LeanTTA: A Backpropagation-Free, Stateless Approach to On-Device Test-Time Adaptation

Cynthia Dong[1] **Young D. Kwon**[2,4] **Hong Jia**[3] **George Rizos**[2] **Cecilia Mascolo**[2]
[1]University of Washington [2]University of Cambridge [3]University of Auckland
[4]Samsung AI Center-Cambridge, UK
cyndo@cs.washington.edu yd.kwon@samsung.com

Abstract

For small, compressed models constrained by edge hardware and limited data storage or availability, test-time adaptation (TTA) can restore robustness against common corruptions that induce covariate shift. However, many TTA methods rely on backpropagation and large batch sizes, which we show fail under constrained data and hardware. In this work, we present **LeanTTA**, a combination of memory and compute-efficient tactics which allow us to deploy TTA on-device, even under quantized paradigms. LeanTTA consists of two components: (1) stateless, backpropagation-free statistics adjustment using an efficient calculation of the Mahalanobis distance and (2) partial adaptation through layerwise updates. LeanTTA reaches up to a 16% reduction in corruption error on CIFAR10 with minimal additional compute time and memory required over normal inference, demonstrating the potential of backpropagation-free methods for efficient on-device TTA.
Code: https://github.com/cam-mobsys/LeanTTA.

1 Introduction

Test-Time Adaptation (TTA) promises robust model deployment under distribution shifts, without access to data labels. For small models compressed to fit on edge hardware, TTA on-device can provide extra adaptability lost to quantization or compression—important, given that edge deployments encounter in-the-wild data which may be not only missing from the training set, but also highly limited in quantity, such as one recording of a rare bird under unusual conditions, or a single photo of a rare skin condition in a remote clinic [7, 27, 50]. However, mainstream approaches rely heavily on backpropagation and large batch sizes [52], which become untenable within edge limitations, given memory constraints, the instability of quantized fine-tuning, and low data availability. We find in this work that small batch sizes can in fact collapse model accuracy to random guessing with most backpropagation-driven TTA methods, illustrating the incompatibility of most TTA methods and on-device deployment.

To address this gap, we introduce **LeanTTA**, a combination of memory- and compute-efficient tactics which allow us to deploy TTA on-device, even under low-data and quantized paradigms. We first present stateless, backpropagation-free statistics adjustment via Mahalanobis distance, then demonstrate the accuracy and efficiency benefits of selective layerwise adaptation, which allow for fusion of unadapted layers for further computational gains. We evaluate on a common edge architecture (ResNet18 [16], others in Appendix) and multiple datasets (ImageNet-C, CIFAR10-C [17], and Warblr [50]), benchmarking against SOTA methods on a resource-limited Raspberry Pi Zero 2W. Evaluated across corruption benchmarks, LeanTTA achieved up to a 16% reduction in corruption error ($\frac{Adaptation-Corruption}{Baseline-Corruption}$, calculated on CIFAR-10 with a baseline of 93%) *across both gradual and abrupt shifts* while operating within an 11.2 MB peak memory footprint on ResNet18,

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: AXIOM: Foundations of Efficient Deep Learning.

delivering an order-of-magnitude reduction in memory consumption over similarly-performing TTA methods.

2 Background and Related Works

TTA aims to reduce the effect of test-time covariate shift through fine-tuning or direct manipulation of the source statistics frozen in batch normalization layers at test time [27]. Normalization layers ensure that intermediate activations arrive at subsequent layers with a standardized mean and variance [21, 46]; when deployment-time data arrives with shifted statistics, frozen training statistics cannot standardize activations into the operating region expected by other layers [33].

The early work Tent [52] and many variants [31, 42, 43, 53] rely on entropy-supervised backpropagation, where batch normalization statistics are *continuously* updated using gradient descent, supervised by entropy (and thereby confidence) as a proxy for accuracy drops due to covariate shift. Methods like [31, 42, 43] iterate on entropy-backpropagation methods through selective gradient updates, while methods like [15, 53, 58] aim to decrease model collapse and error accumulation due to continual adaptation. For example, MECTA [19] aims to reduce memory through pruning and selective activation caching, and TinyTTA [24] through early-exit ensembles on microcontrollers. These works require backwards passes and gradient calculations—untenable with limited compute.

Forward-only adaptation as proposed in [33, 44, 51, 57] improves memory- and compute efficiency. TEMA [51] estimates statistics over previous test batches, FOA [44] relies on a particular architecture, and PEA [57] lifts this restriction yet relies on an exponential moving average of test-time statistics.

LeanTTA builds on the efficiency of forward-only methods and adds several features. To mitigate degradation from continual adaptation, we demonstrate that adaptation relative to *original* source statistics *per-instance* is sufficient to improve accuracy. LeanTTA also leverages for more compute efficiency the finding that shallow-layer adaptation matches or exceeds full-model adaptation, and proposes fusion of non-adaptive layers for further memory and compute savings (see extended discussion in Appendix A.1).

3 The Method

Problem Description: A model $f_\theta(\mathbf{x})$ is trained over a source dataset with inputs and labels x_s, y_s, sampled from the joint probability distribution $P_{source}(x_s, y_s)$. At test-time, the model $f_\theta(\mathbf{x})$ with parameters θ encounters inputs x_t with unknown labels y_t, where x_t, y_t are distributed over $P_{target}(x_t, y_t)$, and $P_{target}(x_t, y_t)$ may have diverged from $P_{source}(x_s, y_s)$. The model $f_\theta(\mathbf{x})$ must adapt to any differences between P_{source} and P_{target} with access only to x_t.

Given a pretrained model, LeanTTA updates and adapts normalization layers to target statistics while maintaining its performance on the source domain. We discuss the four key steps below.

Extract Statistical Information: We extract statistical information from the input to each normalization layer x_t^l. We then calculate the mean ($\mu_t \in \mathbb{R}^c$) and variance ($\sigma_t^2 \in \mathbb{R}^c$): $\mu_t \leftarrow \frac{1}{H \times W} \sum x_t^l$, $\sigma_t^2 \leftarrow Var(x_t^l)$. H and W represent the height and width of the input data per-channel, respectively, where c is the number of channels. The calculated target statistics are combined with the saved source statistics at that layer (μ_s, σ_s^2) using a weighted average, parameterized by τ: $\mu_b \leftarrow \tau\mu_s + (1-\tau)\mu_t$, $\sigma_b^2 \leftarrow \tau\sigma_s^2 + (1-\tau)\sigma_t^2$.

Target statistics, though informative, are highly variable between single data samples and cannot directly replace source statistics. Instead, we use an estimated moving average via τ, which determines the weight assigned to the source (τ) and the target statistics ($1-\tau$).

Calculate Sample Divergence: To effectively adapt to single data points, we calculate a distance measure, d, incorporating Mahalanobis distance [10] to measure the divergence between source and our now-stabilized target distributions: $d \leftarrow 1 - e^{-(\mu_b - \mu_s)^T (\Sigma_s^2)^{-1} (\mu_b - \mu_s)}$. Composing the Σ_s^2 as a diagonal variance matrix $\in \mathbb{R}^{c \times c}$, inversion can be replaced by faster simple division. We apply Mahalanobis distance as it accounts for the variance structure of both source and target distributions, and has been applied in SOTA out-of-distribution detection tasks.

Balance Source and Target: The Mahalanobis distance indicates distribution shift severity, informing whether to give more weight to the source or stabilized incoming statistics. For severe shifts, more emphasis is placed on source statistics; less-severe shifts place more emphasis on stabilized incoming statistics: $\mu_{\text{new}} \leftarrow d\lambda\mu_s + (1 - d\lambda)\mu_b$, $\sigma^2_{\text{new}} \leftarrow d\lambda\sigma^2_s + (1 - d\lambda)\sigma^2_b$.

λ ensures that even when the Mahalanobis distance d is 1, some of the target statistic is still incorporated to improve accuracy and ensure a smooth transition between source and target statistics. (See Appendix for justification of hyperparameter selection).

Normalize and Reset: The final step involves normalizing the input data x_t^l using the updated statistics, resulting in the output x_t^{l+1}: $x_t^{l+1} = \gamma \left(\frac{x_t^l - \mu_{\text{new}}}{\sigma_{\text{new}}} \right) + \beta$. We then reset the model per-sample, preventing catastrophic forgetting and ensuring robustness against rapid data shifts.

Partial Fusion Strategy: Quantized on-device backpropagation is impractical in standard deployments, requiring specialized kernels [24, 35]. The simplicity of our method makes quantization straightforward—in particular, layer fusion, which is commonly incorporated into compression pipelines. We also employ **partial layerwise adaptation** alongside **layer fusion**. Fusing convolutional and normalization layers reduces both memory cost and compute time, but it obscures batch statistics necessary for TTA. Our proposed partial-adaptation strategy fuses only deeper layers, then updates only unfused layers, allowing accuracy gains to coexist with fusion. Given that shallow layers tend to learn low-level, corruption-related features as opposed to the more structural features learned in deeper layers [20], this is a theoretically sensible method.

4 Results

4.1 Baseline Comparisons

We establish the performance of existing TTA baselines under both an abruptly-changing, data-scarce regime, where only **one data sample per-distribution-shift** exists, and a more common gradual-shift regime, where the corruption changes gradually. Three datasets with domain shift were used: image-based datasets CIFAR10-C, ImageNet1k-C [17], and the audio datasets BirdVox-DCASE-20k (train) + Warblr (test) [50]. The Warblr dataset contains citizen-recorded birdsong with natural abrupt shifts; we shuffled the image dataset to achieve the same effect. For Warblr, we report F1 score due to class imbalance. ResNet18 [16] and VGGish [18] models were tested for image and audio, respectively. For timing and memory, we deployed all baselines on a Raspberry Pi Zero 2W.

Beginning with **abrupt shifts** (Table 1), we demonstrate that, even at batch size 64, backpropagation may fail to achieve accuracies comparable to adaptation-free models, illustrating how entropy loss suffers when test-time statistics are inconsistent (abrupt domain shift) or noisy (small batches).

Under abrupt shifts, our method achieves **the only consistent accuracy improvements** (11.8% and 4.6% error reduction for CIFAR10-C and ImageNet-C). At the same time, our backpropagation-free method allows us to easily quantize and deploy LeanTTA on CPU-only devices. Post-quantization, our model requires 11.2 MB, an order-of-magnitude less than batch-64 Tent and EATA.

Though MECTA at batch-16 (in lieu of 64 per author recommendation) runs faster than LeanTTA with batch-1, high timing variance exposes an underlying problem: with noisy gradients, many batches are deemed detrimental for adaptation and skipped by MECTA, limiting adaptation efficacy.

TTA is rarely evaluated over binary classification tasks such as the Warblr dataset. Here, Tent outperforms LeanTTA with batch-64; similarly to trends CIFAR10 and ImageNet, performance on Warblr is good when batch sizes are large. However, we found backpropagation with batch size 64 on the Raspberry Pi was impractical due to runtime and memory constraints.

The relatively improved performance of baselines under **gradual shift** at batch size 64 (Table 1) demonstrates the reliance of most TTA algorithms on statistically even batches. While LeanTTA may not be the best-performing algorithm under this paradigm, it consistently demonstrates accuracy improvements at a low memory footprint and batch size of one. Our method is well-suited for consistent, resource-efficient and fast adaptation but may not match traditional TTA when full resources are available or when statistics are less noisy; engineers must select the best choice for their hardware and use case.

Overall, we demonstrate that *existing baselines underperform in both accuracy and system performance under the realistic deployment scenarios of data scarcity and edge device constraints*. LeanTTA avoids this trap by utilizing stabilized forward-only statistics updates to **consistently** improve accuracy under system constraints and quantization.

Table 1: LeanTTA versus baselines on different batch sizes (BS). N=5 trials. CIFAR10 and ImageNet report accuracy under abrupt and gradual shifts on ResNet18; Warblr reports scaled F1 on VGGish. Memory (MB) and Runtime (s) are reported for ResNet18. Grey = accuracy decrease, **Bold = best**, Underline = runner-up. We did not run MECTA on Warblr due to MECTA's implementational incompatibility with a binary classifier.

		CIFAR10		ImageNet		Warblr	Efficiency	
	BS	Abrupt	Gradual	Abrupt	Gradual	(f1)	Mem (MB)	Time (s)
None (32)	1	76.3±0.0	76.3±0.0	34.1±0.5	34.5±0.4	71.1±0.0	44.7	0.11±0.04
None (8)	1	75.9±0.1	75.9±0.1	–	–	70.1±0.0	11.2	0.16±0.0
Tent	1	9.8±1.6	9.0±0.0002	0.12±0.02	0.47±0.04	47.1±26.1	48.0	0.32±0.19
	64	69.6±1.7	78.4±3.73	32.3±0.5	45.4±0.4	**73.2±0.16**	101.1	0.83±0.20
EATA	1	9.0±0.0	8.96±0.0	0.10±0.05	0.05±0.01	44.1±3.7	47.2	0.15±0.22
	64	75.3±0.74	**83.0±1.1**	37.0±0.13	**46.7±0.4**	69.1±0.0	100.3	0.44±0.18
DeYO	1	9.96±0.2	10.1±0.1	0.11±0.02	0.13±0.02	60.5±0.0	47.2	0.13±0.18
	64	70.2±1.2	76.6±0.6	35.7±0.2	45.0±0.5	69.0±0.1	100.3	1.21±0.09
MECTA	1	72.4±0.4	78.5±0.3	15.3±0.8	17.1±0.2	–	46.4	0.21±0.06
	16	75.4±0.3	80.5±0.3	32.2±0.7	41.0±0.5	69.4±0.08	59.8	**0.10±0.14**
SAR	1	10.02±0.2	10.2±0.1	0.12±0.02	0.12±0.03	57.8±1.2	47.2	0.62±0.11
	64	76.2±0.6	79.8±0.5	35.7±0.4	45.9±0.43	69.2±0.1	100.3	2.00±0.16
TEMA	1	63.4±0.1	78.1±1.8	28.9±0.2	43.1±0.6	66.2±0.1	44.7	3.89±0.11
Ours (32)	1	**79.1±0.0**	79.1±0.0	**37.1±0.5**	37.1±0.7	72.5±0.0	44.7	0.21±0.01
Ours (8)	1	78.8±0.0	78.8±0.0	–	–	71.7±0.0	**11.2**	0.19±0.02

4.2 Layer-wise Updates with Adaptive Fusion

Building on the advantages of our stateless, forward-only method, we found that we can achieve further computational gains while preserving (if not improving) accuracy, by applying LeanTTA to a partial subset of normalization layers within the model, allowing for the adaptive fusion of the unadapted layers. Fusing the deep and adapting the shallow halves of ResNet18 nets a 1.6x speedup while effectively matching the accuracy gains of full adaptation (Table 2).

Our novel adaptation-plus-fusion method further improves the resource efficiency of LeanTTA while retaining our previously proven consistent accuracy gains. We also reveal an interesting trait of adaptation — effectiveness appears to saturate in the shallow half of the vision models (Table 2). Further investigation of the differential effectiveness of TTA on different layers is needed.

5 Discussion and Conclusion

In this work, we first identify gaps in the suitability of existing TTA methods for efficient edge deployment, namely, high memory and data consistency requirements and reliance on backpropagation, which may not be accurate under quantization. We then propose LeanTTA, a stateless, forward-only instance-based adaptation method combined with partial layerwise adaptation which matches or beats existing baselines under realistic abrupt data shifts while maintaining a minimal memory footprint and low runtimes.

Table 2: Accuracies and timings for ResNet18 with varying levels of fusion and adaptation.

	CIFAR10	CIFAR100	ImageNet	Timing (s)
No Fusion	78.9±1.2	50.4±3.8	37.4±0.4	0.184±0.016
Half-Fused	78.7±0.2	50.7±4.3	36.1±0.2	0.118±0.020
All-Fused	75.7±0.2	48.7±4.1	34.3±0.2	0.065±0.008

References

[1] Motasem Alfarra, Hani Itani, Alejandro Pardo, Shyma Yaser Alhuwaider, Merey Ramazanova, Juan Camilo Perez, Zhipeng Cai, Matthias Müller, and Bernard Ghanem. Evaluation of Test-Time Adaptation Under Computational Time Constraints. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 976–991. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/alfarra24a.html.

[2] Eunsu Baek, Sung hwan Han, Taesik Gong, and Hyung-Sin Kim. Adaptive camera sensor for vision models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=He2FGdmsas.

[3] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting Batch Normalization for Improving Corruption Robustness. In *2021 IEEE WACV*, January 2021. doi: 10.1109/WACV48630.2021.00054.

[4] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-Free Online Test-Time Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8344–8353, June 2022.

[5] Guohao Chen, Shuaicheng Niu, Geng Li, Yunbei Zhang, Shilin Shan, Chunyan Miao, and Jianfei Yang. EVA-0: Test-Time Model Evolution with Only Two Forward Passes per Sample, 2026. URL https://arxiv.org/abs/2605.18867.

[6] Francesco Corti, Dong Wang, Young D. Kwon, Cecilia Mascolo, and Olga Saukh. On the Interaction Between Model Compression and Test-Time Adaptation. In *Conference on Lifelong Learning Agents*, 2026. URL https://openreview.net/forum?id=aMcPqVyYhd.

[7] Xiangfeng Dai, Irena Spasić, Bradley Meyer, Samuel Chapman, and Frederic Andres. Machine learning on mobile: An on-device inference app for skin cancer detection. In *2019 fourth international conference on fog and mobile edge computing (FMEC)*, pages 301–305. IEEE, 2019.

[8] Michal Danilowski, Soumyajit Chatterjee, and Abhirup Ghosh. BoTTA: Benchmarking on-device Test Time Adaptation, 2025. URL https://arxiv.org/abs/2504.10149.

[9] Michal Danilowski, Alexander Murphy, Young D. Kwon, and Abhirup Ghosh. MORPHEUS: Meta Test-Time Adaptation via Neural Collapse Geometry. In *Third Workshop on Test-Time Updates (Main Track)*, 2026. URL https://openreview.net/forum?id=YfyApJH1Rj.

[10] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[11] Zeshuai Deng, Guohao Chen, Shuaicheng Niu, Hui Luo, Shuhai Zhang, Yifan Yang, Renjie Chen, Wei Luo, and Mingkui Tan. Test-Time Model Adaptation for Quantized Neural Networks. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3754730. URL https://doi.org/10.1145/3746027.3754730.

[12] Haoyu Dong, Nicholas Konz, Hanxue Gu, and Maciej A. Mazurowski. Medical image segmentation with intent: Integrated entropy weighting for single image test-time adaptation, 2024.

[13] Cheng Fang, Sicong Liu, Zimu Zhou, Bin Guo, Jiaqi Tang, Ke Ma, and Zhiwen Yu. AdaShadow: Responsive test-time model adaptation in non-stationary mobile environments. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 295–308, 2024.

[14] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. NOTE: Robust Continual Test-time Adaptation Against Temporal Correlation, 2023.

[15] Taesik Gong, Yewon Kim, Taeckyung Lee, Sorn Chottananurak, and Sung-Ju Lee. SoTTA: Robust test-time adaptation on noisy data streams. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[16] 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*, pages 770–778, 2016.

[17] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[18] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *2017 IEEE ICASSP*, pages 131–135, 2017. doi: 10.1109/ICASSP.2017.7952132.

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

[20] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR*, abs/1703.06868, 2017. URL http://arxiv.org/abs/1703.06868.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[22] Yusuke Iwasawa and Yutaka Matsuo. Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2427–2440. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/1415fe9fea0fa1e45dddcff5682239a0-Paper.pdf.

[23] Hong Jia, Young D. Kwon, Dong Ma, Nhat Pham, Lorena Qendro, Tam Vu, and Cecilia Mascolo. UR2M: Uncertainty and Resource-Aware Event Detection on Microcontrollers. In *2024 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, 2024. doi: 10.1109/PerCom59722.2024.10494467.

[24] Hong Jia, Young D. Kwon, Alessio Orsino, Ting Dang, Domenico Talia, and Cecilia Mascolo. TinyTTA: Efficient Test-time Adaptation via Early-exit Ensembles on Edge Devices. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 43274–43299. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4c454d34f3a4c8d6b4ca85a918e5d7ba-Paper-Conference.pdf.

[25] Hong Jia, Young D. Kwon, Dong Ma, Nhat Pham, Lorena Qendro, Tam Vu, and Cecilia Mascolo. A cascade framework for on-device uncertainty-aware event detection on microcontrollers. *Pervasive and Mobile Computing*, 119:102208, 2026. ISSN 1574-1192. doi: https://doi.org/10.1016/j.pmcj.2026.102208. URL https://www.sciencedirect.com/science/article/pii/S1574119226000490.

[26] Ansh Khurana, Sujoy Paul, Piyush Rai, Soma Biswas, and Gaurav Aggarwal. SITA: single image test-time adaptation. *CoRR*, 2021.

[27] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, et al. WILDS: A benchmark of in-the-wild distribution shifts. *CoRR*, abs/2012.07421, 2020.

[28] Young D. Kwon, Jagmohan Chauhan, and Cecilia Mascolo. YONO: Modeling Multiple Heterogeneous Neural Networks on Microcontrollers. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 285–297, 2022. doi: 10.1109/IPSN54338.2022.00030.

[29] Young D. Kwon, Jagmohan Chauhan, Hong Jia, Stylianos I. Venieris, and Cecilia Mascolo. LifeLearner: Hardware-Aware Meta Continual Learning System for Embedded Computing Platforms. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*, SenSys '23, page 138–151, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704147. doi: 10.1145/3625687.3625804. URL https://doi.org/10.1145/3625687.3625804.

[30] Young D. Kwon, Rui Li, Stylianos I. Venieris, Jagmohan Chauhan, Nicholas D. Lane, and Cecilia Mascolo. TinyTrain: Resource-Aware Task-Adaptive Sparse Training of DNNs at the Data-Scarce Edge. In *International Conference on Machine Learning*, 2024.

[31] Jonghyun Lee, Dahuin Jung, Saehyung Lee, Junsung Park, Juhyeon Shin, Uiwon Hwang, and Sungroh Yoon. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Taeckyung Lee, Sorn Chottananurak, Junsu Kim, Jinwoo Shin, Taesik Gong, and Sung-Ju Lee. Test-time adaptation with binary feedback. In *International Conference on Machine Learning*, 2025.

[33] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation, 2016.

[34] 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. doi: 10.1007/s11263-024-02181-w. URL https://doi.org/10.1007/s11263-024-02181-w.

[35] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256kb memory, 2024.

[36] Ke Ma, Jiaqi Tang, Bin Guo, Fan Dang, Sicong Liu, Zhui Zhu, Lei Wu, Cheng Fang, Ying-Cong Chen, Zhiwen Yu, and Yunhao Liu. SURGEON: Memory-Adaptive Fully Test-Time Adaptation via Dynamic Activation Sparsity. In *CVPR*, 2025.

[37] Xiao Ma, Young D. Kwon, and Dong Ma. EmbodiTTA: Resource-Efficient Test-Time Adaptation for Embodied Visual Systems. *IEEE Internet of Things Journal*, 2026.

[38] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer, 2022.

[39] M. Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The Norm Must Go On: Dynamic Unsupervised Domain Adaptation by Normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14765–14775, June 2022.

[40] Alexander Murphy, Michal Danilowski, Soumyajit Chatterjee, and Abhirup Ghosh. NEO — No-Optimization Test-Time Adaptation through Latent Re-Centering. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mVlIKLiizr.

[41] Zachary Nado, Shreyas Padhy, D. Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating Prediction-Time Batch Normalization for Robustness under Covariate Shift, 2020. URL https://arxiv.org/abs/2006.10963.

[42] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 16888–16905. PMLR, 17–23 Jul 2022.

[43] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world, 2023.

[44] Shuaicheng Niu, Chunyan Miao, Guohao Chen, Pengcheng Wu, and Peilin Zhao. Test-time model adaptation with only forward passes, 2024.

[45] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[46] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

[47] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation, 2020. URL https://arxiv.org/abs/2006.16971.

[48] 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 (CVPR)*, pages 11920–11929, June 2023.

[49] Sudarshan Sreeram, Young D. Kwon, and Cecilia Mascolo. Tempora: Characterising the Time-Contingent Utility of Online Test-Time Adaptation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ivB3PM7XfD.

[50] D. Stowell, Y. Stylianou, M. Wood, H. Pamuła, and H. Glotin. Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge. *Methods in Ecology and Evolution*, 2018.

[51] Zixian Su, Jingwei Guo, Kai Yao, Xi Yang, Qiufeng Wang, and Kaizhu Huang. Unraveling batch normalization for realistic test-time adaptation, 2024.

[52] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization, 2021.

[53] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation, 2022.

[54] Xingyu Wang and Tao Wang. FOZO: Forward-Only Zeroth-Order Prompt Optimization for Test-Time Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2026.

[55] Junrui Xiao, Zhikai Li, Lianwei Yang, Yiduo Mei, and Qingyi Gu. TTAQ: Towards Stable Post-training Quantization in Continuous Domain Adaptation, 2024. URL https://arxiv.org/abs/2412.09899.

[56] MA Xiao, Young D. Kwon, and Dong Ma. Efficient test-time adaptation via decoupled BN update for edge devices. In *Third Workshop on Test-Time Updates (Main Track)*, 2026. URL https://openreview.net/forum?id=35oSXGUjDH.

[57] MA Xiao, Young D. Kwon, Pan Zhou, and Dong Ma. Architecture-Agnostic Test-Time Adaptation via Backprop-Free Embedding Alignment. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7kLNGaAHaw.

[58] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios, 2023. URL https://arxiv.org/abs/2303.13899.

[59] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test Time Robustness via Adaptation and Augmentation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 38629–38642. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/fc28053a08f59fccb48b11f2e31e81c7-Paper-Conference.pdf.

[60] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation, 2023. URL https://arxiv.org/abs/2306.03536.

A Appendix

A.1 Extended Related Work

TTA Methods. A number of diverse approaches to TTA have been proposed in literature, from human-in-the-loop reinforcement learning to optimize normalization statistics [32], to adaptation of the sensor settings based on model outputs [2] (refer to Liang et al. [34] for a detailed survey). Here, we focus on methods which modify model parameters at test-time in a traditional model deployment setting. The restrictive data assumptions inherent in straightforward entropy minimization undermine its robustness in practical applications [52]. Prior research efforts address several of these limitations. CoTTA [53] tackles catastrophic forgetting and continuous shifts, while NOTE [14] and RoTTA [58] both reserve batches of data to simulate independent and identically distributed (i.i.d.) data even under non-i.i.d. conditions. MEMO [59] adapts to a single test point by minimizing the marginal entropy over augmented copies of it. While these methods improve Tent's robustness, they rely on entropy-loss backpropagation, meaning their accuracy improvements depend on the optimizer and hyperparameters used [60]; using entropy loss could also induce model collapse [44]. Being backpropagation-dependent means that they are, like Tent, much more memory-intensive than standard inference. Evaluation setups matter as well: Alfarra et al. [1] show that, once adaptation speed is accounted for, simple and fast methods can outperform more sophisticated but slower ones; Tempora [49] shows that method rankings from conventional evaluation change under latency constraints; BoTTA [8] benchmarks TTA under on-device data and resource constraints; and MORPHEUS [9] observes that no single method wins across shifts.

Resource-efficient TTA. Some resource-efficient TTA methods attempt to reduce the depth or frequency of backpropagation while also controlling for the noisiness of entropy as a minimization target. EATA [42] and SAR [43] backpropagate selectively to reduce computation of unhelpful weight updates and avoid noisy gradients. Extending these works, SOTTA [15] filters out noisy samples and accuracy-damaging gradients, while DeYO [31] adds an additional filtering metric which uses structural perturbation to boost detection of distribution shift.

MECTA [19] takes a different approach by pruning the gradient cache of batch normalization layers prior to backpropagation; SURGEON [36] proposes an architecture-agnostic activation pruning method to reduce memory consumption during backpropagation. EcoTTA [48] attaches lightweight meta networks to a frozen backbone to shrink the activations stored for backpropagation. AdaShadow [13] reduces latency by separately predicting the latency and importance of model layers in adaptation, then optimizing for the best schedule of layers to activate for adaptation. However, these methods require a sufficiently large batch of samples and/or multiple passes through models; in addition, all aforementioned methods rely on backpropagation, which means their worst-case computational cost is still on-par with Tent and may involve complex implementations if quantized and moved on-device [24, 29, 30, 35].

For edge devices, where models are already deployed for inference on microcontrollers [23, 25, 28], TinyTTA [24] adapts early-exit ensembles in a batch-agnostic strategy. DBU [56] decouples the update of BN statistics from that of BN affine parameters so that the memory-heavy updates run at small batch sizes, and EmbodiTTA [37] adapts only when a shift is detected. In addition, quantization and compression add further complication. Specifically, TTAQ [55] addresses the performance degradation of post-training quantization under continual domain shift, and Corti et al. [6] show that the gap between TTA and supervised fine-tuning widens as structured compression increases.

Backpropagation-free TTA. Inference-only adaptation methods comprise a divergent branch of TTA methods. Being backpropagation-free, their memory and energy consumption yield the same order of magnitude as inference.

Li et al. [33] propose directly updating the frozen source statistics in normalization layers of a model with statistics calculated over the entirety of the target data. A more practical approach utilizes only mini-batch target statistics [3, 41], though this method fails as batch sizes approach one. Schneider et al. [47] extend the original approach by reducing the target data required through the use of a weighted average of source and target statistics. Further reducing the target data needed down to a single image, SITA [26] and InTEnt [12], respectively, use feature augmentation and entropy-weighted integration across a hyperparameter space, though both methods require multiple passes through the entire model per-adaptation. DUA [39] also adapts normalization statistics without

backpropagation, but accumulates them over the test stream, so its estimate depends on previously seen samples. FOA [44] enables forward-only adaptation but is ViT-specific, relies on a class token, and still requires large batch sizes or cached historical data. ZOA [11] reports that quantized models are more sensitive to domain shift than their full-precision counterparts, and adapts quantized CNNs and ViTs through zeroth-order optimization with two forward passes per sample; FOZO [54] and EVA-0 [5] likewise rely on zeroth-order, forward-only optimization. TEMA [51] takes a different approach that involves two forward passes, trading off the hyperparameter search for a long warm-up period of roughly 1,000 test images for batch size 1.

Other methods act on classifier outputs or embeddings: T3A [22] adjusts the classifier with prototypes built from past test samples, LAME [4] corrects the model's outputs rather than its parameters, NEO [40] re-centers target embeddings at the origin, and PEA [57] aligns intermediate embeddings with two forward passes, using an exponential moving average of test-time statistics.

Novelty of LeanTTA. Our proposed method sits at the intersection of consistent accuracy improvement, low memory footprint, and fast runtime, giving it several advantages over existing baselines. Like other methods, LeanTTA is backpropagation-free, but it eschews ensembles or augmentations and adapts dynamically per data point — even if points are sampled from vastly different domains — independent of batch size or previous data. Our statistics stabilization strategy eliminates the need for a long warm-up period, and our per-data-point reset prevents any chance of long-term model collapse. Finally, LeanTTA improves both accuracy and computational efficiency by proposing adaptation on a subset of the model and fusing the non-adaptive layers.

A.2 Methods

We illustrate the methodology behind LeanTTA visually in Figure 1.

As an algorithm, LeanTTA is designed to be forward-only and quantization-compatible, making it well-suited for resource-limited devices. It requires **only a single forward pass with a batch size of one** to improve accuracy. As each data instance progresses through the model's normalization layers, statistics are updated sequentially. Once the intermediate representation is computed at a layer, the statistics are **immediately reset**, preparing the system for the next data point and completely mitigating any concerns around catastrophic forgetting — the model returns to its original state at each iteration.

Across datasets and models, we found that $\tau = 0.9$ (simulating momentum over 9 source samples and 1 test-time sample) strikes a reasonable balance: target statistics have sufficient influence to improve accuracy, while source statistics mitigate possible accuracy losses from noise or outliers (Figure 2). Other selections could yield higher accuracy on different shift severities, but require foresight into the nature of the shift.

A.3 Further Experiment Information

Model Training Details. For the setting of on-device computation, we employ compact architectures, namely, MobileNetV2 [45] and ResNet18 [16]; we also show efficacy on a hybrid architecture through MobileViT [38]. We utilized models pre-trained on ImageNet1k, then fine-tuned the models on the training sets of CIFAR10 and CIFAR100. For the audio modality, we use VGGish [18], trained for 7,000 iterations on 16,000 BirdVox samples, with 4,000 samples held out.

Abrupt Domain Shift Simulation. Non-continuous distribution shifts change abruptly, from the perspective of the model. To simulate real-world scenarios where data comes from different domains — i.e. photographs or intermittent recordings — we randomly shuffled each image dataset across shift types and severity levels, following prior works [43, 51]. For the image classification datasets, we sampled 100 images from each of the 19 corruption categories and 5 corruption levels, then shuffled those images together. The bird datasets are, by way of being from many different people, devices, and locations, *naturally abruptly shifting*.

Baselines. We compare LeanTTA with seven SOTA TTA benchmarks: Tent [52], EATA [42], TEMA [51], CoTTA [53], SAR [43], DeYO [31], and MECTA [19], where we selected EATA as the baseline method for MECTA. We followed suggestions from their respective papers for implementation. For MECTA, we followed the advice of the authors and limited ourselves to a batch

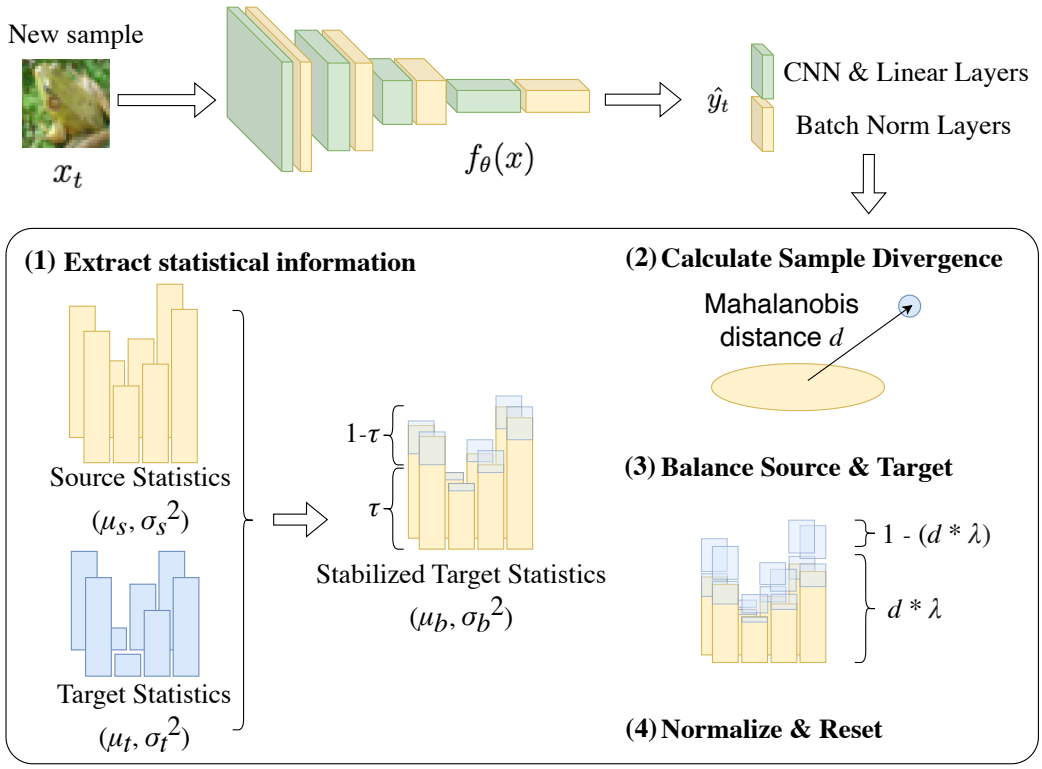


Figure 1: Proposed updates to the normalization layer. For each intermediate activation, target statistics are (2) recorded and stabilized using source statistics. The (2) Mahalanobis distance, d, estimates how far out-of-distribution the stabilized statistics have moved at each layer. The source and target statistics are then (3) balanced and recombined according to the scaled distance and used for (4) normalization.

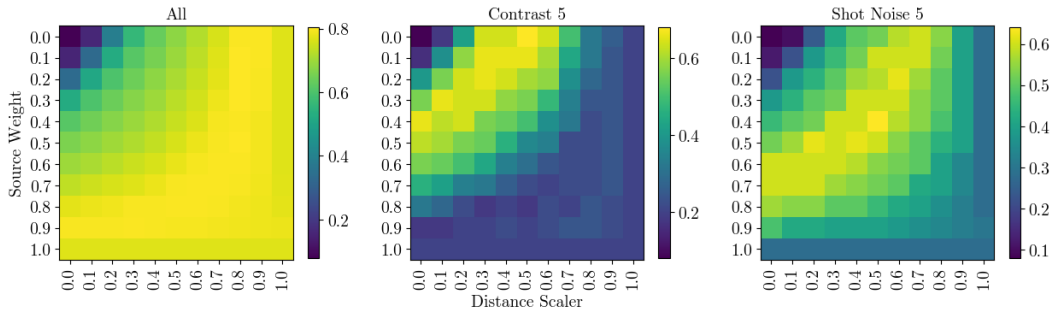


Figure 2: Hyperparameter space search for Source Weight τ and Distance Scaler λ with ResNet18 applied to CIFAR10-C. From left to right, "All" shows accuracies across hyperparameter choices from all 19 categories and 5 severity levels, whereas "Contrast 5" and "Shot Noise 5" show accuracies from the corresponding noise type and severity. Lighter shaded squares indicate higher accuracy.

size of 16 and below for best accuracy. We applied documentation-suggested hyperparameter settings, and broadly found that single-instance batches collapsed regardless of hyperparameter settings.

A.4 Evaluation Over Multiple Architectures

To demonstrate the generalizability of our algorithm to multiple architectures beyond ResNet18, we also tested LeanTTA on MobileNetV2 [45], VGGish [18], and MobileViT [38]. While all utilize convolutional layers, MobileViT notably also incorporates transformer layers.

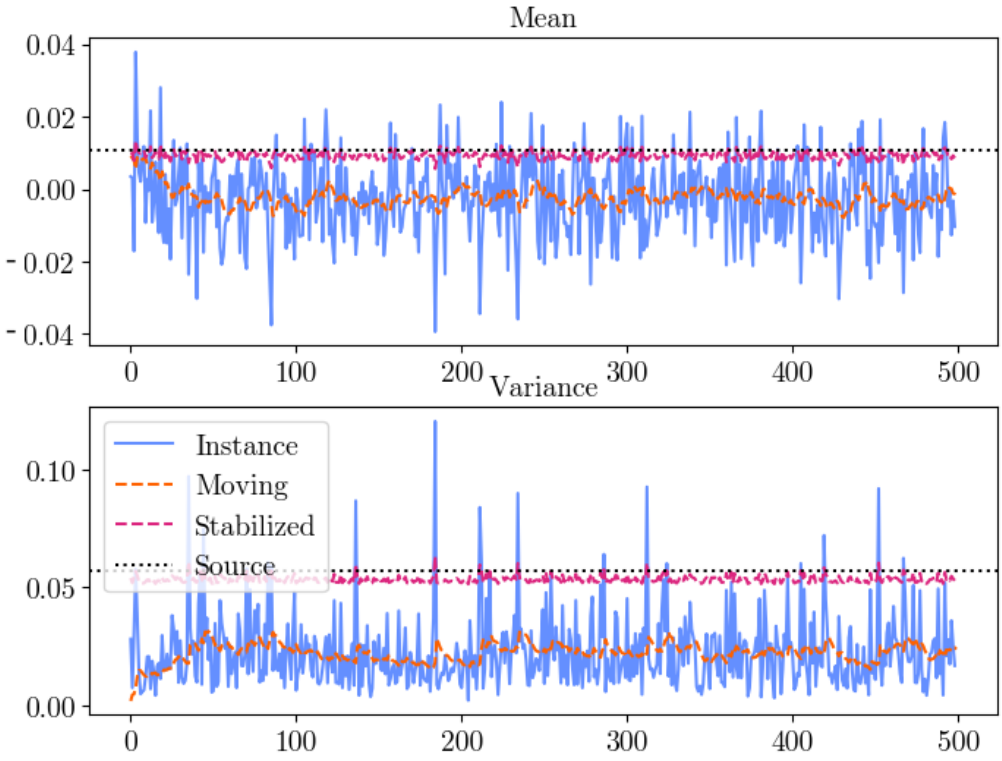


Figure 3: Mean (top) and variance (bottom) recorded from a single layer of ResNet18 on the shift-free CIFAR10 dataset for a running average with momentum=0.9 (orange), with statistics calculated at each image (blue) and with stabilized statistics (red).

Table 3 displays the memory and runtime performance across ResNet18, MobileNetV2, and VGGish on the Raspberry Pi Zero 2W. Notably, running large batch size TTA methods crashed the Raspberry Pi with VGGish and MobileNetV2. TTA over large batch sizes also demonstrated significant latency — time *per-frame*, not per-batch, stretched well into the seconds, if not tens of seconds for the more computationally intensive backpropagation-based methods. TEMA was also slow despite being forward-only, owing to its per-batch recalculation of full data distributions.

Table 4 shows the performance of LeanTTA on CIFAR10 against our baseline set. We demonstrate that our system works similarly well on modern architectures, such as hybrid vision transformers. We also show that our finding of comparable or even high accuracies obtained with partial adaptation of shallow layers hold true, even in models with varied architectures.

A.5 Fusion & Layerwise Adaptation

In Table 2, we show the comparable accuracy resultant from adaptation of only the shallow layers in a model across datasets. Our analysis underscores the utility of only conducting partial fusion — doing so allows for not only faster adaptation, but also layer fusion, as discussed in the main body of this work.

Skipping shallow layer adaptation reduced accuracy improvements, indicating that these layers are more important to adaptation (Figure 4). We hypothesize that deep layers focus on larger structures, while shallow layers learn general patterns and components. Such "stylistic" information is related to distribution shift [3]. Looking at Figure 5, we see support for this hypothesis: deeper layers (for the most part) have higher divergence between train- and test-time statistics.

Table 3: System measurements for memory (MB) and time (seconds-per-instance) for the three evaluated models on the Raspberry Pi Zero 2W. "-" indicates that the Pi was unable to run adaptation due to resource constraints. ** indicates that the loss became NaN. Bold text indicates best performance, underlined indicates second-best (with the exception of the adaptation-free baseline).

Method	**Variant**	**VGGish**		**ResNet18**		**MobileNetV2**	
		Memory	Time	Memory	Time	Memory	Time
None	BS 1	18.7	1.46±1.40	44.7	0.113±0.04	8.95	0.18±0.11
	int8	4.77	0.48±0.10	11.2	0.159±0.0	2.24	0.50±0.06
TEMA	BS 1	18.7	4.13±0.60	44.7	3.89±0.11	8.95	19.6±0.16
Tent	BS 1	78.2	2.59±1.49	48.0	0.319±0.19	36.0	0.45±0.30
	BS 16	306.8	-	60.7	0.402±0.12	138.4	4.4±0.50
	BS 64	1038.2	-	101.1	0.830±0.20	466.1	-
EATA	BS 1	63.5	1.52±1.17	47.2	0.147±0.22	29.3	**0.219±0.013**
	BS 16	292.0	-	59.8	**0.088±0.16**	131.7	3.44±0.44
	BS 64	1023.4	-	100.3	0.442±0.18	459.4	-
CoTTA	BS 1	263.8	55.31±5.39	279.8	53.9±4.5	152.4	14.8±1.16
	BS 16	1098.8	-	340.6	7.31±0.39	652.7	14.78±1.16
	BS 64	3770.9	-	535.2	5.78±1.84	2253.5	-
DeYO	BS 1	63.5	1.74±0.3	47.2	0.13±0.18	29.3	0.22±0.008
	BS 16	292.0	30.0±1.3	59.8	0.41±0.05	131.7	6.42±0.84
	BS 64	1023.4	-	100.3	1.21±0.09	459.4	8.86±0.85
MECTA	BS 1	50.1	**	46.4	0.21±0.06	23.3	0.44±0.04
	BS 16	210.0	47.8±2.2	59.8	0.102±0.14	95.0	2.83±0.87
SAR	BS 1	63.5	5.19±0.13	47.2	0.617±0.107	29.3	0.88±0.16
	BS 16	292.0	-	59.8	0.74±0.06	131.7	11.7±1.6
	BS 64	1023.4	-	100.3	2.0±0.16	459.4	16.7±1.4
LeanTTA	BS 1	18.7	1.54±0.34	44.7	0.205±0.01	8.95	0.531±0.030
	int8	**4.77**	**0.85±0.20**	**11.2**	0.185±0.021	**2.24**	0.947±0.01

Table 4: Accuracy on abruptly changing CIFAR10-C datasets using the MobileViT-s architecture (32 BN layers) fine-tuned to the CIFAR10 dataset. Our results are shown with adaptation across varying numbers of layers. N=5 trials. **Bold**: best, underline: 2nd best, grey text: worse than baseline.

Method	**Variant**	**Accuracy**
None	Batch 1	72.0 ± 0.4
TEMA	Batch 2	15.5 ± 0.4
Tent	Batch 16	69.8 ± 1.0
	Batch 64	75.2 ± 0.4
EATA	Batch 16	71.1 ± 0.7
	Batch 64	74.2 ± 0.2
CoTTA	Batch 16	57.7 ± 0.5
	Batch 64	60.9 ± 1.0
DeYO	Batch 16	71.4±0.7
	Batch 64	74.7±0.4
MECTA	Batch 16	15.7±0.7
SAR	Batch 16	70.8±0.7
	Batch 64	74.3±0.4
LeanTTA (ours)	Full Adap.	73.8 ± 0.7
	1st 8	75.3 ± 0.3
	1st 16	**75.6 ± 0.3**
	1st 24	74.8 ± 0.5

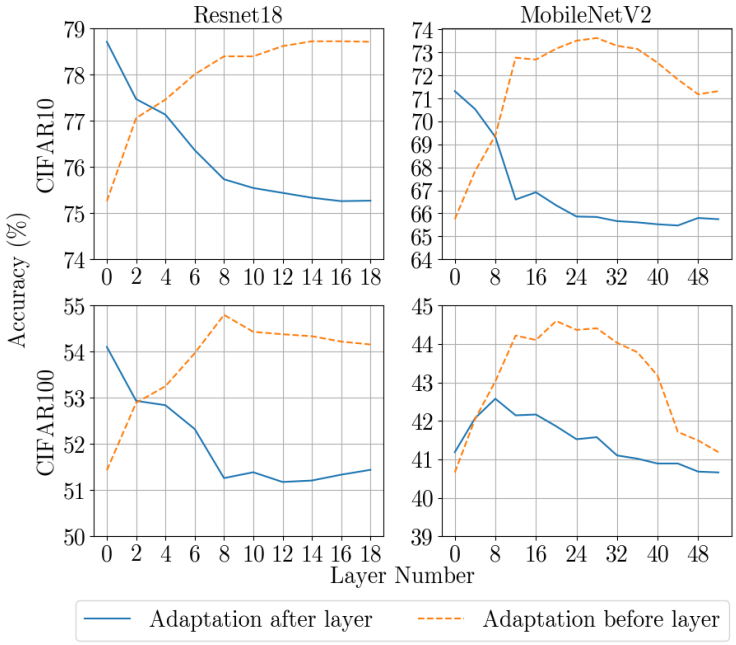


Figure 4: Layer-wise ablation analysis: (1) progressively removing adaptation from shallow layers (blue solid line) and (2) progressively adding adaptation to deeper layers (orange dotted line). Results are from full-precision (fp32) ResNet18 and MobileNetV2 evaluated on the Abrupt CIFAR10 dataset.

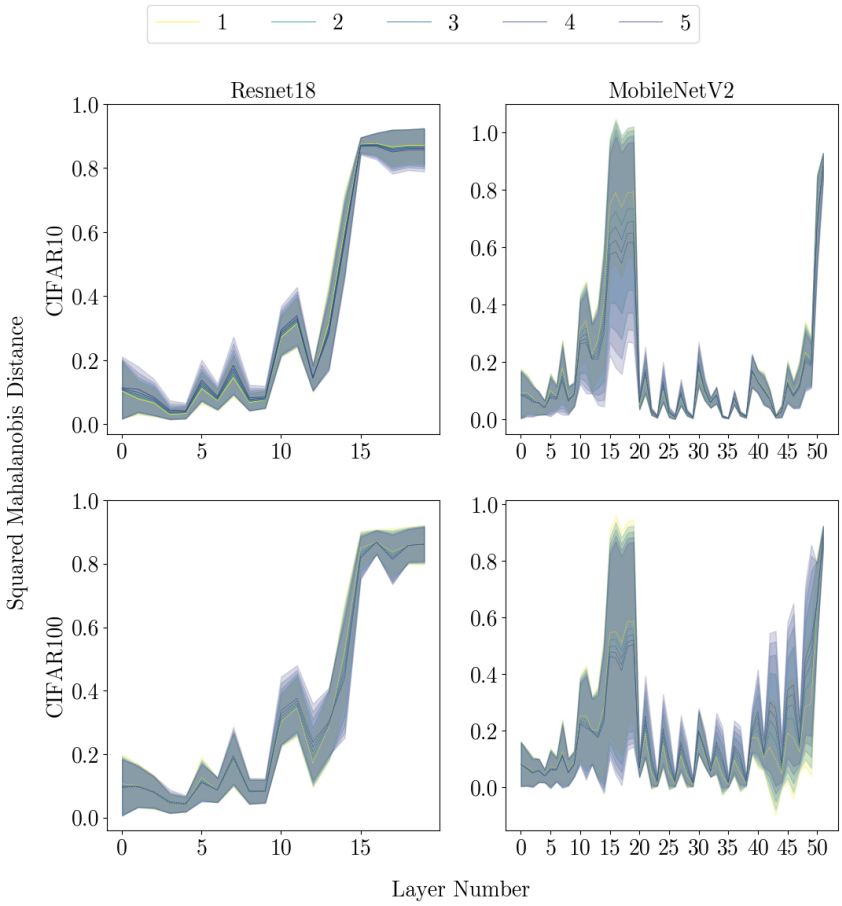


Figure 5: Layerwise Mahalanobis distance d calculated after stabilization in Equation 3, separated by severity of distribution shift in the CIFAR10/100-C datasets (light being the least severe, dark being the most). Shading indicates one standard deviation on either side of the average over N=5 trials.