

A Appendix

A.1 Implementation Details

In this work, we compare our proposed approach against several state-of-the-art test-time adaptation (TTA) baselines. For fair comparison, we carefully reproduce each method based on official implementations and papers, and unify the training environment as much as possible. Below, we detail the optimization and hyperparameter settings used for each method.

All experiment including **TENT**, **DeYo**, **SAR**, **CMF**, and **ROID** are optimized using the Adam optimizer with a learning rate of $1e-3$, $\beta = 0.9$, and zero weight decay.

EATA shares the same optimizer configuration as **TENT** (Adam, LR= $1e-3$, $\beta = 0.9$, WD=0.). Additionally, we set the Fisher regularization strength to 1.0 and the confidence margin d to 0.4, following the original implementation. For the source distribution sampling, we use 2000 samples to compute the Fisher information matrix.

These unified settings ensure that performance differences primarily arise from the intrinsic mechanisms of each method, rather than discrepancies in optimization or tuning.

All experiments on **CIFAR10-C**, **CIFAR100-C**, and **ImageNet-C** [1] are conducted under **Severity 5** settings, following standard protocol to evaluate robustness under the highest level of corruption.

CIFAR10W [10] is a web-collected dataset constructed to evaluate model robustness under realistic distribution shifts. It is composed of three distinct subsets—**DF (Diffusion)**, **KW (Keyword)**, and **KWC (Keyword with Cartoon)**—each reflecting different data generation strategies and semantic characteristics. To comprehensively evaluate the generalization ability of each adaptation method, we conduct separate experiments on all three subsets.

All experiments are performed using NVIDIA RTX A6000 GPUs.

For the experiment in **Section 4.2.6 (Continuously Changing Domains)**, we use a **batch size of 16** during adaptation, which balances stability and responsiveness to gradual domain shifts.

For the analysis in **Section 4.3.2 (Effect of α)**, we intentionally deviate from the configuration presented in Section 4.3.1. Specifically, we insert the Buffer Layer *only after a very first single activation layer*, rather than applying it throughout the network. This simplified setting isolates the effect of the scaling parameter α , allowing for a more controlled analysis of its influence on adaptation performance.

All Buffer Layers are composed of **randomly initialized convolutional modules, without any pretraining**. They are optimized solely during test time, reinforcing the simplicity and modularity of the proposed approach.

Following the common practice in test-time adaptation, we configure BatchNorm(BN) layers to use target-domain batch statistics for normalization, while keeping the affine parameters frozen. This allows the model to respond to distributional shifts in the input without altering any trainable components of the normalization layers.

A.2 Pseudo-code of Implementing Buffer Adaptation

Algorithm 1 Test-Time Adaptation with Buffer Layer

- 1: **Input:** test sample \mathbf{x} , pretrained model \mathcal{F}_θ , buffer module \mathcal{B}_ϕ , TTA algorithm \mathcal{A}_ψ
 - 2: **Output:** adapted prediction $\hat{\mathbf{y}}$
 - 3: Initialize adaptation method \mathcal{A}_ψ
 - 4: Attach buffer module \mathcal{B}_ϕ in parallel to \mathcal{F}_θ
 - 5: Freeze all parameters in \mathcal{F}_θ
 - 6: Enable gradients for ϕ
 - 7: $\hat{\mathbf{y}} \leftarrow \mathcal{F}_\theta(\mathbf{x}) + \mathcal{B}_\phi(\mathbf{x})$
 - 8: $\mathcal{L} \leftarrow \mathcal{A}_\psi(\hat{\mathbf{y}})$
 - 9: Update ϕ using $\nabla_\phi \mathcal{L}$
 - 10: **return** $\hat{\mathbf{y}}$
-

The adaptation procedure with the proposed Buffer Layer is intentionally designed to be simple and modular, as illustrated in Algorithm 1. Given any existing TTA method, the only modification required is to attach an external Buffer layer in parallel to the pretrained model and enable gradient updates for the buffer’s parameters. The backbone network remains entirely frozen, ensuring that the source-domain representations are preserved, while the Buffer Layer acts as a residual path that provides localized feature-level corrections during adaptation. This design not only minimizes implementation overhead but also ensures broad compatibility across different architectures and optimization schemes. As a result, our method can be easily incorporated into existing TTA pipelines with minimal code changes, without disrupting the original model structure or its pretrained functionality.

A.3 Mixed Domains

Table 1: CIFAR100-C under mixed domain shifts.

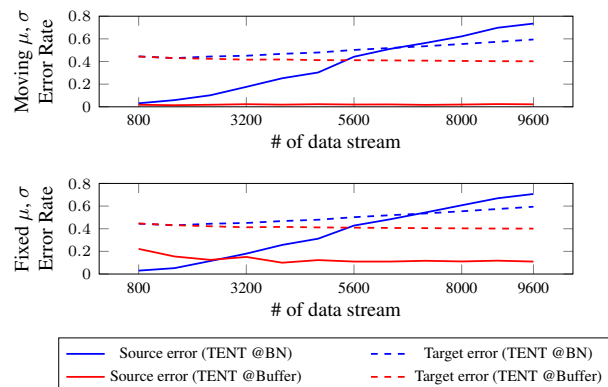
Dataset Models	ImageNet-C	
	BS	2
TENT [11]	@BN	98.92
	@Buffer	98.63
EATA [6]	@BN	76.53
	@Buffer	76.58
CMF [2]	@BN	98.95
	@Buffer	83.41
DeYo [3]	@BN	96.28
	@Buffer	76.65
SAR [7]	@BN	76.42
	@Buffer	76.42
ROID [5]	@BN	95.76
	@Buffer	76.90

maintains low target-domain error while preserving source-domain accuracy, even in the absence of domain boundaries or resets. These results highlight the adaptability and stability of our approach in dynamic TTA settings, reinforcing its practical utility for real-world applications where domain shifts occur unpredictably and continuously.

A.4 Further Experiment on Catastrophic Forgetting

As discussed in the main text, applying existing TTA approaches such as Tent to BN layers can lead to source domain forgetting, a phenomenon that becomes significantly more pronounced under relatively small batch sizes, often resulting in increased target domain error. In contrast, the proposed Buffer Layer exhibits strong resistance to such forgetting, maintaining source performance while enabling stable adaptation to the target domain. Tab. 2 presents results on the CIFAR100-C dataset, further confirming that this robustness generalizes beyond the settings reported in the main experiments.

Table 2: Source and target errors on CIFAR100-C dataset, ResNeXT, batch size 16, Gaussian Noise.



While catastrophic forgetting has been recognized as a critical challenge in TTA, prior studies have paid limited attention to establishing a standardized and fair evaluation protocol for measuring it. In particular, the question of how best to evaluate a model’s retention of source-domain performance after adaptation remains largely unaddressed in the existing literature [6]. This leaves open an important methodological consideration—when assessing forgetting, should the adapted model be evaluated on source-domain data using updated source statistics (moving μ, σ)? Or should the model

92 instead be evaluated using the target-domain statistics fixed during adaptation, thereby preserving the
 93 post-adaptation state (fixed μ, σ)?

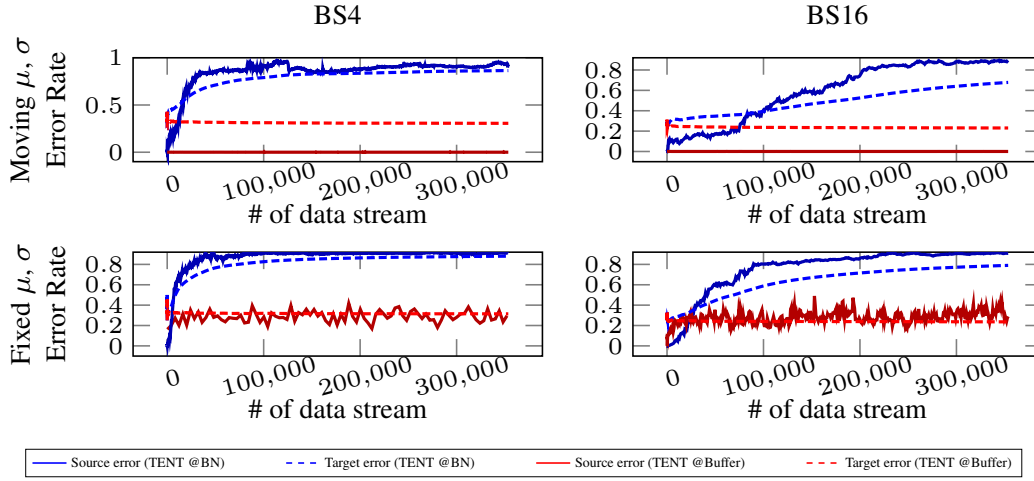


Figure 1: Catastrophic forgetting experiments on WRN28, CIFAR10-W (KW). Blue: TENT @BN, Red: TENT @Buffer.

94 To clarify this methodological ambiguity, we conduct experiments under both evaluation protocols.
 95 By comparing both settings, we are able to more precisely assess the impact of the Buffer Layer on
 96 preserving source-domain performance, particularly under small batch size conditions. While both
 97 protocols yield complementary insights, we consider **the use of source-domain statistics (moving**
 98 **μ, σ)** to be more practical and informative, as it reflects realistic deployment scenarios in which
 99 source-aligned calibration may still be available or preferable. Importantly, these moving statistics
 100 are collected during the standard forward pass over unlabeled source-domain data, and do not require
 101 any additional supervision or training overhead. From this perspective, using moving μ and σ is not
 102 only cost-free but also fully consistent with test-time constraints. Accordingly, we adopt this protocol
 103 for the results presented in the main paper. As shown in Tab. 2 and Fig. 1, our method demonstrates
 104 consistently strong anti-forgetting performance across CIFAR100-C and CIFAR10-W, supporting the
 105 efficacy of the Buffer Layer in mitigating forgetting even under challenging conditions. This effect is
 106 also clearly reflected in the source-domain results shown in Fig. 2.

107 This finding highlights a clear deviation from the commonly observed trade-off in TTA, wherein
 108 performance improvements on the target domain are typically accompanied by degradation on the
 109 source. The Buffer Layer, however, achieves simultaneous gains in both domains, indicating a more
 110 favorable balance between adaptation and retention.

111 A central factor behind this effect is the non-intrusive and parallel architecture of the Buffer Layer.
 112 Unlike conventional methods that adapt BN layers by updating their affine parameters, our approach
 113 avoids modifying any pretrained, learnable parameters in the backbone. In many existing methods,
 114 such updates overwrite source-domain representations, and once altered, the original alignment to the
 115 source distribution becomes difficult to recover. In contrast, the Buffer Layer operates as a structurally
 116 independent residual branch that leverages only the input’s batch statistics—mean and variance—for
 117 adaptation. Notably, this auxiliary layer can be interpreted as implicitly fulfilling the role of affine
 118 transformation in a parallel and externalized manner, enabling domain-specific modulation without
 119 interfering with the main pathway. As a result, it preserves the integrity of source-domain features
 120 while allowing effective target-domain adaptation, thereby offering enhanced robustness against
 121 catastrophic forgetting.

122 By maintaining a strictly parallel configuration, the Buffer Layer allows source-domain inputs to be
 123 processed exclusively through the unmodified backbone, entirely bypassing adaptation-specific paths.
 124 This architectural decoupling provides a compelling explanation for the observed reductions in both
 125 source and target domain errors—achieved without violating the trade-off constraints that typically
 126 characterize TTA.

127 A.5 Feature-level Analysis

128 To further validate the effectiveness of the proposed Buffer layer, we conducted an analysis at
 129 the feature level by examining how different adaptation strategies affect internal representations.
 130 Following the methodology of [11], we visualized the distributional statistics—specifically, the
 131 channel-wise mean and variance—of intermediate features to observe the model’s response to
 132 domain shift. For this purpose, we randomly selected a subset of feature maps from the output of
 133 stage2 (corresponding to the 18th layer in ResNeXT), which captures mid-level semantics critical to
 134 downstream predictions.

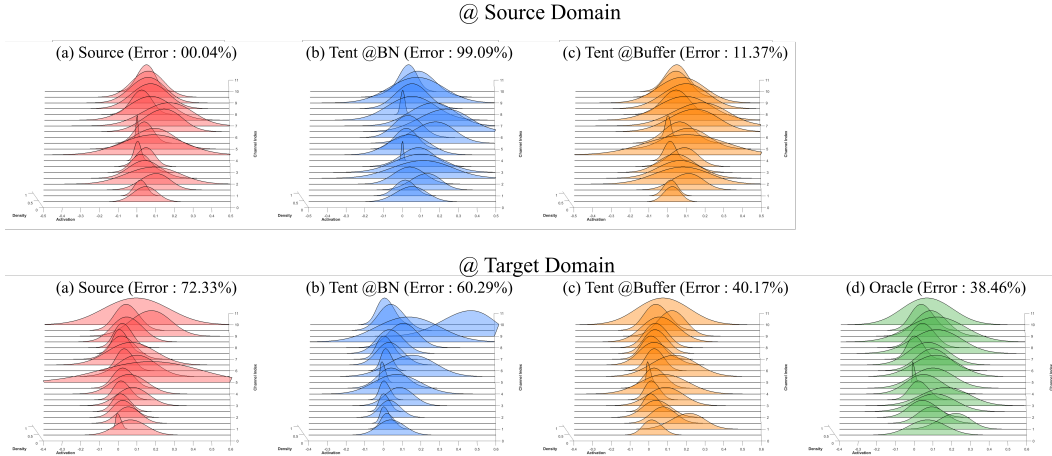


Figure 2: Feature distribution comparison across adaptation strategies on CIFAR100-C (batch size = 16). Despite being applied only at the early stage of the network, Buffer-based adaptation (orange) yields feature statistics (mean and variance) that are more closely aligned with the Oracle model (green), compared to TENT applied at BN layers (blue). This suggests that the Buffer layer effectively propagates adaptation signals throughout the network, enabling target-aware representations even in deep feature spaces.

135 We compared four adaptation settings: (a) Source (no adaptation), (b) TENT applied to BN layers
 136 (TENT@BN), (c) TENT applied to the Buffer layer (TENT@Buffer), and (d) Oracle, which is trained
 137 with full access to target-domain labels using cross-entropy loss, following the setup in [11]. All
 138 experiments were conducted on the CIFAR100-C dataset with a batch size of 16. The visualizations
 139 indicate that the feature distributions obtained from TENT@Buffer are more closely aligned with
 140 those of the Oracle model than those from TENT@BN.

141 This result suggests that the Buffer layer enables a more effective form of test-time adaptation by
 142 producing internal representations that better reflect the target-domain characteristics. The alignment
 143 with Oracle-level feature statistics provides further support for the Buffer layer’s capacity to generalize
 144 under domain shift without requiring access to target supervision.

145 A.6 Why the Buffer Layer Works Well?

146 Unlike conventional TTA methods that rely on modifying internal components of the pretrained
 147 model, our approach introduces adaptation externally—through a structurally independent buffer
 148 layer. This design choice is not merely architectural; it fundamentally alters how adaptation interacts
 149 with the existing representation space. By avoiding direct updates to the backbone, the buffer layer
 150 prevents destructive interference with source-domain features, a common cause of catastrophic
 151 forgetting in BN-based adaptation. Instead, the pretrained backbone remains intact, serving as a
 152 stable foundation throughout the adaptation process.

153 More importantly, this separation enables a distinct mode of representation learning. Instead of
 154 altering or overwriting existing features, the Buffer Layer introduces complementary target-specific
 155 activations that coexist with the pretrained representations. This leads to an effective expansion of the
 156 class-conditional feature space, as the model learns to associate semantic concepts with a broader
 157 range of domain-specific variations. Such behavior is conceptually aligned with findings in multi-

domain and domain generalization literature, where diverse exposure to distributional shifts—without altering label semantics—has been shown to improve generalization [4]. In this light, the Buffer Layer can be interpreted as enabling a form of adaptation-as-augmentation: it allows the model to incorporate new domain signals without sacrificing previously acquired knowledge.

In contrast to recent additive-layer approaches [9, 8] that inject adaptation modules directly into intermediate layers of the backbone, often modifying the main information flow, the Buffer Layer remains fully external and modular. While such additive methods may retain partial structural separation, they still introduce parameter updates or architectural interference that can compromise source-domain representations. The Buffer Layer, by comparison, performs domain-specific modulation purely through residual pathways and without altering any pretrained parameters, effectively emulating the role of affine adaptation in a parallel, non-destructive manner.

Overall, the Buffer Layer enables a form of adaptation that sidesteps the destructive interference commonly observed in BN-based or entangled additive-layer methods. By maintaining a clean separation between the adaptation mechanism and the pretrained model, it preserves the model’s original capabilities while selectively enhancing its responsiveness to target-domain signals. This balance between stability and adaptability offers a scalable and reliable foundation for test-time deployment in dynamic or continuously shifting environments.

References

- [1] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.
- [2] Jae-Hong Lee and Joon-Hyuk Chang. Continual momentum filtering on parameter space for online test-time adaptation. In *The Twelfth International Conference on Learning Representations*, 2024.
- [3] 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. *arXiv preprint arXiv:2403.07366*, 2024.
- [4] Seunghun Lee, Sunghyun Cho, and Sunghoon Im. Dranet: Disentangling representation and adaptation networks for unsupervised cross-domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15252–15261, 2021.
- [5] Robert A Marsden, Mario Döbler, and Bin Yang. Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2555–2565, 2024.
- [6] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022.
- [7] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.
- [8] Jin Shin and Hyun Kim. L-tta: Lightweight test-time adaptation using a versatile stem layer. *Advances in Neural Information Processing Systems*, 37:39325–39349, 2024.
- [9] 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*, pages 11920–11929, 2023.
- [10] Xiaoxiao Sun, Xingjian Leng, Zijian Wang, Yang Yang, Zi Huang, and Liang Zheng. Cifar-10-warehouse: Broad and more realistic testbeds in model generalization analysis. *arXiv preprint arXiv:2310.04414*, 2023.
- [11] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.
- [12] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023.