

---

# LeanTTA: A Backpropagation-Free and Stateless Approach to Quantized Test-Time Adaptation on Edge Devices

---

Cynthia Dong<sup>1,2</sup> Hong Jia<sup>3,2</sup> Young D. Kwon<sup>4,2</sup> Georgios Rizos<sup>5</sup> Cecilia Mascolo<sup>5</sup>

## Abstract

While there are many advantages to deploying machine learning models on edge devices, the resource constraints of mobile platforms, the dynamic nature of the environment, and differences between the distribution of training versus in-the-wild data make such deployments challenging. Current test-time adaptation methods are often *memory-intensive* and not designed to be quantization-compatible or deployed on low-resource devices. To address these challenges, we present LeanTTA, a novel backpropagation-free and stateless framework for quantized test-time adaptation tailored to edge devices. Our approach minimizes computational costs by dynamically updating normalization statistics without backpropagation, which frees LeanTTA from the common pitfall of relying on large batches and historical data, making our method robust to realistic deployment scenarios. Our approach is the first to enable further computational gains by combining partial adaptation with quantized module fusion. We validate our framework across sensor modalities, demonstrating significant improvements over state-of-the-art TTA methods, including a 15.7% error reduction, peak memory usage of only 11.2MB for ResNet18, and fast adaptation within an order-of-magnitude of normal inference speeds on-device. LeanTTA provides a robust solution for achieving the right trade offs between accuracy and system efficiency in edge deployments, addressing the unique challenges posed by limited data and varied operational conditions.

## 1. Introduction

Performing inference directly on edge devices instead of relaying information to server-hosted models brings security, reliability, and accessibility benefits (Qayyum et al., 2020; Mollah et al., 2017; Guo & Li, 2018; Dai et al., 2019). However, deploying machine learning models at the edge is challenging: mobile devices are resource-poor in memory

and energy, and often lack the scale and power of cloud-hosted GPU acceleration (Dhar et al., 2021).

By nature, on-device models operate with varied sensors in a range of environments, processing data from distributions not seen during training (Zhou et al., 2022). A potential mobile deployment of deep learning is bioacoustics (Zuallkernan et al., 2021), specifically citizen science, where people capture images or audio of flora and fauna with their devices (Stowell et al., 2018). Another is on-device diagnostics of cough or heart sounds collected using mobile phones (Han et al., 2022; Shariat Panah et al., 2022). In such scenarios, *only a single, critical data point is available* — one rare bird video in a certain forest, or one measurement from a specific patient; distributions may shift *abruptly*. Models trained on data gathered in a lab or clinic may not anticipate varied conditions in the wild, potentially resulting in high error rates (Koh et al., 2020). Even large deep learning models can significantly drop in accuracy under previously unseen distributions (Liang et al., 2023); edge-deployed models, compressed to save memory and compute, are even less capable of generalization (Zhou et al., 2022).

Test-time adaptation (TTA), as illustrated in Figure 6, seeks to improve model accuracy under distribution shift *without access to the original source or labeled target data* (Liang et al., 2023; Koh et al., 2020). Changes in distribution can be broad and unpredictable (Liang et al., 2023; Koh et al., 2020).

Significant challenges stand in the way of on-device deployment of current TTA methods. *Backpropagation-based optimization* of normalization layers (Wang et al., 2021; Niu et al., 2023; Wang et al., 2022; Niu et al., 2022) severely strains resource-constrained devices, requiring a nontrivial increase in memory and power over backpropagation-free methods (Xu et al., 2022; Lin et al., 2024; Huang et al., 2023; Kwon et al., 2024).

Data availability at the edge further complicates on-device TTA: many TTA methods perform very poorly under temporal correlation (Wang et al., 2022; Gong et al., 2023) or limited data (Benz et al., 2021). Not only does memory constrain batch size, but in the case of low-frequency inference where data is captured intermittently (such as citizen

science apps or health signals analysis) (Stowell et al., 2018; Dong et al., 2024), *there may only be a single data point per distribution shift, and few data points overall*. Most TTA methods will *induce model collapse* with the small batch sizes and abruptly changing domains characteristic to mobile deployments, and cannot adapt to data instances collected from different domains individually.

To address these gaps, we propose a novel method of backpropagation-free, stateless, and quantized TTA for edge devices: LeanTTA, which adapts by dynamically updating the quantized model’s normalization statistics. Specifically, (1) we propose a backpropagation-free TTA module capable of adapting to each unlabeled data point regardless of prior data availability or continuity. (2) So that the module adjusts to different distribution shifts without over-reliance on incoming information, we propose a stateless statistics update mechanism based on the per-sample divergence between train-time statistics and stabilized incoming statistics. (3) Lastly, we propose layer-wise updates with adaptive fusion, a novel combination which enables rapid adaptation with quantized models and provides novel insights into how layer depth impacts the efficacy of adaptation.

By working with the ubiquitous batch normalization layer (Ioffe & Szegedy, 2015), present in most edge models, such as MobileViT (Mehta & Rastegari, 2022), EfficientNet (Tan & Le, 2020), and ShuffleNet (Zhang et al., 2017), our work remains generalizable in the edge model space.

We evaluated LeanTTA on a typical edge device, over different sensor modalities and datasets, and conducted extensive experiments. Figure 1 shows that of the assessed systems, LeanTTA alone *consistently* reduces distribution shift error with low latency and memory consumption on par with inference — for both quantized and unquantized models. LeanTTA provides a robust and efficient solution, which achieves high system efficiency on edge devices, addressing the unique challenges posed by limited system resources, scarce data, and diverse operational conditions.

## 2. Related Work

### 2.1. Conventional TTA

The restrictive data assumptions inherent in Tent undermine its robustness in practical applications. Later research efforts consider several of these limitations. CoTTA (Wang et al., 2022) tackles catastrophic forgetting and continuous shifts, while NOTE (Gong et al., 2023) and RoTTA (Yuan et al., 2023) both reserve batches of data to simulate independent and identically distributed (i.i.d.) data even under non-i.i.d. conditions. While these methods improve on Tent’s robustness, they rely on entropy-loss backpropagation, meaning their accuracy improvements may be dependent on the optimizer and hyperparameters used (Zhao et al., 2023); entropy

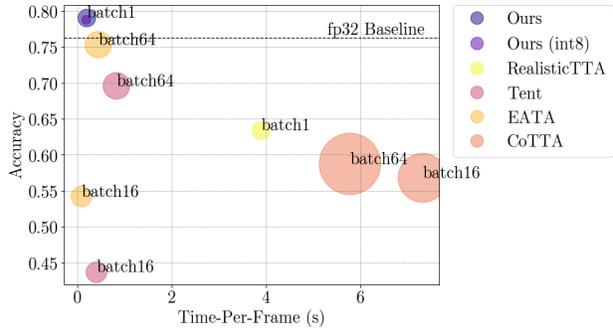


Figure 1. LeanTTA comparison against SOTA TTA methods under abruptly changing distribution shifts. Circle size is scaled to the maximum memory required for adaptation. Labels indicate batch size. For methods where models collapsed at batch size one, results were omitted. Results below the baseline indicate that adaptation worsened accuracy. The data are comprised of shuffled images, sampled across all distributions — the method is introduced in B.1.

loss may also lead to model collapse (Niu et al., 2024). Being backpropagation-dependent means that they are also, like Tent, much more memory-intensive than standard inference.

### 2.2. Memory-efficient TTA

Some memory-efficient TTA methods attempt to reduce the depth or frequency of backpropagation. EATA (Niu et al., 2022) and SAR (Niu et al., 2023) backpropagate selectively to reduce the computation requirements and avoid noisy gradients. MECTA (Hong et al., 2023) takes a different approach to improving memory efficiency, by combining pruning and selective training. However, reliance on backpropagation means that, for all aforementioned methods, their worst-case computational cost is still on-par with Tent, and may involve complex implementations if quantized and moved on-device.

### 2.3. Backpropagation-free TTA

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

Li et al. (2016) propose directly updating the frozen source statistics in normalization layers of a model with statistics calculated over the entirety of the target data (Li et al., 2016). A more practical approach utilizes only mini-batch target statistics (Benz et al., 2021), though this method fails as batch sizes approach one. An alternative method extends the original approach by reducing the target data required through the use of a weighted average of source and target statistics (Schneider et al., 2020). Further reducing the target data needed down to a single image, SITA (Khurana et al.,

2021) and InTEnt (Dong et al., 2024), respectively, use feature augmentation and entropy-weighted integration across a hyperparameter space, though both these methods require multiple passes through the entire model per-adaptation. FOA (Niu et al., 2024) enables forward-only adaptation, but is ViT-specific and still requires large batch sizes or cached historical data. RealisticTTA takes a different approach involving two forward passes, trading off the hyperparameter search for a long warm-up period of roughly 1,000 test images for batch size one (Su et al., 2024).

Compared with existing baselines, our proposed method has several advantages. Like other methods, LeanTTA is backpropagation-free; yet, 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 warmup period, and our per-data-point reset prevents any chance of long-term model collapse. Finally, LeanTTA improves accuracy alongside computational efficiency by proposing adaptation on a subset of the model and fusing the non-adaptive layers.

### 3. The Method

#### 3.1. Problem Description

We formulate the test-time adaptation problem as follows: a given 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  $\theta$  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  $x_t$ . For realistic on-device deployment on edge devices, our method must also function on models that have been compressed from floating point (32-bit) to integer precision (8-bit).

Ideally, TTA for edge devices should adapt to scarce data, require limited hyperparameter tuning, and function well across different modalities. In addition, given that data encountered in the wild are unlikely to perfectly match the distribution of finite training data, TTA should be widely applied (Koh et al., 2020). As such, the method must be lightweight, quantization-compatible, low-latency, and memory-efficient, and should consume little power relative to normal inference.

#### 3.2. LeanTTA

The proposed method LeanTTA is illustrated in Figure 2. Given a pretrained model, the system updates and adapts normalization layers to target domain statistics while maintaining its performance on the source domain. This process

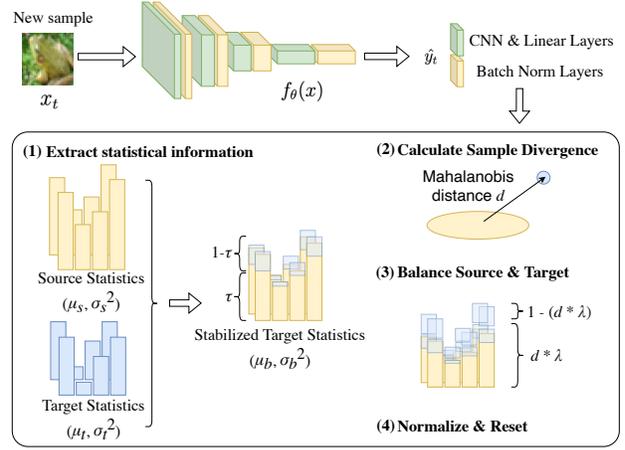


Figure 2. Proposed updated normalization layer. For each intermediate activation, target statistics are recorded and stabilized using source statistics. The Mahalanobis distance,  $d$ , estimates how far out-of-distribution the stabilized statistics have moved at each layer. The source and target statistics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

consists of four key steps: (1) extract and stabilize incoming statistics, (2) calculate sample divergence, (3) balance source and target statistics, and (4) normalize and *reset* the model to adapt to the next sample without any chance of model collapse.

##### 3.2.1. EXTRACT STATISTICAL INFORMATION

Our method is designed to adapt to single data instances under unknown distribution shifts, enhancing performance in both continuous and sharply-changing domains. The process begins by extracting statistical information from the input data  $x_t^l$ . We calculate the mean ( $\mu_t$ ) and variance ( $\sigma_t^2$ ) per-feature as follows:

$$\mu_t \leftarrow \frac{1}{H \times W} \sum x_t^l \quad (1)$$

$$\sigma_t^2 \leftarrow Var(x_t^l) \quad (2)$$

Here,  $H$  and  $W$  represent the height and width of the input data, respectively. These target statistics are combined with source statistics ( $\mu_s, \sigma_s^2$ ) using a weighted average, parameterized by  $\tau$ , to stabilize the target statistics:

$$\mu_b \leftarrow \tau \mu_s + (1 - \tau) \mu_t \quad (3)$$

$$\sigma_b^2 \leftarrow \tau \sigma_s^2 + (1 - \tau) \sigma_t^2 \quad (4)$$

Directly replacing source statistics with target statistics from a single data point — highly variable, as shown by the solid blue line in Figure 3 — lowers accuracy precipitously. Here, we first stabilize incoming statistics with the original

training statistics, simulating momentum using  $\tau$  without relying on prior data (Figure 3, pink dashed line).  $\tau$  is a parameter ranging between 0 and 1 which determines the weight assigned to the source ( $\tau$ ) and the target statistics ( $1 - \tau$ ). This balance allows for a nuanced stabilization depending on  $\tau$ .

We found that, across datasets and models,  $\tau = 0.9$  stabilizes the statistics such that, if  $\lambda$  is high enough, accuracy not only increases, but will not decrease under distribution shift relative to when  $\tau = 1$  (no dependence on target statistics). Other selections may yield higher accuracy on different levels of distribution shift, but require foresight into the nature of the shift. See Figure 7 in the Appendix for further analysis.

### 3.2.2. CALCULATE SAMPLE DIVERGENCE

To effectively adapt to single data points, we calculate the Mahalanobis distance  $d$  (De Maesschalck et al., 2000), which has recently been applied in many SOTA out-of-distribution detection tasks (Podolskiy et al., 2021; Colombo et al., 2022), to measure the divergence between source and our now-stabilized target distributions:

$$d \leftarrow 1 - e^{-(\mu_b - \mu_s)^T (\Sigma_s^2)^{-1} (\mu_b - \mu_s)} \quad (5)$$

Here,  $(\mu_b - \mu_s)^T (\Sigma_s^2)^{-1} (\mu_b - \mu_s)$  calculates the squared Mahalanobis distance, taking into account the variance and mean of the source distribution. The exponential transforms this distance into a metric ranging from 0 to 1, and subtracting it from 1 makes  $d$  a measure of divergence, where larger values indicate greater distribution shifts.  $\Sigma_s^2$  is the diagonal variance matrix, and can be efficiently inverted.

### 3.2.3. BALANCE SOURCE AND TARGET

The Mahalanobis distance indicates distribution shift severity, thus informing whether we 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.

$$\mu_{\text{new}} \leftarrow d\lambda\mu_s + (1 - d\lambda)\mu_b \quad (6)$$

$$\sigma_{\text{new}}^2 \leftarrow d\lambda\sigma_s^2 + (1 - d\lambda)\sigma_b^2 \quad (7)$$

Here, we set  $\lambda$  as a balancing parameter to ensure that even when the Mahalanobis distance  $d$  is 1, some of the target statistic will still be incorporated to improve upon the accuracy to ensure a smooth transition between source and target statistics.

### 3.2.4. NORMALIZE AND RESET

The final step involves normalizing the input data  $x_t^l$  using the updated statistics, resulting in the output  $x_t^{l+1}$ :

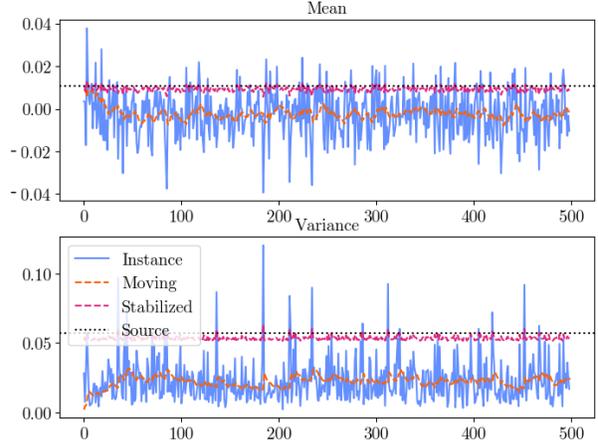


Figure 3. Mean (top) and variance (bottom) recorded from a single layer of ResNet18 on the the shift-free CIFAR10 dataset, for a running average with momentum=0.9 (orange), with statistics calculated at each image (blue), and with stabilized statistics (red).

$$x_t^{l+1} = \gamma \left( \frac{x_t^l - \mu_{\text{new}}}{\sigma_{\text{new}}} \right) + \beta \quad (8)$$

We reset the model at each data instance, preventing catastrophic forgetting and ensuring robust performance across diverse data shifts.

As such, our method 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 TTA 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.

### 3.3. Partial Fusion Strategy

Our method’s inherent efficiency and compatibility with quantization is because it is backpropagation-free. When these methods are quantized, they necessitate real, quantized training on-device, which is typically infeasible without specially designed kernels to facilitate stable training, as discussed in previous studies (Gholami et al., 2021; Lin et al., 2024).

In this study, we introduce a partial-fusion quantized TTA strategy applicable to any pretrained models. We not only circumvent reliance on specially-designed kernels, but also further improve the efficiency of LeanTTA. We were inspired by results from Figure 4: updating the first half of the model achieves similar or higher accuracy compared to updating the entire model across different architectures and datasets. From Algorithm 1 (steps 1-8), given the set

**Algorithm 1** Backpropagation-Free Quantized Test-Time Adaptation

---

**Require:** Model  $f_\theta$ ; Data  $X_s, X_t$ , Unfused Layers  $L$

- 1 Begin quantization calibration **for**  $x_s$  in  $X_s$  **do**
- 2    $\lfloor f_\theta(x_s)$
- 3 **for**  $l$  in  $L$  **do**
- 4    $\lfloor$  Separate BN and Conv layer,  $l$ , in  $f_\theta$
- 5 **for**  $l$  not in  $L$  **do**
- 6    $\lfloor$  Fuse a layer,  $l$ , in  $f_\theta$
- 7 Quantize the fused model  $\hat{f}_\theta$  **for** normalization layer  $l$  **do**
- 8    $\lfloor$  Replace  $l$  with LeanTTA layer
- 9 Save original  $\theta$  for normalization layers **for**  $x_t$  in  $X_t$  **do**
- 10    $\lfloor \hat{y}_t \leftarrow \hat{f}_\theta(x_t)$  Reset all normalization layers

---

of unfused layers for architecture, our proposed partial fusion quantized strategy fuses only deeper layers. Then, it updates only the unfused layers based on our TTA method (Section 3.2).

This strategy further improves the computational efficiency of the update step of LeanTTA, as the unfused layers are slower than fused layers with QNNPACK, the quantization engine which is currently supported on ARM architectures. Moreover, fused layers do not require storing BN layer parameters and activations for the update step.

## 4. Results & Discussion

### 4.1. Experimental Setup

We briefly explain experimental setups in this section. For more details, refer to Appendix B.

**Datasets.** We employ commonly used TTA datasets such as CIFAR10-C and CIFAR100-C (Hendrycks, 2019a;b) and realistic, on-device applications with different modalities, specifically the audio datasets, such as BirdVox-DCASE-20k (Stowell et al., 2018) and Warblr (Stowell et al., 2018) (bird song detection task). We use BirdVox to pretrain our model for audio modality and test it using Warblr for the abrupt domain shift.

**Models.** For image modality, we employ two compact architectures, namely, MobileNetV2 (Sandler et al., 2018) and ResNet18 (He et al., 2016). Then, for audio modality, we adopt VGGish (Hershey et al., 2017). We highlight the adaptability of our method across varying model architectures and underscore its robustness in handling datasets of increasing complexity.

**Abrupt Domain Shift.** To simulate the real-world scenario of abruptly changing domains (e.g., rare events like photographs or intermittent recordings), we used two datasets:

(1) CIFAR10-C and CIFAR100-C shuffled randomly across shift types and severity levels, following prior works (Niu et al., 2023; Su et al., 2024) and (2) the Warblr dataset collected under diverse conditions and sensors intermittently.

**Gradual Domain Shift.** As opposed to the abruptly changing datasets above, gradually changing datasets illustrate a different use case. Here, the data changes domain slowly, from severity level 1, to 5, then from 5 back to 1, then to the next domain. We use CIFAR10/100-C datasets to simulate this. Label shift is still present, as we otherwise used the same sampling technique as above.

**Baselines.** We compare LeanTTA with four SOTA TTA benchmarks: Tent (Wang et al., 2021), CoTTA (Wang et al., 2022), EATA (Niu et al., 2022), and RealisticTTA (Su et al., 2024). In implementing the TTA methods, we followed the suggestions from their respective papers.

**Device.** For system measurements, we employ Raspberry Pi Zero 2W, with 4 GB of swap memory available. We measure end-to-end latency. Then, we approximate the energy using a power draw recording device, whose measurements are sampled throughout and averaged, then multiplied by time to obtain power in watt-hours.

**Quantization.** Quantized models were generated using the QNNPACK backend using PyTorch; post-static quantization scales and zero-points were recorded over 100 batches of size 64 from the CIFAR10/CIFAR100 training datasets. When necessary, layer fusion was done on adjacent convolutional and batch normalization layers.

### 4.2. LeanTTA Accuracy Improvement

#### 4.2.1. ABRUPTLY CHANGING DOMAINS

Under abrupt shifts, applying state-of-the-art TTA methods, even at large batch sizes, can lower accuracy instead of increasing it. As shown in Table 1, across all distribution shifts in the CIFAR10/100-C datasets, and across the natural heterogeneity of the Warblr dataset, LeanTTA more consistently improves accuracy relative to model without adaptation under abruptly changing domains. For example, it reduces shift error by 2.8% and 6.5% for ResNet18 and MobileNetV2 on CIFAR10-C, respectively. Compared to TTA-based methods, LeanTTA outperformed the best baseline, RealisticTTA, by 15.7% and 12% on CIFAR10-C and CIFAR100-C, respectively, when using ResNet18 under abruptly changing domains. When using MobileNetV2, LeanTTA achieved improvements of 27.3% on CIFAR10-C and 13.2% on CIFAR100-C. In history-free, single-batch datasets, shown in Table 1, most methods collapse, with accuracies comparable to random guessing. Even when other methods are tested with a batch size of 64—pushing the limits of many mobile devices—under abrupt domain shifts, as shown in Table 3, most fail to improve upon the

Table 1. % Accuracy on abruptly and gradually changing datasets for batch size = 1. N=5 trials. **Bold**: best, underline: 2nd best, *italics*: worse than baseline.

Method	Variant	CIFAR10-C Abrupt		CIFAR10-C Gradual		CIFAR100-C Abrupt		CIFAR100-C Gradual	
		ResNet18	MobilenetV2	ResNet18	MobilenetV2	ResNet18	MobilenetV2	ResNet18	MobilenetV2
None	Batch 1	76.3±0.0	67.0±0.0	76.3±0.0	67.0±0.0	51.4±0.0	40.7±0.0	51.4±0.0	40.7±0.0
	int8	75.9±0.1	66.4±0.0	75.9±0.0	66.4±0.0	51.6±0.0	40.4±0.0	51.6±0.0	40.4±0.0
RealisticTTA	Batch 1	<i>63.4±0.1</i>	<i>46.2±0.0</i>	78.1±1.8	68.4±1.5	<i>42.1±0.3</i>	<i>28.0±0.4</i>	<i>51.2±0.3</i>	<i>32.7±0.3</i>
Tent	Batch 1	<i>9.8±1.6</i>	<i>12.3±2.3</i>	<i>9.0±0.002</i>	<i>14.0±0.01</i>	<i>0.0±0.0</i>	<i>0.002±0.004</i>	<i>0.0±0.0</i>	<i>0.0±0.0</i>
EATA	Batch 1	<i>9.0±0.0</i>	<i>13.5±0.1</i>	<i>8.96±0.0</i>	<i>13.4±0.1</i>	<i>0.18±0.0</i>	<i>0.44±0.04</i>	<i>0.18±0.0</i>	<i>0.49±0.0</i>
CoTTA	Batch 1	<i>9.0±0.0</i>	<i>9.6±0.1</i>	<i>9.0±0.0</i>	<i>9.78±0.18</i>	<i>0.77±0.03</i>	<i>0.0±0.0</i>	<i>0.75±0.02</i>	<i>0.0±0.0</i>
LeanTTA (Ours)	Batch 1	<b>79.1±0.0</b>	<b>73.5±0.0</b>	<u>79.1±0.0</u>	<b>73.5±0.0</b>	<u>54.1±0.0</u>	<b>41.2±0.0</b>	<u>54.1±0.0</u>	<b>41.2±0.0</b>
	int8	<u>78.8±0.0</u>	<u>72.3±0.0</u>	<u>78.8±0.0</u>	<u>72.3±0.0</u>	<b>54.4±0.0</b>	<u>40.0±0.0</u>	<b>54.4±0.0</b>	<u>40.0±0.0</u>

baseline adaptation-free accuracy; In comparison, LeanTTA improves accuracy gains *consistently*. Methods reliant on entropy-loss backpropagation or warmup periods struggle to extract stable, representative statistics from mini-batches composed of data points collected under different distribution shifts, especially with small batches. Practically, such methods could not adapt on the first collected data point without waiting to accrue further data, while our method is history- and batch-size agnostic.

LeanTTA’s success with abrupt distribution shifts and single-instance batches extends across architectures and datasets. (See Appendix G for experiments on MobileViT). Even on the label-shifted and challenging real-world scenario presented by Warblr, our method was able to improve the weighted F1 score of the model, shown in Table 2. (Weighted F1 was used due to class imbalance). Methods with batch sizes of 16 and 64 achieved a similar improvement in performance, but no backpropagation-based methods could run with these batches on our edge device, due to memory constraints.

#### 4.2.2. GRADUALLY CHANGING DOMAINS

As expected, while LeanTTA never induces precipitous drops in accuracy, we do see that other methods outperform our method with larger batch sizes under gradually changing domains. As shown in the column marked “Gradual” in Table 1, our method improves accuracy, though not as much as backpropagation-driven methods Tent and EATA — but only when these methods are equipped with large batch sizes, which comes at a significant memory cost. For example, we were unable to realistically use Tent, EATA, and CoTTA on the Warblr dataset on our memory-constrained device. Thus, slightly lower accuracy gains may be the cost of high-efficiency adaptation. RealisticTTA also performs well, but requires a long warmup period — accuracy begins at a random guess and does not stabilize until about 1000 data points of analysis (see Table 5); it is also high-latency.

In mobile and edge applications, the continuity of distribu-

Table 2. Weighted F1 scores for TTA methods applied to the Warblr dataset. **Bold** indicates best performance, underline indicates second-best.

Method	Variant	Birds (Weighted F1)
None	Batch 1	71.1±0.0
	int8	70.1±0.0
RealisticTTA	$m = 0$	66.2±0.1
Tent	Batch 1	47.1±26.1
	Batch 16	71.0±1.4
	Batch 64	<b>73.2±0.16</b>
EATA	Batch 1	44.1±3.7
	Batch 16	69.0±0.2
	Batch 64	69.1±0.0
CoTTA	Batch 1	45.4±0.0
	Batch 16	51.9±0.9
	Batch 64	59.1±0.9
LeanTTA (Ours)	Batch 1	<u>72.5±0.0</u>
	int8	71.7±0.0

tion shifts is often unknown, or expected to change drastically between individual data instances. From Tables 1 and 3, we suggest that LeanTTA is better-suited for such environments, and for deployments where efficiency is a concern.

### 4.3. System Evaluation

When assessed for latency and memory usage, especially on-device, LeanTTA outperforms existing TTA methods capable of correcting distribution shift error (Table 4). For example, EATA is faster than LeanTTA at a batch size of 1, and sometimes 16. However, EATA does not consistently improve accuracy at these batch sizes, as shown in Table 3. Backpropagation-based methods running on memory-constrained devices must trade off improved accuracy from large batch sizes and the corresponding increased

Table 3. % Accuracy on abruptly and gradually changing CIFAR10-C datasets for backpropagation-based methods with batch sizes 16 and 64. These results may result in out-of-memory errors on highly memory-constrained devices. N=5 trials. **Bold**: best, underline: 2nd best, *italics*: worse than baseline. CIFAR100-C may be more imbalanced due to sampling. Note that Tent requires at least 16 batches to perform better than LeanTTA on the CIFAR100-C dataset, while LeanTTA requires only a single batch during TTA.

Method	Variant	CIFAR10 Abrupt		CIFAR10 Gradual		CIFAR100 Abrupt		CIFAR100 Gradual	
		ResNet18	MobileNetV2	ResNet18	MobileNetV2	ResNet18	MobileNetV2	ResNet18	MobileNetV2
None	Batch 1	76.3±0.0	67.0±0.0	76.3±0.0	67.0±0.0	51.4±0.0	40.7±0.0	51.4±0.0	40.7±0.0
	int8	75.9±0.1	66.4±0.0	75.9±0.0	66.4±0.0	51.6±0.0	40.4±0.0	51.6±0.0	40.4±0.0
Tent	Batch 16	43.7±5.3	19.9±2.8	49.3±3.5	35.2±3.1	51.0±1.1	42.77±0.79	64.3±0.4	51.2±4.0
	Batch 64	69.6±1.7	52.6±2.8	78.4±3.73	66.6±1.6	53.9±0.47	<b>44.83±0.69</b>	<b>68.5±0.6</b>	<b>58.2±1.3</b>
EATA	Batch 16	54.2±4.6	62.8±0.4	68.4±3.1	61.6±3.2	50.4±0.25	39.5±0.19	63.2±0.38	51.2±0.67
	Batch 64	75.3±0.74	65.8±0.1	<b>83.0±1.1</b>	<b>79.6±0.9</b>	49.8±0.19	39.76±0.11	62.9±1.2	<u>52.5±0.1</u>
CoTTA	Batch 16	56.8±0.2	50.4±0.3	58.0±0.7	51.9±0.4	29.0±0.33	23.17±0.28	31.8±0.2	22.8±0.04
	Batch 64	58.8±0.2	51.9±0.3	64.7±0.37	56.5±0.13	32.2±0.12	27.8±0.27	36.1±0.06	30.4±0.13
LeanTTA (Ours)	Batch 1	<b>79.1±0.0</b>	<b>73.5±0.0</b>	79.1±0.0	73.5±0.0	54.1±0.0	41.2±0.0	54.1±0.0	41.2±0.0
	int8	<u>78.8±0.0</u>	<u>72.3±0.0</u>	78.8±0.0	72.3±0.0	<b>54.4±0.0</b>	40.0±0.0	54.4±0.0	40.0±0.0

Table 4. % System measurements for memory (MB), time (seconds-per-instance), and energy (joules-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. Bold text indicates best performance, underlined indicates second-best (with the exception of the adaptation-free baseline).

Method	Variant	VGGish			ResNet18			MobileNetV2		
		Memory	Time	Energy	Memory	Time	Energy	Memory	Time	Energy
None	Batch 1	18.7	1.46±1.40	4	44.7	0.113±0.04	0.3	8.95	0.18±0.11	0.5
	int8	4.77	0.48±0.10	1	11.2	0.159±0.0	0.4	2.24	0.50±0.06	1
RealisticTTA	Batch 1	18.7	4.13±0.60	10	44.7	3.89±0.11	8.6	8.95	19.6±0.16	46
Tent	Batch 1	78.2	2.59±1.49	6	48.0	0.319±0.19	0.8	36.0	<u>0.45±0.30</u>	<u>1</u>
	Batch 16	306.8	-	-	60.7	0.402±0.12	0.9	138.4	4.4±0.50	5
	Batch 64	1038.2	-	-	101.1	0.830±0.20	1	466.1	-	-
EATA	Batch 1	63.5	<u>1.52±1.17</u>	3	47.2	<b>0.147±0.22</b>	<u>0.4</u>	29.3	<b>0.219±0.013</b>	<b>0.6</b>
	Batch 16	292.0	-	-	59.8	0.088±0.16	<b>0.2</b>	131.7	3.44±0.44	4
	Batch 64	1023.4	-	-	100.3	0.442±0.18	0.8	459.4	-	-
CoTTA	Batch 1	263.8	55.31±5.39	74	279.8	53.9±4.5	70	152.4	14.8±1.16	25
	Batch 16	1098.8	-	-	340.6	7.31±0.39	12	652.7	14.78±1.16	27
	Batch 64	3770.9	-	-	535.2	5.78±1.84	11	2253.5	-	-
LeanTTA (Ours)	Batch 1	<u>18.7</u>	1.54±0.34	<u>2</u>	<u>44.7</u>	0.205±0.01	0.6	<u>8.95</u>	0.531±0.030	<u>1</u>
	int8	<b>4.77</b>	<b>0.85±0.20</b>	<b>1</b>	<b>11.2</b>	<u>0.185±0.021</u>	0.5	<b>2.24</b>	0.947±0.01	2

latency and memory consumption. For MobilenetV2, in fact, we found that, on the Raspberry Pi Zero 2W, the backpropagation required for batch 64 overwhelms the available memory, crashing the device (see Table 4); the same for batch sizes of 16 and 64 on the VGGish model with the Warblr data. RealisticTTA, while memory-efficient, requires a significant amount of compute time even with only a forward pass, and fails to produce improvements in accuracy under certain data conditions. Thus, LeanTTA strikes the most suitable and consistent tradeoff between efficiency and accuracy improvement for edge computing environments. Refer to Appendix E for further latency analysis depending on system architectures.

#### 4.4. Impact of Layers on Adaptation

Restricting adaptation to certain layers can further improve system efficiency — even more so, in the case of quantized models, where only adapting certain layers allows us to fuse the remaining layers as part of the quantization process. Figure 4 shows the effects of ablating shallow layers closer to the input (blue, solid line) or deep layers closer to the output (orange, dashed line). Removing adaptation from deep layers does not meaningfully lower accuracy — sometimes even increasing it. Removing adaptation from shallow layers resulted in low accuracy improvements, indicating that shallow layers are more important to adaptation. We hypothesize that this is because deep layers focus on larger structures while shallow layers learn general patterns and components. Such “stylistic” information is related to

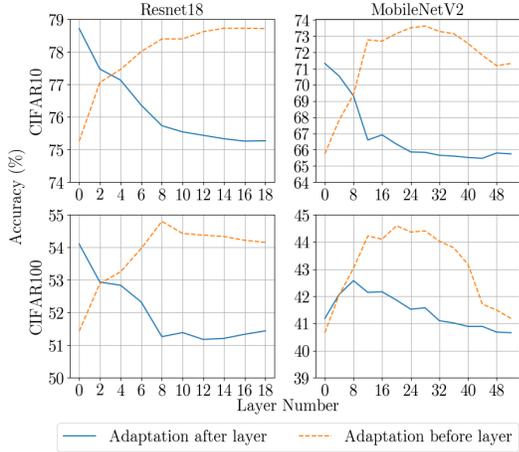


Figure 4. Layerwise ablation analysis in two directions: (1) progressively removing adaptation from shallow layers (blue solid line) and (2) progressively adding adaptation to deeper layers (orange dotted line). Results are from full-precision (fp32) Resnet18 and MobileNetV2 evaluated on the Abrupt CIFAR10 dataset.

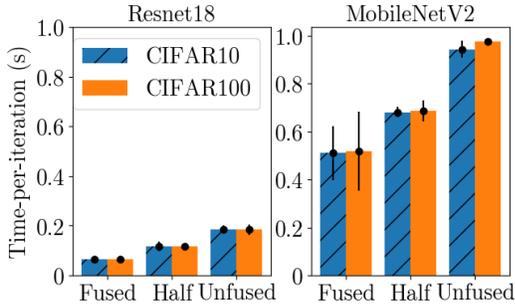


Figure 5. Fused, half-fused, and unfused quantized time-per-iteration for LeanTTA applied to ResNet18 (left) and MobileNetV2 (right) with a 500-image subset of CIFAR10-C and CIFAR100-C. For “Half”, batch normalization and convolutional layers are fused in the half of the model closest to the outputs.

distribution shift (Benz et al., 2021).

Figures 4 and 5 demonstrate that for both the CIFAR10 and CIFAR100 datasets, and across Resnet18 and MobileNetV2 architectures, applying our adaptation method to only the shallow half of the model still results in accuracy improvements. (See Appendix G for half-adaptation on MobileViT). Meanwhile, fusing the other half significantly enhances runtime efficiency. This observation led to our proposed partial fusion strategy, detailed in Section 3.3.

#### 4.5. Hyperparameters

Guided by the TTA literature (Niu et al., 2024; Benz et al., 2021), we suggest setting two hyperparameters to conservative static values. Even under different distribution shifts that might have an alternative optimal set of hyperparameters, our hyperparameter search demonstrates that by staying conservatively close to the source statistic weight, our

method will likely decrease shift error, or at the least, not worsen accuracy. In Figure 7, we can see that staying at 0.9 for both  $\tau$  and  $\lambda$  will generally keep the accuracy higher than when  $\tau = 1.0$  and  $\lambda = 1.0$  — the far lower-right square, representative of inference without adaptation.

## 5. Conclusions

Inspired by the ineffectiveness of existing TTA methods for quantized models deployed on low-resource devices, and their poor performance under diverse, realistic scarce-data scenarios, we presented LeanTTA, a novel advancement in on-device TTA. LeanTTA consistently improves accuracy under the challenging low-data, low-memory, low-power environment of edge deployments. LeanTTA dynamically analyzes each incoming data point independently based on per-sample shift severity measured with Mahalanobis distance, without having to maintain historical data or large batches. Given its layerwise nature, LeanTTA also enables adaptation alongside quantized module fusion, further optimizing the system efficiency of our proposed quantized TTA while also providing insights into the importance of layer depth to adaptation. Under data-scarce, rapidly shifting distributions, LeanTTA avoids model collapse, outperforms the state-of-the-art while also reducing latency and memory consumption.

## Impact Statement

LeanTTA looks to advance the accessibility of machine learning by enabling robust, adaptive methods on low-resource, often inexpensive devices. The system’s ability to adapt with limited data to domains not present during training broader the applications to which machine learning models can be applied, and generally improve accuracy.

**Generalizability.** The main criteria for the application of our method to a given model is the existence of normalization layers. An area of future research is the interaction of our method with layerwise and group normalization layers, or significantly different architectures, like ViT. Experiments with MobileViT can be found in the Appendix.

**Further accuracy improvements.** Our suggested method relies on the selection of two hyperparameters,  $\tau$  and  $\lambda$ . We suggest a conservative choice. One might be able to make a more optimal selection to reach even higher accuracies, but in real TTA settings, there may not be sufficient data available to tune hyperparameters. Future work could investigate the possibility of more noise-resistant metrics for calculating how far out-of-distribution an individual data point lies.

**Applicability to other data scenarios.** In this work, we focus on the data scenario where inputs to the model are widely distributed across space and time. However, our system performs well under both continual and static distribution shift, as it operates independent of prior data. For situations where distribution shift does not change, further efficiency gains are possible: investigating whether our method could adapt on a single representative image, then freeze the updated statistics until the distribution shifts again, is a promising area of future research. Problematically, this assumes foresight into the nature of test data, which is often impossible to obtain.

## References

- Benz, P., Zhang, C., Karjauv, A., and Kweon, I. S. Revisiting batch normalization for improving corruption robustness. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 494–503, 2021.
- Colombo, P., Dadalto, E., Staerman, G., Noiry, N., and Piantanida, P. Beyond mahalanobis distance for textual ood detection. *Advances in Neural Information Processing Systems*, 35:17744–17759, 2022.
- Dai, X., Spasić, I., Meyer, B., Chapman, S., and Andres, F. 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)*, pp. 301–305. IEEE, 2019.
- De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.
- Dhar, S., Guo, J., Liu, J., Tripathi, S., Kurup, U., and Shah, M. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Transactions on Internet of Things*, 2(3):1–49, 2021.
- Dong, H., Konz, N., Gu, H., and Mazurowski, M. A. Medical image segmentation with intent: Integrated entropy weighting for single image test-time adaptation, 2024. URL <https://arxiv.org/abs/2402.09604>.
- Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021. URL <https://arxiv.org/abs/2103.13630>.
- Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., and Lee, S.-J. Note: Robust continual test-time adaptation against temporal correlation, 2023.
- Guo, J. and Li, B. The application of medical artificial intelligence technology in rural areas of developing countries. *Health Equity*, 2(1):174–181, 2018. doi: 10.1089/heq.2018.0037. URL <https://doi.org/10.1089/heq.2018.0037>. PMID: 30283865.
- Han, J., Xia, T., Spathis, D., Bondareva, E., Brown, C., Chauhan, J., Dang, T., Grammenos, A., Hasthanasombat, A., Floto, A., et al. Sounds of covid-19: exploring realistic performance of audio-based digital testing. *NPJ digital medicine*, 5(1):16, 2022.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Hendrycks, D. Cifar-10-c and cifar-10-p, January 2019a. URL <https://doi.org/10.5281/zenodo.2535967>.
- Hendrycks, D. Cifar-100-c, December 2019b. URL <https://doi.org/10.5281/zenodo.3555552>.
- Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017. doi: 10.1109/ICASSP.2017.7952132.
- Hong, J., Lyu, L., Zhou, J., and Spranger, M. MECTA: Memory-economic continual test-time model adaptation.

- In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=N92hjSf5NNh>.
- Huang, K., Yang, B., and Gao, W. Elastictrainer: Speeding up on-device training with runtime elastic tensor selection. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services, MobiSys '23*, pp. 56–69, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701108. doi: 10.1145/3581791.3596852. URL <https://doi.org/10.1145/3581791.3596852>.
- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL <http://arxiv.org/abs/1502.03167>.
- Khurana, A., Paul, S., Rai, P., Biswas, S., and Aggarwal, G. SITA: single image test-time adaptation. *CoRR*, abs/2112.02355, 2021. URL <https://arxiv.org/abs/2112.02355>.
- Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A benchmark of in-the-wild distribution shifts. *CoRR*, abs/2012.07421, 2020. URL <https://arxiv.org/abs/2012.07421>.
- Kong, Q., Iqbal, T., Xu, Y., Wang, W., and Plumbley, M. D. DCASE 2018 challenge baseline with convolutional neural networks. *CoRR*, abs/1808.00773, 2018. URL <http://arxiv.org/abs/1808.00773>.
- Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images, 2009.
- Kwon, Y. D., Li, R., Venieris, S. I., Chauhan, J., Lane, N. D., and Mascolo, C. Tinytrain: Resource-aware task-adaptive sparse training of dnns at the data-scarce edge. In *International Conference on Machine Learning*, 2024.
- Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation, 2016. URL <https://arxiv.org/abs/1603.04779>.
- Liang, J., He, R., and Tan, T. A comprehensive survey on test-time adaptation under distribution shifts, 2023.
- Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., Gan, C., and Han, S. On-device training under 256kb memory, 2024.
- Mehta, S. and Rastegari, M. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer, 2022. URL <https://arxiv.org/abs/2110.02178>.
- Mollah, M. B., Azad, M. A. K., and Vasilakos, A. Security and privacy challenges in mobile cloud computing: Survey and way ahead. *Journal of Network and Computer Applications*, 84:38–54, 2017.
- Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. Efficient test-time model adaptation without forgetting. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/niu22a.html>.
- Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., and Tan, M. Towards stable test-time adaptation in dynamic wild world, 2023.
- Niu, S., Miao, C., Chen, G., Wu, P., and Zhao, P. Test-time model adaptation with only forward passes, 2024.
- Podolskiy, A., Lipin, D., Bout, A., Artemova, E., and Pi-ontkovskaya, I. Revisiting mahalanobis distance for transformer-based out-of-domain detection. *Proceedings of the AAAI conference on artificial intelligence*, 35(15): 13675–13682, 2021.
- Qayyum, A., Ijaz, A., Usama, M., Iqbal, W., Qadir, J., Elkhatib, Y., and Al-Fuqaha, A. Securing machine learning in the cloud: A systematic review of cloud machine learning security. *Front. Big Data*, 3:587139, November 2020.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.
- Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation, 2020. URL <https://arxiv.org/abs/2006.16971>.
- Shariat Panah, D., Hines, A., McKeever, J. A., and McKeever, S. An audio processing pipeline for acquiring diagnostic quality heart sounds via mobile phone. *Computers in Biology and Medicine*, 145:105415, 2022. ISSN 0010-4825. doi: <https://doi.org/10.1016/j.combiomed.2022.105415>. URL <https://www.sciencedirect.com/science/article/pii/S0010482522002074>.
- Stowell, D., Stylianou, Y., Wood, M., Pamuła, H., and Glotin, H. Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge.

- Methods in Ecology and Evolution*, 2018. URL <https://arxiv.org/abs/1807.05812>].
- Su, Z., Guo, J., Yao, K., Yang, X., Wang, Q., and Huang, K. Unraveling batch normalization for realistic test-time adaptation, 2024.
- Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.
- Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. URL <https://arxiv.org/abs/1905.11946>.
- Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization, 2021.
- Wang, Q., Fink, O., Gool, L. V., and Dai, D. Continual test-time domain adaptation, 2022.
- Xu, D., Xu, M., Wang, Q., Wang, S., Ma, Y., Huang, K., Huang, G., Jin, X., and Liu, X. Mandheling: Mixed-Precision On-Device DNN Training with DSP Offloading. In *Annual International Conference on Mobile Computing And Networking (MobiCom)*, 2022.
- Yuan, L., Xie, B., and Li, S. Robust test-time adaptation in dynamic scenarios, 2023. URL <https://arxiv.org/abs/2303.13899>.
- Yun, S. and Wong, A. Do all mobilenets quantize poorly? gaining insights into the effect of quantization on depthwise separable convolutional networks through the eyes of multi-scale distributional dynamics. *CoRR*, abs/2104.11849, 2021. URL <https://arxiv.org/abs/2104.11849>.
- Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017. URL <https://arxiv.org/abs/1707.01083>.
- Zhao, H., Liu, Y., Alahi, A., and Lin, T. On pitfalls of test-time adaptation, 2023. URL <https://arxiv.org/abs/2306.03536>.
- Zhou, K., Zhang, Y., Zang, Y., Yang, J., Loy, C. C., and Liu, Z. On-device domain generalization. *arXiv preprint arXiv:2209.07521*, 2022.
- Zuolkernan, I., Judas, J., Mahbub, T., Bhagwagar, A., and Chand, P. An aiOT system for bat species classification. In *2020 IEEE international conference on Internet of Things and Intelligence System (IoT&IS)*, pp. 155–160. IEEE, 2021.

# Supplementary Material

## LeanTTA: A Backpropagation-Free and Stateless Approach to Quantized Test-Time Adaptation on Edge Devices

---

### A. Background

#### A.1. Test-time Adaptation

Test-time adaptation (TTA), different from the related approaches of domain adaptation and generalization, is restricted to the realistic scenario where the model cannot access source data, nor the labels and potential distribution shifts of test data (Liang et al., 2023). Many TTA methods employ normalization layers, frequently present in deep learning models, to correct distribution shifts. During training, these layers record sample statistics — the expected value or mean of the input data ( $\mu_B$ ) and the variance of the input data ( $\sigma_B^2$ ) for each mini-batch  $B$ . A momentum term estimates the moving average across mini-batches. An affine transformation of the input is learned and applied after normalization with parameters  $\gamma$  and  $\beta$  (Ioffe & Szegedy, 2015):

$$y_i = \gamma \frac{x_i - \mu_B}{\sigma_B^2 + \epsilon} + \beta \quad (9)$$

Here,  $x_i$  represents the input,  $\epsilon$  is a constant added for numerical stability, and  $\gamma$  and  $\beta$  are learned scaling and shifting parameters, respectively.

Standard inference assumes that the recorded mean and variance will not change between training and test data. However, when statistics do shift, test-time adaptation methods like Tent (Wang et al., 2021) unfreeze and update normalization layer parameters, allowing adaptation to different distributions. While swapping sample statistics  $\mu_s$  and  $\sigma_s^2$  for per-batch running statistics is straightforward, updating learned parameters  $\gamma$  and  $\beta$  requires backpropagation. As there no labeled data is available, Tent proposes entropy minimization (Wang et al., 2021), which has become a common foundation for subsequent works.

#### A.2. Quantization

Increasingly, many TTA methods strive to be more efficient than baselines such as Tent (Niu et al., 2024; Hong et al., 2023; Khurana et al., 2021), but few have been designed to be both backpropagation-free and quantization-compatible.

Quantization is the compression of model parameters to low-precision, which speeds up computation and reduces memory usage during inference and training. On accelerators designed for high performance, lower memory costs lead to compounding improvements in latency and energy consumption, as energy costs may increase exponentially with each memory hierarchy level traversed (Sze et al., 2017). Layer fusion reduces adjacent convolution and batch normalization layers (as well as activation functions) to a single algebraically equivalent convolutional layer, which further reduces memory savings and decreases runtime.

One could use a system for low-memory quantized training (Lin et al., 2024) for either continual learning or TTA to adapt models to distribution shifts, but such methods may still consume more time and memory than quantized, inference-only adaptation, while achieving similar accuracies; additionally, these methods may not work for rare, discontinuous data.

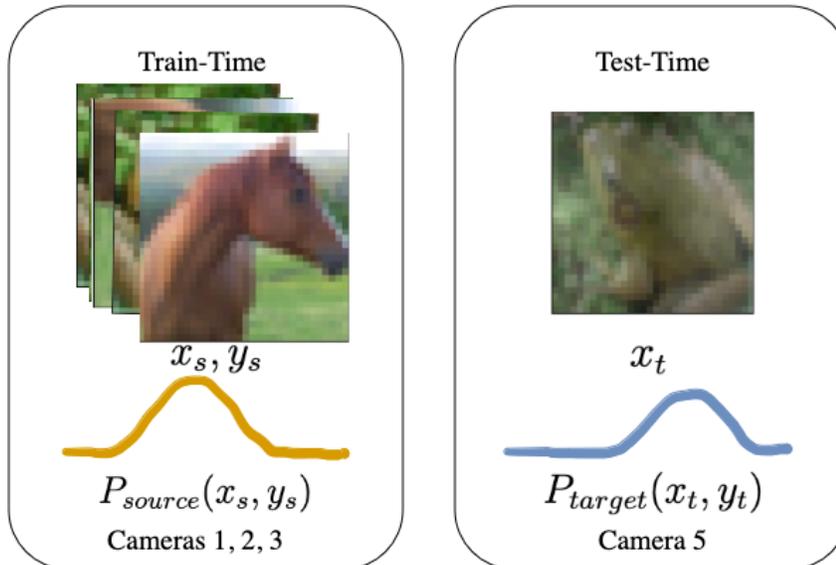


Figure 6. Illustration of an example TTA problem. Here, the data distribution learned by the model during training over  $x_s$ , in this example, gathered by 3 different cameras in different conditions, differs from the distribution of  $x_t$  encountered at test-time from a different camera in a different location.

## B. Detailed Experimental Setup

### B.1. Datasets

This section describes the LeanTTA benchmark tasks: CIFAR10-C, CIFAR100-C, and realistic, on-device applications with different modalities, specifically the audio datasets BirdVox-DCASE-20k and Warblr. These datasets were selected to represent common TTA application domains across various conditions.

For image classification, we use the CIFAR10-C and CIFAR100-C datasets (Hendrycks, 2019a;b), widely-used benchmarks for test-time adaptation. The CIFAR10/100-C datasets contain 19 distinct domains, each with 5 levels of severity. The images are 3-channel, with  $32 \times 32$  pixels each. The corrupted datasets are based on the 10- and 100-class CIFAR10 and CIFAR100 (Krizhevsky et al., 2009) datasets.

For a real-world test case without artificially induced domain shift, in an audio setting, we use the BirdVox-DCASE-20k (Stowell et al., 2018) and the Warblr (Stowell et al., 2018) datasets. BirdVox-DCASE-20k is an acoustic birdsong dataset collected from 6 locations around Ithaca, NY over the span of a single night in fall 2015; 50.09% of the 20,000 10-second samples contain birdsong. Warblr is a collection of 8,000 10-second samples collected by users around the United Kingdom using their personal smartphones, with 76% of the samples containing birdsong. The other 24% contain miscellaneous sounds, such as weather, human chatter, and traffic noise (Stowell et al., 2018). We simulate an experiment where a model is trained on BirdVox, then deployed on mobile devices to analyze Warblr. Warblr represents a distribution shift consisting of sounds not included in the training set, different audio sensors, and different locations. The two birdsong datasets were provided as part of the DCASE 2018 challenge (Stowell et al., 2018).

### B.2. Models

#### B.2.1. CIFAR MODELS

To rigorously evaluate our method on the CIFAR10 dataset, we employ two compact model architectures: MobilenetV2 and ResNet18. MobilenetV2 is noteworthy for its design tailored to mobile computing, offering an efficient balance of performance and resource usage (Yun & Wong, 2021). To investigate the impact of scaling the number of classes, we extend our experimentation by training on the more complex CIFAR100 dataset. We highlight the adaptability of our method across varying model architectures and underscore its robustness in handling datasets of increasing complexity.

### B.2.2. BIRD CALL DETECTION MODELS

For the bird call detection task, we used a VGGish architecture model based on the baseline architecture provided as part of the DCASE challenge (Kong et al., 2018). We trained the model on 16,000 samples from the BirdVox dataset with 4,000 samples held out for validation and quantization calibration. Training was carried out for 7,000 iterations, past the stabilization of the validation loss, and the best-performing model was selected. The inference task was to detect whether the sound recorded in a 10-second sample contained a real bird call, or other noises.

## B.3. Domain Shift Cases

### B.3.1. ABRUPT DOMAIN SHIFT

Non-continuous distribution shifts — such as those captured by photography, or recordings spaced out across time or distance, especially of rare events or subjects — are, from the perspective of the model conducting adaptation, abruptly changing domains. To simulate the real-world scenario of intermittent data collection throughout variable domains, we used two datasets: first, a benchmark dataset of CIFAR10-C and CIFAR100-C shuffled randomly across shift types and severity levels, and secondly, the Warblr dataset.

The Warblr dataset is an exemplar of the challenges of TTA: training on data gathered from a contained spatiotemporal area, using uniform hardware, then testing on data gathered under diverse conditions and sensors. Warblr is also challenging, as unlike the training BirdVox dataset, it is imbalanced — 75.6% of the audio samples collected by citizen scientists contain birdsong, while the other 24.4% contain other sounds. We simulate a similar setting for the image modality with CIFAR10/100-C, following the example of SAR (Niu et al., 2023) and RealisticTTA (Su et al., 2024) to sample and shuffle together 100 images from each of the 19 corruption categories and 5 severity levels. There is some label shift present in the CIFAR10/100-C datasets — compared to the training set, the sampling process unbalances the class datasets.

Quantized models were generated using the QNNPACK backend using PyTorch; post-static quantization scales and zero-points were recorded over 100 batches of size 64 from the CIFAR10/CIFAR100 training datasets. When necessary, layer fusion was done on adjacent convolutional and batch normalization layers.

### B.3.2. GRADUAL DOMAIN SHIFT

As opposed to the abruptly changing datasets above, gradually changing datasets illustrate a different use case. Here, data changes domain slowly, from severity level 1, to 5, then from 5 back down to 1, then to the next domain. We used the CIFAR10/100-C datasets to simulate this. Label shift is still present, as we otherwise used the same sampling technique as above.

## B.4. TTA Baselines

We compared our method to 4 state-of-the-art TTA benchmarks: Tent (Wang et al., 2021), CoTTA (Wang et al., 2022), EATA (Niu et al., 2022), and RealisticTTA (Su et al., 2024). In implementing the TTA methods, we followed the suggestions from their respective papers. For Tent, we used the Adam optimizer with a learning rate of 0.001, no weight decay, and betas of 0.9 and 0.999. For CoTTA, we used SGD with momentum at 0.9 as the optimizer and a learning rate of 0.001; in addition, we only optimized the batch normalization layers. The fisher values for EATA were calculated using 300 batch 64 data samples; TTA was conducted with SGD + momentum at 0.9. The  $e\_margin$  value was set to  $0.4 * \ln(numclasses)$ . RealisticTTA was implemented with a batch normalization momentum of 1.0; we selected a momentum value of 0 for abrupt and 0.99 for gradual shifts.

## B.5. Systems

Systems analysis was done on a Raspberry Pi Zero 2W, with 4 GB of swap memory available. All timings were conducted on the Raspberry Pi. Energy was approximated using a power draw recording device, whose measurements were sampled throughout and averaged, then multiplied by time to obtain power in watt-hours.

Table 5. Accuracy from various adaptation methods on the CIFAR10-C abruptly shifting dataset using only data from the highest try severity level, with the ResNet18 model. N=5. **Bold text** indicates highest performance per-column.

Method	Gaussian Noise	Shot Noise	Impulse Noise	Defocus Blur	Frosted Glass Blur	Motion Blur	Zoom Blur	Snow	Frost	Fog
None	13.91±1.08	17.34±1.44	17.02±0.54	53.04±1.84	42.8±1.16	62.02±0.91	59.33±1.1	72.66±1.74	58.39±1.68	67.48±1.33
None (int8)	13.61±0.36	16.77±0.51	17.88±0.56	52.44±0.76	43.49±1.13	58.59±0.67	59.76±1.61	72.1±1.55	58.89±0.85	63.71±1.67
RealisticTTA (1st 1k)	39.92±1.66	42.3±0.49	35.92±0.72	66.10±1.21	44.58±1.05	64.76±1.06	63.76±0.94	59.32±1.17	59.42±0.64	65.70±1.35
RealisticTTA (2nd 1k)	43.3±0.42	44.86±1.09	40.1±1.06	73.97±1.0	49.97±1.15	70.11±1.2	69.63±1.48	65.37±1.1	65.17±1.46	72.49±0.99
Tent (1)	9.97±0.61	11.13±0.44	10.07±0.85	9.75±0.89	9.97±0.64	9.71±1.16	9.97±0.42	10.25±0.7	9.51±0.77	9.91±0.82
Tent (16)	43.25±0.69	46.17±4.76	42.58±2.18	64.19±2.46	43.97±5.04	59.27±2.54	63.06±2.49	56.71±4.1	56.67±2.17	61.11±3.76
Tent (64)	<b>59.24±1.81</b>	58.73±1.37	53.55±1.7	80.12±1.46	56.95±2.27	75.53±1.47	79.26±1.98	74.86±1.4	72.01±1.9	78.87±0.73
EATA (1)	10.67±0.51	10.29±0.83	9.97±0.54	12.69±0.73	10.89±0.49	12.51±0.89	11.97±0.35	11.75±0.66	12.11±0.93	12.57±0.94
EATA (16)	47.38±6.0	46.33±0.85	41.63±4.02	72.66±2.08	47.5±3.92	69.86±1.8	72.38±1.87	66.13±1.61	62.1±0.6	70.69±2.88
EATA (64)	57.44±1.92	<b>60.06±0.77</b>	<b>54.37±1.02</b>	<b>84.49±1.37</b>	<b>62.46±1.11</b>	<b>81.05±0.44</b>	<b>82.71±0.35</b>	75.23±0.77	74.53±0.87	<b>82.85±1.06</b>
CoTTA (1)	9.31±0.88	9.97±0.76	10.37±0.86	12.15±1.27	10.61±0.28	10.77±0.88	10.95±0.96	10.89±0.55	10.27±0.15	11.35±0.63
CoTTA (16)	49.86±1.45	46.29±0.85	53.61±1.18	53.55±1.4	43.35±1.56	53.17±1.46	54.74±1.19	58.29±0.41	62.34±1.2	47.52±0.93
LeanTTA (Ours)	32.59±2.21	38.52±2.21	43.5±1.42	72.19±0.97	52.27±1.62	74.73±1.52	72.29±0.96	<b>77.94±0.98</b>	<b>74.73±1.65</b>	77.86±1.21
LeanTTA int8 (Ours)	30.61±1.11	37.42±1.66	42.24±0.61	68.67±1.16	50.11±1.18	71.33±0.85	70.21±1.32	77.8±1.16	72.43±1.64	76.82±0.63

Method	Brightness	Contrast	Elastic	Pixelate	JPEG Compression	Gaussian Blur	Saturate	Spatter	Speckle Noise	-
None	88.31±0.57	19.74±1.26	70.5±1.25	46.33±1.35	69.03±1.54	36.71±0.97	84.01±0.94	<b>79.46±0.77</b>	22.16±1.8	-
None (int8)	88.35±1.11	17.16±0.9	68.99±1.13	46.83±0.95	<b>70.44±1.29</b>	36.83±2.07	83.29±1.01	78.57±0.94	21.69±1.06	-
RealisticTTA (1st 1k)	68.38±0.56	62.90±0.98	54.08±1.66	56.96±1.21	46.46±1.68	65.06±1.27	68.90±1.40	59.74±1.13	42.42±1.06	-
RealisticTTA (2nd 1k)	76.36±1.38	71.39±1.2	59.02±0.93	62.28±1.04	51.77±0.65	72.53±1.01	75.96±1.13	64.9±2.31	46.89±1.5	-
Tent (1)	9.65±0.89	9.65±0.92	9.99±0.93	9.75±0.93	10.15±0.6	10.51±0.99	9.03±0.72	10.43±0.37	9.97±1.22	-
Tent (16)	66.69±3.21	59.54±3.27	52.26±2.27	56.79±3.64	50.44±1.76	62.56±5.3	69.98±2.22	62.66±2.91	46.85±3.82	-
Tent (64)	82.71±1.74	78.24±1.08	67.4±1.27	72.77±1.1	63.01±1.92	81.11±1.57	83.85±1.3	75.86±1.38	58.44±3.59	-
EATA (1)	12.07±0.83	12.51±0.23	11.79±0.55	12.39±1.52	11.19±0.52	12.19±0.93	12.85±0.14	12.35±0.82	9.79±0.48	-
EATA (16)	76.01±1.81	70.28±1.99	59.05±1.64	64.58±2.36	57.58±0.56	72.02±1.74	79.8±1.29	65.34±1.29	47.64±2.95	-
EATA (64)	87.07±1.94	<b>81.64±1.03</b>	<b>71.02±1.26</b>	<b>74.18±1.74</b>	66.8±0.86	<b>83.71±1.91</b>	86.29±0.76	76.86±0.55	<b>60.0±1.57</b>	-
CoTTA (1)	11.97±0.22	11.53±1.05	11.35±0.46	10.71±0.66	11.07±0.89	11.31±0.64	12.03±0.46	11.75±0.5	9.83±0.72	-
CoTTA (16)	68.91±1.44	50.95±0.98	48.21±1.44	52.76±1.34	54.05±0.97	53.37±1.07	62.66±1.67	58.97±2.18	43.13±1.01	-
LeanTTA (Ours)	88.75±0.73	47.45±1.89	68.47±1.86	55.12±1.35	64.56±0.93	60.02±0.82	<b>86.81±0.87</b>	78.68±1.34	45.13±0.67	-
LeanTTA int8 (Ours)	<b>89.39±1.02</b>	36.66±0.92	66.25±1.66	54.93±1.21	65.23±1.14	58.46±1.14	86.27±0.98	77.1±0.84	44.36±1.07	-

Model Precision	Resnet18	MobileNetV2
FP32 (no adaptation)	10.09±0.14	7.02±0.29
INT8 (no adaptation)	2.26±0.50	4.77± 0.09
LeanTTA (none fused)	8.81±0.17	29.70±0.41
LeanTTA (half fused)	4.84±0.07	13.06±0.16
LeanTTA (all fused)	1.82±1.10	3.55± 0.08

Table 6. Time-per-iteration in ms for 500 frames from the abruptly-changing dataset. FP32 and INT8 have no attached adaptation; convolutional and batch norm layers were fused for our method.

### C. Hyperparameter Exploration

An exploration of the effect of hyperparameter tuning indicated to us that, as shown in Figure 7, hyperparameter choice can impact the *degree to which adaptation improves accuracy* across different domain shifts and severity levels; however, we found that in keeping the hyperparameters set to 0.9, indicating a conservative balance between the source (90%) and target (10%) data, we achieved consistent improvement on baseline accuracies. Consistency is key, as it is difficult to anticipate the severity of the domain shift the model may encounter in the wild, when operating under limited data and computational resources.

### D. High-Severity Domain Shifts

The results presented in Table 5 illustrate the performance of various adaptation methods on the CIFAR10-C dataset, focusing on data from the highest severity level, using a ResNet18 model. The table highlights the effectiveness of different approaches in handling a range of corruptions, including Gaussian Noise, Shot Noise, and Impulse Noise, among others. Notably, our method, even with only one sample, demonstrates robust performance across several corruption types, achieving the highest accuracy in categories such as Snow, Frost, Brightness, and Saturate, with scores of 77.94%, 74.73%, 89.39%, and 86.81%, respectively. While some methods, such as EATA (which requires batch sizes of 64) and Tent (which also requires batch sizes of 64), achieve higher accuracies in certain noise categories, our method remains competitive in others. A significant advantage of our approach is its efficiency, as it only requires a batch size of 1, making it highly adaptable and resource-efficient compared to other methods that necessitate larger batch sizes for optimal performance. This efficiency

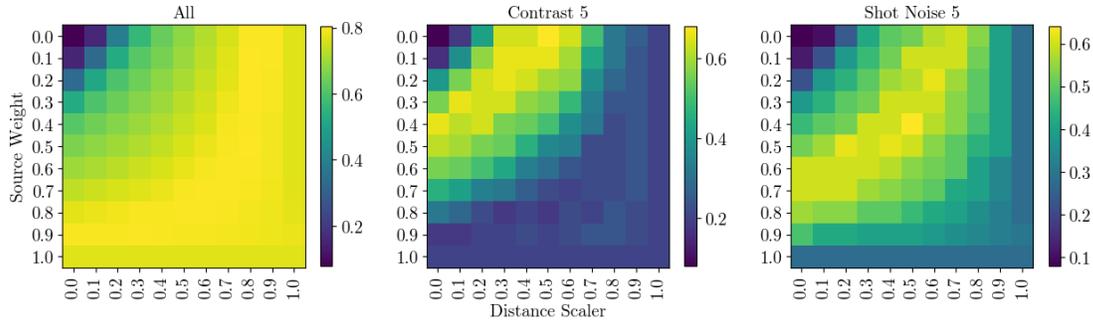


Figure 7. Hyperparameter space search for Source Weight  $\tau$  and Distance Scaler  $\lambda$  with ResNet18 applied to CIFAR10-C. From left to right, “All” shows accuracies across hyperparameter choices from all 19 categories and 5 severity levels, whereas “Contrast 5” and “Shot Noise 5” show accuracies from the corresponding noise type and severity. Lighter shaded squares indicate higher accuracy.

does not come at the cost of accuracy, as evidenced by our method’s strong performance across various corruption types. The results suggest that our method strikes an effective balance between computational efficiency and accuracy, making it a viable option for real-time applications where resources may be limited. Additionally, the int8 variant of our method maintains a commendable performance, further underscoring the adaptability and robustness of our approach in different computational settings. Overall, the results indicate that our method is a promising solution for adapting to severe dataset corruptions, providing a balance between high performance and low computational requirements.

## E. FBGEMM Timings

For the Raspberry Pi, which uses the ARM architecture, a QNNPACK backend is required; however, for other computing platforms, (FaceBook GEneral Matrix Multiplication) FBGEMM or x86 can be used as alternative quantization engines. In Table 6, we profiled the timings of our method with FBGEMM on a AMD Ryzen 7 3700X 8-Core Processor to demonstrate the terrific efficiency scaling of our system on other platforms.

We observed that quantization with the QNNPACK backend engine can slow down quantized inference with and without adaptation, especially for MobileNetV2, (Table 4). With the FBGEMM engine, we observed a significant speedup post-quantization; however, FBGEMM is incompatible with ARM architectures, like that of the Raspberry Pi line. We anticipate that future improvements to the QNNPACK implementation will lead to greater acceleration after quantization. The VGGish architecture did not suffer the same post-QNNPACK quantization slowdown, likely because it is a shallow model with very few layers relative to the deeper Resnet18 and MobileNetV2 architectures, which allow inefficiencies to compound.

Table 6 presents a comparative analysis of time-per-iteration in milliseconds for 500 frames from the abruptly changing dataset, evaluating the performance of ResNet18 and MobileNetV2 across different model precision and adaptation strategies. Our method of partial layer fusion alongside partial adaptation, provides significant improvements in TTA speed. Specifically, for ResNet18, our “none fused” approach reduces the time-per-iteration to 8.81 ms compared to the FP32 baseline of 10.09 ms, marking an improvement of approximately 12.7%. In the MobileNetV2 model, which has many more normalization layers (52) than ResNet18 (18), the “none fused” configuration achieves 29.70 ms; even with GPU acceleration, Tent with a batch size of one takes  $139.7 \pm 8.5$  ms to analyze one frame.

With partial fusion (“half fused”), ResNet18’s iteration time further drops to 4.84 ms, improving by 52.8% from the “none fused” setup. For MobileNetV2, the “half fused” configuration reduces the iteration time to 13.06 ms, enhancing speed by over 56%. The “all fused” configuration offers the most significant improvements, with ResNet18 achieving 1.82 ms, a 79.3% improvement over the “none fused” time, and MobileNetV2 reaching 3.55 ms, a 88% reduction from the “none fused” time. These enhancements underscore the substantial TTA efficiency gains achieved through our proposed partial layer fusion strategy.

## F. Layerwise Distances

To illustrate the way that the Mahalanobis distance  $d$  between source and target statistics changes across layers, we show  $d$  recorded at each layer during adaptation on subsets of CIFAR10/100-C, disaggregated by the severity of the distribution shift (1 being the least shifted, 5 being the most). From Figure 8, we can see that there are subtle changes between  $d$  at the different severities, indicating that  $d$  is able to detect changes in shift severity and dynamically adjust the balance between

Table 7. % Accuracy on abruptly changing CIFAR10-C datasets using the MobileViT-s architecture (32 BN layers) fine-tuned to the CIFAR10 dataset. N=5 trials. **Bold**: best, underline: 2nd best, *italics*: worse than baseline.

Method	Variant	Accuracy
None	Batch 1	72.0 ± 0.4
Ours	Full Adap.	73.8 ± 0.7
Ours	1st 8	<u>75.3 ± 0.3</u>
Ours	1st 16	<b>75.6 ± 0.3</b>
Ours	1st 24	74.8 ± 0.5
Tent	Batch 64	75.2 ± 0.4
Tent	Batch 16	<i>69.8 ± 1.0</i>
EATA	Batch 64	74.2 ± 0.2
EATA	Batch 16	<i>71.1 ± 0.7</i>
CoTTA	Batch 64	<i>60.9 ± 1.0</i>
CoTTA	Batch 16	<i>57.7 ± 0.5</i>
RealisticTTA	Batch 2	<i>15.5 ± 0.4</i>

source and target statistics in our adaptation module.

The pattern of these curves are specific to their model architectures, but not the dataset. We see two distinct patterns. For the Resnet18 distances, at layers after 12 or 13, low-severity shifts begin to produce greater divergences between source and target statistics, and the distances begin to explode. For MobileNetV2, we see oscillatory behavior of  $d$ , which is a possible area of further investigation but is likely specific to the architecture of the model. There are two peaks of high distance in the MobileNetV2 architecture. For future works, areas of high distance (and thus, implied high variance) may allow us to further optimize which layers we fuse, versus which layers we add adaptation to, potentially leading to accuracy gains.

## G. MobileViT Experiments

MobileViT (Mehta & Rastegari, 2022) is a transformer-convolution hybrid, which uses “transformers as convolutions”. It has been adopted as a state-of-the-art model which combines the representational power of local spatial inductive biases (from convolutions) and the global structural understanding of transformers. The hybrid architecture makes it a good candidate for on-device deployments, as the inclusion of inductive biases improves accuracy over similarly sized transformer models (Mehta & Rastegari, 2022). We applied LeanTTA to the batch normalization layers of a full-precision MobileViT-small in order to understand its performance in interaction with transformers and layerwise normalization. The model was trained on ImageNet and fine-tuned with CIFAR10’s training set, then tested on an abruptly shifting dataset sampled from CIFAR10-C, as described in our experimental methods.

As shown in Table 7, our method, especially when using partial adaptation over the shallowest half of the model (1st 8, 1st 16), outperforms all other methods, even at large batch sizes. We did not include batch 1 analyses because the nature of the normalization layers used in the MobileViT-s did not allow for training with batch size set to 1. Table 7 demonstrates three things: the generalizability of our method, even to architectures with transformer and layer normalization layers; the potential importance of partial layerwise adaptation, supporting our findings from Figure 4; and the breakdown of other methods under certain data conditions.

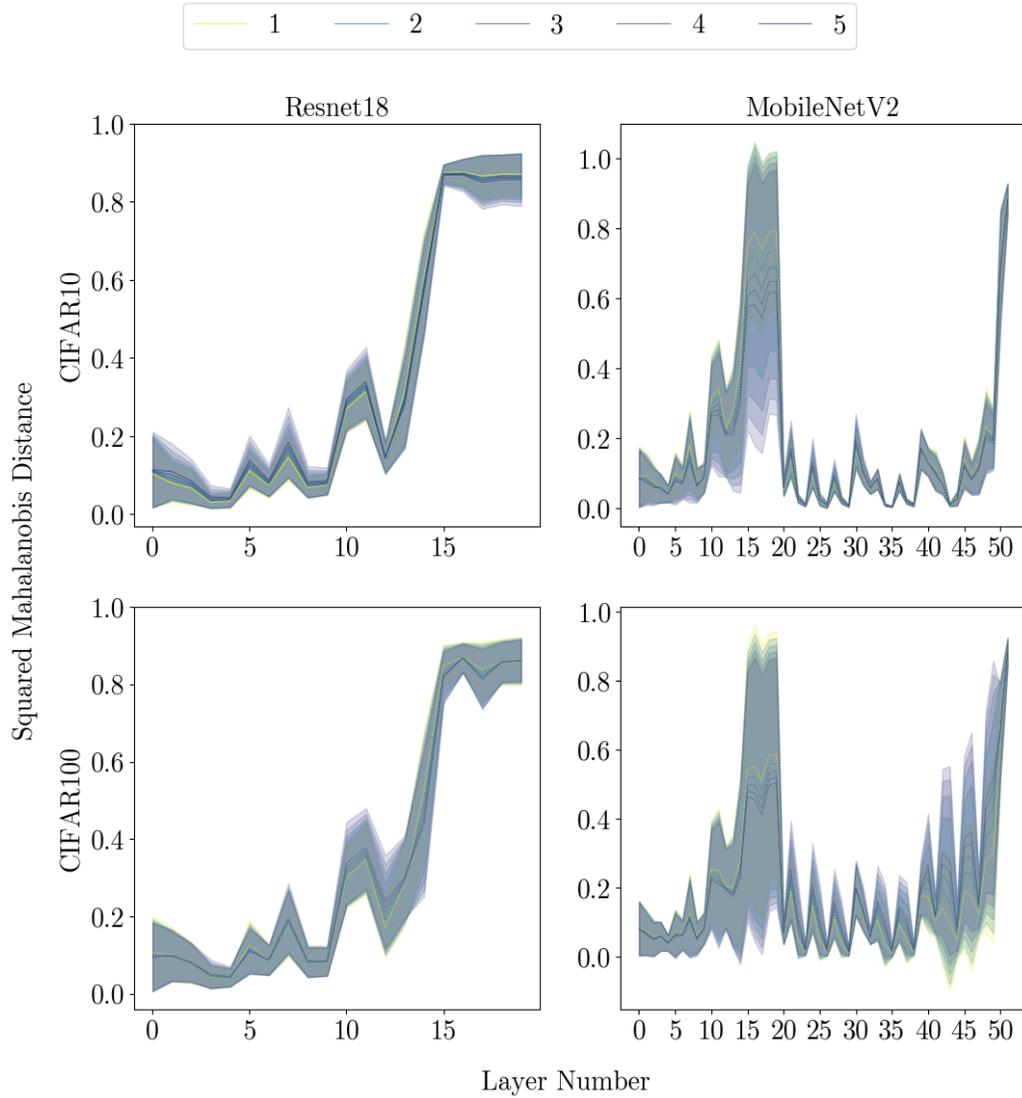


Figure 8. Layerwise Mahalanobis distance  $d$  calculated after stabilization in Equation 3.2.2, separated by severity of distribution shift in the CIFAR10/100-C datasets (light being the least severe, dark being the most). Shading indicates one standard deviation on either side of the average over  $N=5$  trials.