8. To ensure algorithmic consistency across these distinct hardware environments, the entire evaluation pipeline used PyTorch v2.8.0 [2]. Operating within this unified software stack, cross-platform consistency was empirically validated to justify the use of cloud-computed metrics as a proxy for edge performance. Preliminary tests with Gaussian noise corruption on MPU-class edge devices demonstrated an accuracy deviation of approximately 0.2% relative to the cloud GPU results. Given this negligible discrepancy, the cloud infrastructure was then used to exhaustively evaluate all TTA methods and six batch sizes. Nonetheless, this split approach has limitations. First, minor cross-platform floating-point differences may accumulate over extended computation, potentially causing slight deviations in model behavior. Second, cloud-based evaluation may fail to capture performance degradations that arise only under extreme, edge-specific conditions, such as sustained thermal throttling during long-term deployment scenarios.

On-Device Profiling. While functional performance metrics were offloaded to the cloud, all operational metrics were measured directly on the edge AI devices under the aforementioned Gaussian noise to capture hardware constraints and quantitatively characterize the system-level overheads. To evaluate real-time system responsiveness, adaptation latency (ms) was measured as the per-batch in-loop wall-clock time, isolating algorithmic computation time by excluding disk I/O and preprocessing. To quantify the energy cost associated with each update cycle, energy consumption (J) was sampled at 5 Hz with tail integration to mitigate underestimation; on the Jetson platform this was obtained from on-board power sensors, while on the Raspberry Pi 5 it was inferred from CPU-utilization-based estimates. Finally, to verify that the adaptation process remains within hardware limits, memory allocation (MB), including process RSS and peak RAM, was sampled at 10 Hz using *psutil*. For each metric, mean, standard deviation, and min/max were computed before aggregating across corruption levels. Consequently, this on-device profiling explicitly characterizes hardware-specific operational limits, including out-of-memory failures, beyond which target accuracy trajectories cannot be achieved.

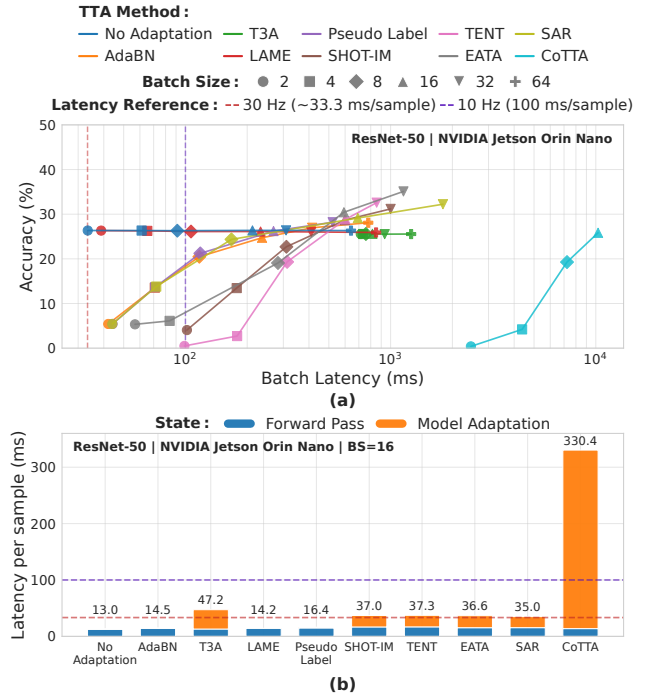


Figure 2: Performance and latency analysis of ResNet-50 across different TTA methods measured on Jetson Orin Nano. (a) Pareto frontier of accuracy vs. batch latency, illustrating the trade-offs between functional performance and computational delay. (b) Per-sample execution time breakdown for a fixed batch size of 16. The overall latency is decomposed into base forward time and method-specific adaptation overhead.

4 Results

This section reports the system-level overheads for nine TTA algorithms on representative edge AI platforms. These methods were systematically evaluated along three dimensions – computational latency, memory footprint, and energy consumption – and their resource demands were analyzed relative to functional accuracy.

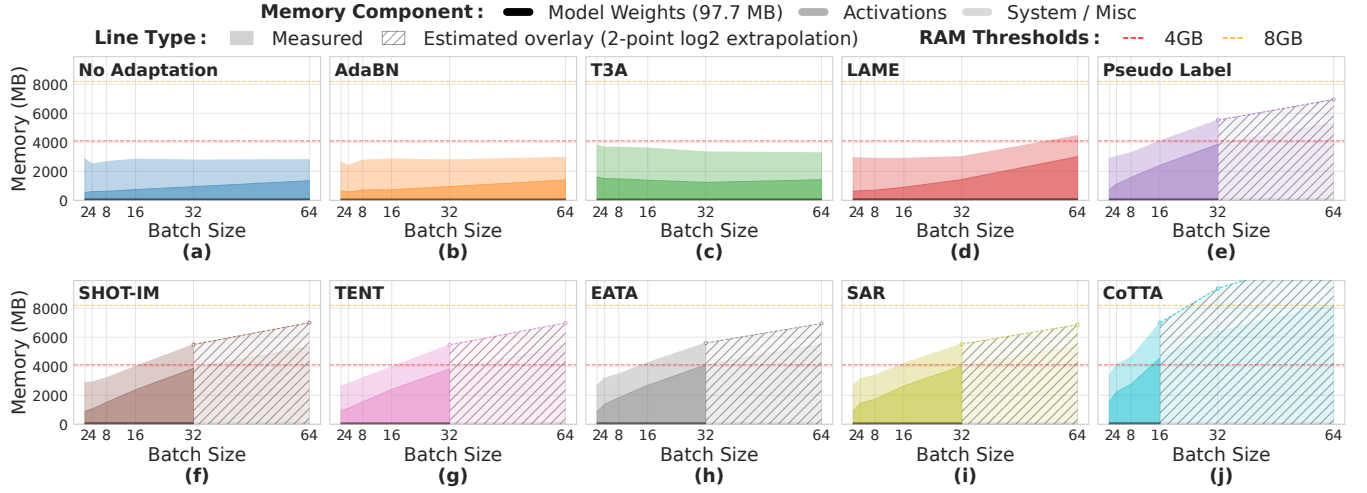


Figure 3: Mean memory usage and scaling characteristics of ResNet-50 under different TTA methods, evaluated on an NVIDIA Jetson Orin Nano. The stacked area plots decompose total VRAM consumption into static model parameters (97.7 MB), batch-dependent intermediate activations, and system/framework overhead. Horizontal reference lines indicate strict hardware memory limits representative for MPU-class edge platforms. The hatched regions indicate the estimated memory utilization, obtained via a two-point $\log_2$ extrapolation, in scenarios where direct empirical measurements were infeasible due to out-of-memory (OOM) failures on the target hardware platform.

4.1 Latency and Real-Time Feasibility

A primary challenge for edge deployment is the temporal overhead introduced by test-time adaptation mechanisms. Figure 2b presents the per-sample execution time breakdown for ResNet-50 (batch size 16) on the Jetson Orin Nano. The total processing latency is decomposed into the base inference time (forward pass) and the algorithm-specific optimization overhead associated with model adaptation. The results indicate a clear separation between optimization-free and gradient-based approaches. Lightweight, statistics-driven methods such as AdaBN and LAME introduce negligible adaptation overhead, adding less than 2 ms of latency over the baseline. Consequently, these methods easily satisfy strict real-time processing requirements when hardware acceleration is available. In contrast, optimization-based techniques that rely on backpropagation and iterative optimizer updates (e.g., TENT, EATA, SAR) considerably exceed the 30 Hz threshold because of the substantial computational cost of backward passes and parameter updates, frequently pushing end-to-end latency beyond typical real-time budgets. In particular, complex, multi-stage approaches such as CoTTA introduce massive adaptation bottlenecks, making them essentially unusable for low-latency, real-time edge scenarios unless adaptation is performed conditionally (e.g., via trigger-based or periodic updates) or moved off the critical inference path.

Finally, Fig. 2 illustrates the accuracy-latency trade-off as a function of batch size. For gradient-based methods, as well as for AdaBN, very small batch sizes (e.g., 8 or less) tend to degrade the model’s functional performance. In contrast, batch sizes of 16 and 32 lead to improved adaptability. LAME, by comparison, maintains approximately consistent functional performance across all evaluated batch sizes.

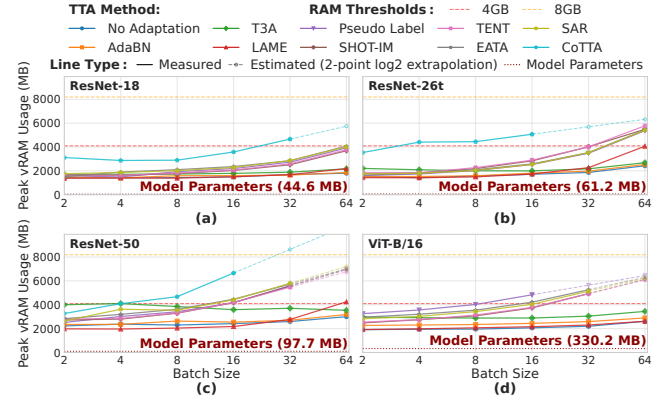


Figure 4: Peak VRAM usage scaling with batch size across four vision architectures under various TTA methods.

4.2 Memory Wall

Although server-class GPUs typically offer plentiful VRAM, edge devices must operate under tight memory budgets. Figure 3 illustrates how the mean memory footprint of ResNet-50 scales across TTA methods. Breaking down total memory usage into static model parameters, batch-dependent intermediate activations, and framework-induced overhead reveals that storage of intermediate activations required for backpropagation is the dominant memory bottleneck during continuous adaptation. This memory component scales approximately linearly with batch size and increases with both network depth and feature-map spatial resolution. Consequently, memory capacity becomes the primary limiting factor

on edge devices equipped with 4 or 8 GB of RAM, often being exhausted before the available computational resources are fully utilized. Methods like CoTTA, which maintain additional temporal ensembles and teacher-student architectures, exhibit an inherently excessive static memory footprint. As batch sizes increase, these algorithms rapidly clash with the hardware memory walls, leading to out-of-memory (OOM) failures. For high-batch configurations where empirical measurements were unattainable due to OOM errors, a 2-point $\log_2$ extrapolation was applied to estimate memory consumption. Figure 4 illustrates how the peak memory usage scales for four vision architectures (ResNet-18, ResNet-26t, ResNet-50, and ViT-B/16) under different TTA methods and batch sizes.

4.3 Energy Efficiency

Energy consumption is the fundamental limiting factor for battery-powered edge devices. The benchmarks highlight a significant difference in energy efficiency depending on both the underlying hardware architecture and the employed TTA method. For example, processing a batch of 16 images with ResNet-50 under high CPU utilization on the Raspberry Pi 5 requires approximately 1.6 J per sample. By contrast, using the Ampere GPU on the Jetson Orin Nano to perform the same workload yields a markedly lower energy cost of approximately 169 mJ per sample, corresponding to a 9.5 $\times$ improvement in energy efficiency, despite the higher peak power draw of the GPU-based system. This finding demonstrates that, for computation-intensive vision workloads, specialized hardware accelerators can achieve lower per-sample energy consumption than general-purpose CPUs, as they execute these tasks with reduced latency and substantially higher throughput.

In general, gradient-based adaptation methods yield measurable accuracy improvements, but substantially increase the energy consumption per batch. This occurs because the backward pass, together with any associated multistep update schedule, imposes additional computational workload and extend the duration of device active operation. As reported in Table 2, these methods require approximately 1.5 to 3 $\times$ more energy per batch relative to the baseline configuration. In contrast, optimization-free approaches (e.g., AdaBN or LAME) maintain near-baseline energy consumption.

5 Conclusions

The large-scale deployment of deep learning models on edge devices is often hindered by performance degradation under distribution shifts. While test-time adaptation methods provide a compelling algorithmic solution, this system-level analysis demonstrates that the existing literature substantially underestimates the hidden costs associated with adaptation. By benchmarking nine distinct TTA algorithms on edge AI platforms (Raspberry Pi 5 and NVIDIA Jetson Orin Nano), this study exposes severe scalability constraints in memory footprint, latency, and energy consumption. Specifically, a stringent memory wall arises from the linear growth of dynamic activations during backward passes, frequently resulting in out-of-memory failures on edge devices with 4 GB (batch size 32) and 8 GB (batch size 64) of RAM. Furthermore, execution-time measurements indicate that gradient-based adaptation methods are generally unable to satisfy typical real-time throughput requirements (30 Hz),

limiting their practical usage to offline scenarios or applications that can tolerate substantial processing delays.

Although these results clearly delineate the system-level constraints of current TTA methods, inherent limitations within the adopted decoupled cloud-edge evaluation framework must be acknowledged. By computing functional performance metrics in the cloud to facilitate large-scale benchmarking, the study may overlook subtle yet cumulatively significant discrepancies in floating-point arithmetic across heterogeneous hardware architectures. Additionally, this methodology does not account for potential algorithmic performance degradations that may occur under extreme, edge-specific operating conditions, such as dynamic thermal throttling or memory swapping, during long-term, real-world deployment. Finally, while the benchmark quantifies the operational costs of TTA on edge AI platforms, the explicit exclusion of MCU-class devices underscores a critical need for future investigations into ultra-lightweight adaptation paradigms.

While the scope of this study is intentionally centered on MPU-class edge platforms and vision workloads, the observed system-level trends are expected to generalize across the broader edge continuum. The memory wall associated with activation storage, the latency overhead of gradient-based parameter updates, and the energy costs of continual optimization represent fundamental architectural limitations that are likely to arise on other edge device types (e.g., mobile SoCs and small NPUs) and in non-vision modalities relying on similar network backbones. The precise magnitude of these overheads will naturally depend on context-specific factors, including model architecture (e.g., transformers vs. CNNs), numerical precision, accelerator support, the deployment envelope (e.g., battery-powered vs. grid-powered, thermally constrained vs. actively cooled), and operational conditions (e.g., intermittent connectivity, or long-duration executions under resource contention).

Collectively, this benchmark advocates for a paradigm shift in TTA research, in which algorithms must be designed to maximize not only functional metrics, such as accuracy, but also explicit hardware awareness and resource efficiency. At the algorithmic level, future work should prioritize computationally and memory-efficient adaptation mechanisms, including activation checkpointing, optimizer state compression, and improved mixed-precision stability, to facilitate robust, continual adaptation in resource-constrained edge environments. At the system level, TTA should be treated as a conditional capability rather than a default configuration, enabled only when expected robustness gains clearly exceed system overhead. Adaptation cycles must be dynamically triggered by distribution shift detectors rather than running continuously, and bounded by explicit, hardware-enforced limits on update frequency, memory allocation, and energy consumption.

Acknowledgments

This research was supported by the European Union's Horizon Europe research and innovation programme under the project dAIEDGE (Grant Agreement No. 101120726), through the dAIEDGE Exchange Programmes Open Call 1, under Sub-grant Agreement No. dAIIOC01. The APC was funded by Poznan University of Technology, internal grant 0214/SBAD/0257.

References

- [1] Motasem Alfarra, Hani Itani, Alejandro Pardo, Shyma Alhuwaider, Merey Ramazanov, Juan C. Pérez, Zhipeng Cai, Matthias Müller, and Bernard Ghanem. 2024. Evaluation of test-time adaptation under computational time constraints. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) (ICML '24). JMLR.org, Article 41, 16 pages.
- [2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhersch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (La Jolla, CA, USA) (ASPLOS '24). Association for Computing Machinery, New York, NY, USA, 929–947. doi:10.1145/3620665.3640366
- [3] Kshitij Bhardwaj, James Diffenderfer, Bhavya Kaikhura, and Maya Gokhale. 2022. Benchmarking Test-Time Unsupervised Deep Neural Network Adaptation on Edge Devices. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 236–238. doi:10.1109/ISPASS55109.2022.00033
- [4] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. 2022. Parameter-free Online Test-time Adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8334–8343. doi:10.1109/CVPR52688.2022.00816
- [5] Michal Danilowski, Soumyajit Chatterjee, and Abhirup Ghosh. 2025. BoTTA: Benchmarking on-device Test Time Adaptation. arXiv:2504.10149 [cs.LG] doi:10.48550/arXiv.2504.10149
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. doi:10.1109/CVPR.2016.90
- [9] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *Proceedings of the International Conference on Learning Representations* (2019).
- [10] Yusuke Iwasawa and Yutaka Matsuo. 2021. Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 2427–2440. https://proceedings.neurips.cc/paper_files/paper/2021/file/1415fe9fa0fa1e45dddcff5682239a0-Paper.pdf
- [11] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. 896.
- [12] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. 2016. Re-visiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779* (2016).
- [13] Jian Liang, Dapeng Hu, and Jiashi Feng. 2020. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*. PMLR, 6028–6039.
- [14] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofu Chen, Shijian Zheng, Peilin Zhao, and Minghui Tan. 2022. Efficient Test-Time Model Adaptation without Forgetting. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 16888–16905. <https://proceedings.mlr.press/v162/niu22a.html>
- [15] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiqian Wen, Yaofu Chen, Peilin Zhao, and Minghui Tan. 2023. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400* (2023).
- [16] NVIDIA Corporation. 2024. *NVIDIA Jetson Orin Nano Super Developer Kit Datasheet*. NVIDIA Corporation. <https://nvdam.widen.net/s/zkfqjmtsd2/jetson-orin-datasheet-nano-developer-kit-3575392-r2> Rev. 2. Accessed: 2026-03-02.
- [17] Raspberry Pi Ltd. 2024. *Raspberry Pi 5*. Raspberry Pi Ltd. <https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf> Rev. 1. Accessed: 2026-03-02.
- [18] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2021. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=uXl3bZLkr3c>
- [19] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. 2022. Continual Test-Time Domain Adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7191–7201. doi:10.1109/CVPR52688.2022.00706
- [20] Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. 2025. In Search of Lost Online Test-Time Adaptation: A Survey. *International Journal of Computer Vision* 133, 3 (01 Mar 2025), 1106–1139. doi:10.1007/s11263-024-02213-5
- [21] Ross Wightman. 2019. PyTorch Image Models. <https://github.com/rwightman/pytorch-image-models>. doi:10.5281/zenodo.4414861
- [22] Ross Wightman, Hugo Touvron, and Hervé Jégou. 2021. ResNet strikes back: An improved training procedure in timm. arXiv:2110.00476 [cs.CV] <https://arxiv.org/abs/2110.00476>
- [23] Yongcan Yu, Lijun Sheng, Ran He, and Jian Liang. 2023. Benchmarking test-time adaptation against distribution shifts in image classification. *arXiv preprint arXiv:2307.03133* (2023).