EFFICIENT TEST-TIME ADAPTATION VIA DECOUPLED BN UPDATE FOR EDGE DEVICES

Xiao Ma
Singapore Management University
Singapore
xiaoma.2022@phdcs.smu.edu.sg

Young D. Kwon
Samsung AI Center–Cambridge
Cambridge, United Kingdom
yd.kwon@samsung.com

Dong Ma*
University of Cambridge
Cambridge, United Kingdom
dm878@cam.ac.uk

ABSTRACT

Test-Time Adaptation (TTA) updates a deployed model during online inference to mitigate domain shift. Most TTA methods adapt to the target domain by updating Batch Normalization (BN) layers with unlabeled test data, but often at high memory cost. We observe that BN adaptation consists of two distinct updates, BN statistics and BN affine parameters, which differ in memory footprint and batch size sensitivity. Based on this insight, we propose Decoupled BN Update (DBU), which updates BN statistics with larger batches to capture reliable domain information, while updating affine parameters with small batches to reduce memory overhead. Experiments show that DBU achieves a better performance–memory trade-off, delivering up to 14.5% accuracy improvement in the small-batch setting.

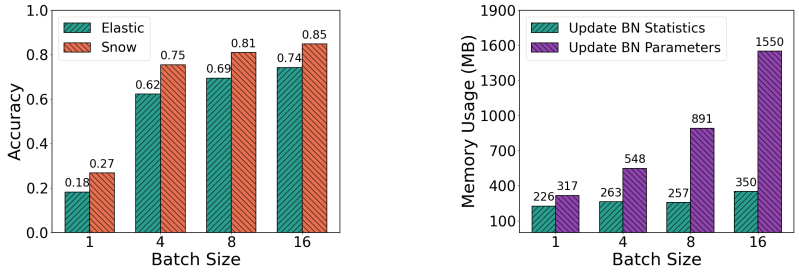
1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable success across a wide range of computer vision tasks (Pouyanfar et al., 2018). However, their performance often degrades significantly under distribution shifts between the training data and unseen test data - a challenge that frequently arises in real-world and real-time applications (Koh et al., 2021; Sun et al., 2022). To address this limitation, DNNs must be able to adapt effectively to such shifts.

Test-time adaptation (TTA) (Liang et al., 2025) enables a pretrained model to adapt online using unlabeled test batches during inference. Representative methods Wang et al. (2020); Niu et al. (2022; 2023); Hong et al. (2023) typically minimize prediction entropy on target data and operate batch by batch, alternating between an *adaptation* step (updating parameters via backpropagation) and an *inference* step with the updated model. To reduce memory overhead, many approaches—especially for CNNs—update only Batch Normalization (BN) layers, since BN is known to capture domain-specific information (Benz et al., 2021). However, BN-based adaptation still relies on sufficiently large batches to estimate reliable statistics, which can be prohibitively memory-intensive: for example, Tent often performs best at batch size 64, exceeding 5.5 GB peak memory and surpassing the budget of resource-constrained edge devices (e.g., Jetson Orin Nano with 4 GB available memory). Moreover, shrinking the batch size can substantially compromise adaptation accuracy.

To address the memory overhead induced by large batches, our preliminary study finds that BN adaptation comprises two components with different batch-size sensitivity. Updating BN statistics benefits from large batches but requires only forward passes, whereas updating BN affine parameters can be done with small batches but requires backpropagation and is memory-intensive. Motivated by this, we propose Decoupled BN Update (DBU), which updates BN statistics using large-batch forward passes and updates affine parameters using small-batch backpropagation, achieving a better accuracy–memory trade-off. To further improve efficiency, DBU adapts only a small subset of samples per target domain and performs standard inference on the remaining data, yet attains accuracy close to full adaptation.

*Corresponding author



(a) (b)

Figure 1: (a) TTA performance under different batch sizes, (b) memory usage of updating BN statistics and parameters.

We evaluate our method on CIFAR-10-C and ImageNet-C, two standard TTA benchmarks, and compare it against representative baseline methods. Experiments show that DBU achieves a better performance–memory trade-off, delivering up to a 14.5% accuracy improvement in the small-batch setting.

2 Related Work

Test-time adaptation (TTA) has emerged as a practical solution for mitigating domain shifts that can severely degrade model reliability in deployment (Wang et al., 2024; Liang et al., 2025; Xiao & Snoek, 2024). Most TTA studies primarily focused on updating the model's normalization layers. For example, simply recalibrating batch normalization (BN) statistics was found to recover some of the accuracy lost under distribution shifts (Benz et al., 2021). Building on this idea, entropy-based optimization techniques such as TENT (Wang et al., 2020) and EATA (Niu et al., 2022) update gradients online under the guidance of prediction entropy, which is often combined with sample filtering or dynamic reweighting to improve stability. Recent TTA research has increasingly focused on improving efficiency from various angles. Memory-aware gradient-based methods aim to reduce the footprint of backpropagation. For example, MECTA (Hong et al., 2023) prunes gradient paths and normalizes only selected layers to lower activation storage, while EcoTTA (Song et al., 2023) leverages compact meta-networks to minimize backpropagation overhead. L-TTA (Shin & Kim, 2024) enhances efficiency by restricting adaptation to shallow stem layers in CNNs.

3 Method Design

3.1 Preliminary

Mainstream lightweight DNN architectures such as ResNet50 typically leverage a Batch Normalization (BN) layer after every convolutional layer to normalize feature values and stabilize the training process. The normalization operation allows BN layers to capture the distribution of the batch, indicating that BN layers play a critical role in TTA. Consequently, *most existing TTA approaches only adapt the BN layers rather than the entire model*, as it is sufficient to align the model with distribution shifts. However, these approaches rely on a large batch size (e.g., 64 in Tent (Wang et al., 2020) and EATA (Niu et al., 2022)) to capture the new domain's distribution. Figure 1(a) validates this by exemplifying the adaptation performance of EATA on the Elastic and Snow domain under different batch sizes, where smaller batch sizes yield poorer performance. On the other hand, larger batch sizes also demand significantly more memory to store intermediate activations and gradients for back-propagation, posing a challenge for memory-constrained edge devices. For example, as shown in Figure 1(b), with a batch size of 4, the memory requirement for updating BN affine parameters of ResNet50 can easily exceed the available memory space of some small edge devices such as Raspberry Pi Zero 2W with only 512MB DRAM.

To address this challenge, we delved into the detailed operation of BN layers during adaptation. Specifically, the adaptation process begins with updating **BN statistics** (mean and variance) and updating **BN affine parameters** simultaneously. However, our detailed analysis reveals that *BN statistics and BN affine parameters exhibit different sensitivities to batch size and impose distinct memory overheads*. BN statistics are obtained by normalizing the input batch, which intuitively favors large batch sizes, whereas BN affine parameters are less sensitive to batch sizes. We further

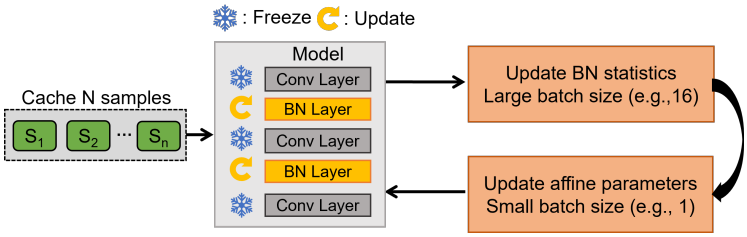


Figure 2: The overview of Decoupled BN Update (DBU).

analyzed the memory usage of updating BN statistics and BN affine parameters under different batch sizes in ResNet50. As shown in Figure 1(b), increasing the batch size results in significantly higher memory consumption when updating BN affine parameters, as this process requires backpropagation. In contrast, the memory usage for updating BN statistics remains relatively stable, since it only involves the forward pass. **This finding reveals that the memory overhead associated with a large batch size primarily stems from updating BN affine parameters.** Conversely, updating BN statistics, which is crucial for capturing data distributions, can be performed using large batch sizes without incurring substantial memory overhead.

3.2 DBU: Decoupled BN Update

Inspired by the findings above Section 3.1, we propose a decoupled BN update strategy, as shown in Figure 2, where the BN statistics are first updated with larger batch sizes during the inference pass, followed by the adjustment of the BN affine parameters with smaller batch sizes during the back-propagation pass. Moreover, to further improve efficiency on edge devices, we adapt the model using only a subset of the incoming data, rather than adapting to all samples.

3.2.1 BN statistics update

First, for each domain, we cache only the first (N) samples for adaptation, rather than all samples. Specifically, the samples are also split into batches of statistics updating size (e.g., 16). Then we use the cached batches to update the BN statistics of each layer. In detail, we employ an Exponential Moving Average (EMA) approach to integrate the BN statistics of the current batch with historical BN statistics, defined as follows:

$$S_k = (1 - m) \cdot S_{k-1} + m \cdot B_k \tag{1}$$

where S_k represents the integrated BN statistics at batch k, m is the momentum factor, S_{k-1} is the BN statistics integrated from the previous batch, and B_k represents the BN statistics of the current batch. Particularly, S_0 is the BN statistics of the source model.

Concretely, for each BN layer, we apply Eq. equation 1 to both the running mean and running variance, and update them iteratively over the cached batches. This design enables stable estimation of domain-specific BN statistics even when the cached set is limited, since the EMA smooths out the noise from individual batches. Importantly, this BN-statistics update requires only forward passes and does not involve backpropagation, making it memory-efficient for on-device adaptation.

The proposed EMA-based updating scheme allows us to seamlessly integrate the similar source model with the new domain samples, ensuring an accurate and robust alignment of both source and new domain distributions.

3.2.2 BN affine parameters update

After updating the BN statistics, the model already captures the distribution of the new data, but the BN affine parameters still need to be fine-tuned accordingly through back-propagation. To fit into the limited on-device memory (e.g., 512 MB as plotted in Figure 1(b)), we aim to update the BN affine parameters with a small batch size (e.g., 1). However, back-propagation using a single sample is challenging due to the inherent instability of unsupervised learning (Niu et al., 2023). To achieve stable fine-tuning, we introduce a sample filter to remove unreliable samples by following

Table 1: Comparison of accuracy (%) on CIFAR-10-C and ImageNet-C using ResNet-50 along with memory consumption on Jetson Orin Nano.

Method	Batch Size = 1			Batch Size = 16			Batch Size = 64		
	Avg. Acc(%)		Memory (MB)	Avg. Acc(%)		Memory (MB)	Avg. Acc.(%)		Memory (MB)
	Cifar10	ImageNet		Cifar10	ImageNet		Cifar10	ImageNet	
Source	59.5	26.9	242	59.5	26.9	358	59.5	26.9	809
EATA	22.8	0.8	506	78.3	31.9	1728	81.0	38.4	5783
L-TTA	10.1	0.1	449	76.8	26.6	1250	81.2	34.5	3580
MECTA	65.0	10.0	**378**	81.3	33.2	**1231**	81.3	33.7	**3836**
Ours	**79.5**	**36.7**	414	**84.3**	**39.4**	1677	**86.0**	**40.4**	5763

EATA (Niu et al., 2022) to refine the entropy minimization. The key intuition is that high-entropy predictions are unreliable (i.e., high uncertainty) and can introduce noisy gradients that make the unsupervised adaptation unstable and even degrade performance. Specifically, following EATA (Niu et al., 2022) and SAR (Niu et al., 2023), we first filter out high-entropy samples when calculating the entropy loss, as the model is less confident in predicting them.

Therefore, the overall loss function is defined as:

$$\mathcal{L}_{\text{ent}} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Big(- \sum_{c=1}^{C} p_{i,c} \log p_{i,c} \Big), \qquad \mathcal{S} = \{\, i \mid H(p_i) < \tau \,\}, \tag{2}$$

In Equation (2), $(-\sum_{c=1}^{C} p_{i,c} \log p_{i,c})$ is the prediction entropy. We compute the entropy-minimization loss only over the reliable subset $\mathcal{S}$, defined as samples whose entropy falls below a threshold τ. Following SAR (Niu et al., 2023), we set $\tau = 0.4 \log(C)$, where C is number of task classes.

4 Evaluations

4.1 Datasets and Baselines

Datasets and Models. Following recent works in TTA (Shin & Kim, 2024; Niu et al., 2024), we conduct a comprehensive evaluation across CIFAR10-C and ImageNet-C, each of which introduces 15 common corruption types applied to the original test sets. We adopt the most severe corruption level (severity = 5). For backbone models, we adopt ResNet-50 (He et al., 2016). We compare our proposed DBU with several efficient TTA approaches. For efficient CNN-based TTA, we include EATA (Niu et al., 2022), MECTA (Hong et al., 2023), and L-TTA (Shin & Kim, 2024). For the adaptation, we use 256/2048 samples for each domain in CIFAR10-C/ImageNet-C and run regular inference for other samples in the domain. The learning rates are set to 1×10^{-4} for CIFAR-10-C and 1×10^{-5} for ImageNet-C.

4.2 Experimental Results

On CIFAR-10-C and ImageNet-C with ResNet-50, DBU consistently achieves the best accuracy across all batch sizes (Table 1). The advantage is most pronounced in the small-batch setup, where many BN-based baselines struggle to obtain reliable statistics: at batch size 1, EATA and L-TTA collapse on CIFAR-10-C/ImageNet-C, while DBU remains strong at 79.5% and 36.7%, substantially outperforming the most competitive baseline, MECTA, which performs 65.0% and 10.0%. Notably, DBU improves over MECTA by 14.5% on CIFAR-10-C at batch size 1, demonstrating that decoupling BN updates is particularly beneficial when the batch size is small.

In terms of memory consumption on Jetson Orin Nano, DBU provides a favorable performance–memory trade-off. At batch size 1, DBU requires only 414 MB, comparable to other lightweight methods (e.g., MECTA at 378 MB) while delivering far higher accuracy. At batch size 16, DBU achieves the best accuracy with 1.68 GB memory, still within a practical on-device budget. As batch size increases to 64, memory grows for all BN-based methods; DBU reaches 5.76 GB, similar to EATA, but attains the highest accuracy. These results validate our key design: large-batch forward-only BN-statistics updates and small-batch affine parameters updates yield robust

adaptation while keeping the memory footprint manageable, especially in the resource-constrained small-batch setting. More evaluatins can be found in Section A.

5 Conclusion

This work begins by revisiting the adaptation mechanism of Batch Normalization (BN) layers and observing that updating BN statistics and affine parameters exhibits markedly different sensitivity to batch size. Motivated by this insight, we propose DBU, a BN-based adaptation method that decouples BN statistics and affine-parameter updates and performs them with different batch sizes. Experiments show that DBU achieves strong adaptation performance while substantially reducing memory consumption, without sacrificing accuracy.

References

Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting batch normalization for improving corruption robustness. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 494–503, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Junyuan Hong, Lingjuan Lyu, Jiayu Zhou, and Michael Spranger. Mecta: Memory-economic continual test-time model adaptation. In *2023 International Conference on Learning Representations*, 2023.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.

Shuaicheng Niu, Chunyan Miao, Guohao Chen, Pengcheng Wu, and Peilin Zhao. Test-time model adaptation with only forward passes. *arXiv preprint arXiv:2404.01650*, 2024.

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM computing surveys (CSUR)*, 51(5):1–36, 2018.

Jin Shin and Hyun Kim. L-tta: Lightweight test-time adaptation using a versatile stem layer. *Advances in Neural Information Processing Systems*, 37:39325–39349, 2024.

Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11920–11929, 2023.

Tao Sun, Mattia Segu, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari, and Fisher Yu. Shift: a synthetic driving dataset for continuous multi-task domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21371–21382, 2022.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey. *International Journal of Computer Vision*, pp. 1–34, 2024.

Zehao Xiao and Cees GM Snoek. Beyond model adaptation at test time: A survey. *arXiv preprint arXiv:2411.03687*, 2024.

Table 2: Adaptation accuracy (%) of using a few samples from the domain (Few-data adaptation). We compared the results with continual Tent which uses entropy minimization on test batch statistics. To illustrate the key role of adapting the BN statistics, we also reported the results of only adapting the BN statistics using EMA BN without updating the affine parameters, as Few-data BN Stats. and adapting only through test batch statistics without updating affine parameters, as Continual BN Stats.

Method	gauss.	shot	impul.	defoc.	glass	motion	zoom	snow	frost	fog	brigh.	contr.	elast.	pixel.	jpeg	Avg. Acc
Continual BN Stats.	68.2	71.3	60.3	83.5	64.6	81.4	85.2	84.6	83.5	86.8	90.7	84.7	74.5	78.8	70.7	77.9
Continual Tent	68.8	72.8	62.5	83.2	65.3	81.3	84.8	83.6	83.3	85.2	89.6	84.8	75.8	79.2	72.3	**78.2**
Few-data BN Stats.	67.9	73.7	62.0	72.1	65.8	79.4	88.6	88.4	85.4	88.5	92.9	80.8	76.9	70.3	71.7	77.6
Few-data Adapt.	67.9	73.7	62.3	72.2	66.1	79.4	88.6	88.5	85.4	88.6	92.9	81.4	76.9	71.0	72.1	77.8

Table 3: Adaptation accuracy (%) of decoupled BN adaptation (Decouple Adapt.) compared to conventional synchronized adaptation using 128 samples. We compared the decoupled adaptation with three baselines: (1) BN Stats: adapting only the BN statistics; (2) BN Adapt: jointly updating BN statistics and affine parameters through entropy minimization; (3) Adapt. with Filter: further adapting the model using only low-entropy data.

Method	gauss.	shot	impul.	defoc.	glass	motion	zoom	snow	frost	fog	brigh.	contr.	elast.	pixel.	jpeg	Avg. Acc
BN Stats.	67.9	73.7	62.0	72.1	65.8	79.4	88.6	88.4	85.4	88.5	92.9	80.8	76.9	70.3	71.7	77.6
BN Adapt.	67.9	73.7	62.3	72.2	66.1	79.4	88.6	88.5	85.4	88.6	92.9	81.4	76.9	71.0	72.1	77.8
BN Adapt. with Filter	68.9	74.6	61.7	70.6	64.8	79.8	88.5	87.8	87.0	87.6	92.5	82.5	77.3	71.4	72.2	77.8
Decoupled Adapt.	70.2	75.1	62.7	74.7	66.5	81.6	89.7	89.6	87.1	89.9	94.1	83.8	78.5	76.0	73.3	**79.5**

A Decoupled BN Adaptation

A.1 Adaptation from only a few data can mitigate domain shift

Inspired by previous work (Benz et al., 2021), corruption robustness can be improved by capturing the BN statistics of only 32 samples. We hypothesize that adapting to a domain using a small number of samples is sufficient to achieve a robust model for the remaining samples in that domain. By employing the EMA strategy to update BN statistics and entropy minimization to update the affine parameters on 128 samples in CIFAR-10-C, the inference accuracy on the remaining samples is illustrated in Table 2. We compared the few-data adaptation with Tent, the continual adaptation method based on test batch statistics and entropy minimization. The momentum of few-data adaptation is set to 0.15.

In Table 2, Continual BN Stats. continuously updates the Batch Normalization (BN) statistics using test batch statistics. Continual Tent adapts the BN affine parameters through entropy minimization while also updating the BN statistics using test batch statistics. Few-data BN Stats. leverages 128 samples to update the BN statistics via an Exponential Moving Average (EMA) approach. Few-data Adapt. further extends this by utilizing the 128 samples to update both the BN statistics with EMA BN and the affine parameters through entropy minimization.

We can see that the few-data adaptation achieves comparable results to continual Tent across most corruption types, with a minimal difference in the average accuracy (77.8% vs. 78.2%). This demonstrates that adapting to the domain using a small subset of samples is sufficient to maintain robust performance on the remaining data. Moreover, this approach enables adaptive triggering of adaptation using only a few samples, making it both practical and efficient.

A.2 Adapting BN statistics vs. Affine parameters

The results of Continual BN Stats. and Continual Tent in Table 2 illustrated that updating the BatchNorm (BN) statistics is crucial for improving adaptation performance, as demonstrated by Fig-

ure 1(a). The sensitivity of BN statistics to batch size is evident from the performance degradation with smaller batch sizes. This indicates that a sufficient number of samples per batch is necessary to capture reliable BN statistics for effective adaptation. However, as shown in Figure 1(b), updating BN statistics with larger batch sizes is feasible from a memory perspective. Even with a batch size of 32, the memory usage for updating BN statistics is significantly lower than updating the affine parameters with a batch size as small as 4. Moreover, with a batch size of 4, the memory requirement for updating BN parameters of ResNet50 can easily exceed the available memory space of some edge devices, such as the Raspberry Pi Zero 2W with only 512 MB DRAM.

This comparison highlights the efficiency of adapting BN statistics alone. It not only delivers robust performance when a sufficient batch size is used but also enables memory-efficient adaptation. This balance between performance and resource usage makes updating BN statistics a practical and effective approach for domain adaptation, especially when resources are constrained.