RESEARCH-ARTICLE

EBaR: Efficient Buffer and Resetting for Single-Sample Continual Test-Time Adaptation

TIANYI MA, University of Technology Sydney, Sydney, NSW, Australia

MAOYING QIAO, University of Technology Sydney, Sydney, NSW, Australia

Open Access Support provided by:

University of Technology Sydney

Published: 27 October 2025

Citation in BibTeX format

MM '25: The 33rd ACM International Conference on Multimedia
October 27 - 31, 2025
Dublin, Ireland

Conference Sponsors:
SIGMM

https://doi.org/10.1145/3746027.3754907
ISBN: 9798400720352

EBaR: Efficient Buffer and Resetting for Single-Sample Continual Test-Time Adaptation

Tianyi Ma
tianyi.ma@student.uts.edu.au
University of Technology Sydney
Sydney, NSW, Australia

Maoying Qiao
maoying.qiao@uts.edu.au
University of Technology Sydney
Sydney, NSW, Australia

Abstract

In test-time adaptation, handling constant domain change using a single sample at a time presents two key challenges: efficiently stabilizing adaptation and effectively preventing catastrophic forgetting. This paper introduces a single-sample continual test-time adaptation (S-CoTTA) task to address these challenges. Existing works mainly either 1) apply continual test-time adaptation methods with an inefficient moving window that increases memory overhead or 2) filter out high-uncertainty samples to maintain stability. The former handled forgetting but failed to stabilize adaptation efficiently, while the latter neglected the catastrophic forgetting issues. We argue that both efficient tuning stabilization and forgetting prevention should be addressed simultaneously. To this end, we proposed a novel **E**fficient **B**uffer **a**nd **R**esetting (EBaR) method for S-CoTTA. EBaR employs a novel memory-efficient buffer to store samples based on their uncertainty levels and utilizes them to update the model with different losses to enhance stability. EBaR also incorporates a novel elastic resetting unit to dynamically reset the parameters based on their sensitivity to domain shift. The elastic resetting strategy effectively mitigates catastrophic forgetting while retaining useful target domain knowledge. Comprehensive experimental evaluations demonstrate the effectiveness and efficiency of both components. Combining their benefits, EBaR surpasses state-of-the-art methods across multiple datasets, including CIFAR10-C, CIFAR100-C, ImageNet-C, and CCC, for the S-CoTTA task.

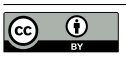
CCS Concepts

• **Computing methodologies** → **Learning settings**.

Keywords

Domain Adaptation, Test-time Adaptation, Continual Learning, Single-sample Adaptation

ACM Reference Format:
Tianyi Ma and Maoying Qiao. 2025. EBaR: Efficient Buffer and Resetting for Single-Sample Continual Test-Time Adaptation. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3754907



MM '25, Dublin, Ireland

ACM ISBN 979-8-4007-2035-2/2025/10
https://doi.org/10.1145/3746027.3754907

1 Introduction

Standard test-time adaptation (TTA) mitigates the domain gap between a source and a target domain [4, 14, 17–19, 22, 33, 45, 48, 55, 57]. Source-free TTA adapts a model with unlabelled target data only [22], distant from supervised [11, 21, 28] or semi-supervised [2, 23, 54, 56] adaptation. A challenging scenario for source-free TTA is single-sample TTA (STTA) [1, 7, 10, 29, 30, 40, 48, 57], where only one target data is available at a time. Since target data can be highly noisy, the model can easily misestimate the distribution of the target domain. Such misestimation leads to severe parameter perturbation that increases the instability of adaptation[33, 40]. Meanwhile, a model may also encounter constant change in target domains. For instance, different weather and road conditions constantly affect an obstacle detection model in autonomous driving. As a result, continual test-time adaptation (CoTTA) [24, 25, 29, 41, 46] is introduced. In CoTTA, a model is adapted to dynamic environments over a long time, making it difficult for the model to remember the source knowledge and causing catastrophic forgetting. When applying a model in the real world, STTA and CoTTA can happen jointly. In such circumstances, a model is adapted to constantly changing target domains with a single sample per time for a long time[48], posing two challenges: 1) efficient and stable adaptation with less memory overhead and timely prediction and 2) effective prevention of catastrophic forgetting.

To systematically address the aforementioned challenges, we introduce a **S**ingle-sample **Co**ntinual **T**est-**T**ime **A**daptation (S-CoTTA) task, as illustrated in Fig.1. In S-CoTTA, a pre-trained model is adapted to target domains that change continually over time using one unlabelled target data at a time. The primary objectives for S-CoTTA are twofold: 1) **Adaptation Stabilization**: Efficiently stabilizing the adaptation of the model using a single target sample at a time, and 2) **Forgetting Prevention**: Effectively remembering the source knowledge over a long time. Recent works primarily address one of them. For instance, some apply a moving window or memory bank to a CoTTA method [29, 41, 46]. Although forgetting is mitigated, these methods failed to stabilize adaptation efficiently, and a large memory overhead is required. Others stabilize adaptation by filtering out high-uncertainty samples [40, 51]. Although stabilizing the adaptation, these methods fail to prevent forgetting. We contend that achieving both efficient adaptation stabilizing and effective prevention of forgetting is essential.

To this end, we propose a novel **E**fficient **B**uffer **a**nd **R**esetting (EBaR) method to address the challenges posed by S-CoTTA. As illustrated in Fig.1 and Fig.4, EBaR consists of two key components: an efficient buffer and an elastic resetting control unit.

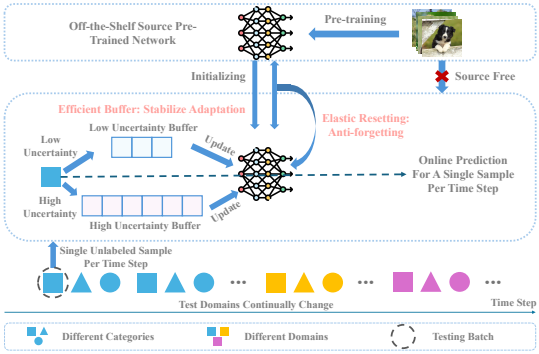


Figure 1: The concept of single-sample continual test-time adaptation (S-CoTTA) and our solution Efficient Buffer and Resetting (EBaR). In S-CoTTA, a model is adapted to target domains that constantly change over time with only one sample per time step. Under such settings, a model usually suffers from 1) unstable adaptation and 2) catastrophic forgetting. To address this, we propose EBaR, which includes an efficient buffer and an elastic resetting unit. The efficient buffer consists of a low and a high uncertainty buffer that stores low-uncertainty and high-uncertainty samples, respectively. The samples stored in two buffers update the model with different losses to stabilize tuning. The elastic resetting resets each parameter differently based on their sensitivity to domain shift, preventing forgetting while retaining necessary target domain knowledge.

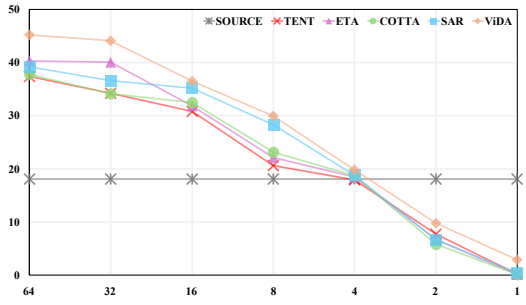


Figure 2: The classification accuracy (%) of the CoTTA methods under different batch size settings with ResNet-50 [43, 44] on ImageNet-C. The accuracy declines drastically following the decrease in batch size because of the tuning instability.

First, EBaR incorporates a novel memory-efficient buffer, named efficient buffer (EB), to stabilize the adaptation progress with less extra memory overhead. EB consists of two buffers: a low uncertainty buffer (LUB) and a high uncertainty buffer (HUB). During adaptation, our model first predicts the category of a sample directly. Then, a symmetric entropy weight is calculated to indicate the uncertainty of the sample [29, 33], and the sample is categorized into high- and low-uncertainty levels with a predefined threshold. Samples with low uncertainty are stored within the LUB of a smaller capacity; others are stored within the HUB of a larger capacity. When full, buffers release samples to update the model. A soft likelihood ratio and a symmetric entropy loss [7] are calculated for LUB samples, while a weighted contrastive learning loss is computed for HUB samples. EB brings two benefits. First, it congregates samples and applies different losses according to sample uncertainty to stabilize adaptation. Second, compared with moving windows and other memory banks, EB is smaller in size. Thus, it requires less memory overhead and allows timely updates, achieving higher efficiency.

Second, inspired by [36], EBaR incorporates a novel elastic resetting (ER) control unit to prevent catastrophic forgetting. ER adopts a reset clock and a dynamic weight reset for each parameter. At each time step, the reset clock increases by 1) a fixed amount plus 2) a value proportional to the updating magnitude of the parameter. When the reset clock exceeds a threshold, the dynamic weight reset is applied to the corresponding parameter based on its sensitivity to domain shift. Then, the reset clock is set back to zero. Elastic resetting ensures that parameters that are more susceptible to domain shift are reset more frequently and closer to the initial value. ER has two benefits. First, it effectively mitigates catastrophic forgetting. Second, compared to [36], where a whole model is reset after a fixed number of time steps, our strategy allows the model to retain target knowledge by resetting insensitive parameters less frequently.

Finally, comprehensive experiments are conducted to evaluate the effectiveness of EBaR. We demonstrate that in EBaR: 1) the efficient buffer significantly improves classification accuracy with reduced extra memory overhead compared to other counterparts like the moving window, and 2) the elastic resetting enables the model to achieve higher classification accuracy stably over a long period of time. Combining the benefits of the two key components, EBaR achieves state-of-the-art performance across multiple corruption datasets, including CIFAR10-C, CIFAR100-C ImageNet-C, and CCC under the S-CoTTA setting with less extra memory overhead.

Our contributions can be summarized as follows:

- We introduce a challenging **S**ingle-sample **Co**ntinual **T**est-**T**ime **A**daptation (S-CoTTA) task. In S-CoTTA, a model is adapted to target domains that change constantly over time using a single unlabelled sample per time step. The objective is to achieve stable, efficient adaptation with one sample at a time and prevent catastrophic forgetting over a long time.
- We proposed a novel **E**fficient **B**uffer **a**nd **R**esetting (EBaR) method for S-CoTTA. To achieve stable adaptation, we introduce a novel memory-efficient buffer named efficient buffer, consisting of a smaller and a larger buffer to store samples with low and high uncertainty levels, respectively. The stored samples then update the model with different losses. To prevent catastrophic forgetting, we introduce a novel elastic resetting control unit to reset each parameter differently based on its sensitivities to domain shifts.
- We conduct comprehensive experiments to evaluate the effectiveness of the efficient buffer and the elastic resetting in EBaR. The results demonstrate the effectiveness of both components.

2 Related Works

2.1 Continual Test-time Adaptation

Continual test-time adaptation (CoTTA) [46] is proposed to deal with constant domain changes during online inference, which is more challenging than standard test-time adaptation, where a single test domain is presented [12, 22, 31, 48, 55, 58]. Recent works primarily focus on mitigating catastrophic forgetting. Catastrophic forgetting is when a model forgets the source knowledge due to noisy pseudo-labels in self-supervised adaptation over time. For instance, in [7, 29, 46], a teacher model trained with exponential

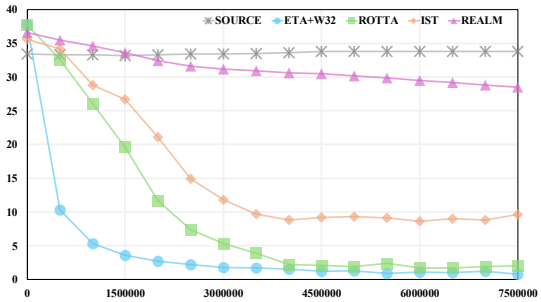


Figure 3: The classification accuracy (%) of the STTA methods under the S-CoTTA setting on the CCC dataset with ResNet-50 baseline. We apply a moving window with size 32 for ETA, denoted as 'ETA+W32'. The accuracy declines following the increase of time step because of catastrophic forgetting.

updates is used to preserve the source knowledge. In [33], an anti-forgetting regulation is introduced to retain the closeness of the adapted model to the source model for anti-forgetting. In [42], a trainable adapter is incorporated in the normalization layer to adapt a model without interfering with the source model. In [3], prototypes generated by the source model are adopted to guide adaptation. In [9], visual prompts are applied to absorb domain-agnostic and domain-specific knowledge separately. In [25], low- and high-rank anti-forgetting adapters are proposed to disentangle domain-shared and domain-specific knowledge from the model. In [32, 36], a simple resetting strategy is explored to restore the source knowledge. In addition to addressing forgetting, other works focus on extracting high-quality target domain knowledge. For instance, [47] reweights samples according to their noise levels to extract unbiased target knowledge. In [12, 51], class-balanced memory banks are proposed to address the class imbalance problem during the test time. In [24], an auto-encoder with a distribution mask is proposed in a vision transformer [8] to facilitate the extraction of target knowledge. However, [48] find that when using a vision transformer backbone, recent CoTTA methods tend to collapse as the test batch size decreases. In Fig.2, we show that this collapse is even more severe for CNN baselines [13]. Therefore, existing CoTTA methods require further improvement to fit the S-CoTTA task, where batch size is only one.

2.2 Single-sample Test-time Adaptation

Single-sample test-time adaptation (STTA) addresses test-time adaptation with only one sample provided at a time. Progress has been made with respect to STTA in image segmentation [37, 55], detection [27], and classification tasks [7, 27, 33, 34, 40]. In this work, we focus on the classification task. Several recent works [7, 33, 34] apply a moving window, a large buffer that stores samples across multiple time steps, to stabilize the adaptation. In fact, the moving window transforms the STTA task into a standard TTA task. However, using a large moving window to congregate samples significantly increases memory overhead and delays the updating. In [51], a class-balanced memory bank is introduced to facilitate STTA classification. However, maintaining such a memory bank requires enormous overhead, making it computationally inefficient. In [40], the entropy weight proposed in [34] is further optimized to filter out high-uncertainty samples more accurately and enhance the adaptation stability while keeping more samples. However, their method can only be applied to the methods without a standard batch normalization, limiting its applicability. In [27], the pre-trained CLIP [38] is employed to facilitate STTA for classification and detection tasks. This approach relies on the CLIP pre-trained on large-scale datasets, which may not be feasible in many scenarios. In Fig.3, we show that the performance of the aforementioned methods declines over a long time. To this end, we propose a memory-efficient buffer and elastic resetting control unit to jointly 1) stabilize adaptation efficiently and 2) prevent catastrophic forgetting.

3 Methodology

3.1 Problem Formulation

This paper focuses on the image classification task under the source-free single-sample continual test-time adaptation (S-CoTTA) setting. We begin with a source model $f_{\Theta_S}(x)$ pre-trained on the initial source domain Φ_S with data and labels $(\mathcal{X}^S, \mathcal{Y}^S)$, where $(\mathcal{X}^S, \mathcal{Y}^S)$ is unavailable for adaptation when testing. The goal is to achieve a higher classification accuracy for images from target test domains. During the test time, the model, initialized with $f_{\Theta_S}(x)$, classifies the target batches $\mathcal{X}^T = (X_1, X_2, ...)$ from test domains Φ_T different from Φ_S. In S-CoTTA, each data batch X_t contains only one sample, and the distribution of X_t changes continually. Thus, we have $X_t = (x_t)$. The data batches are presented to the model following a time-step sequence. At each time step t, the adapted model $f_{\Theta_t}(\mathrm{x})$ classifies the sample x_t while being adapted to Φ_T using unlabelled x_t for the next time step $t+1$.

3.2 Overall Architecture

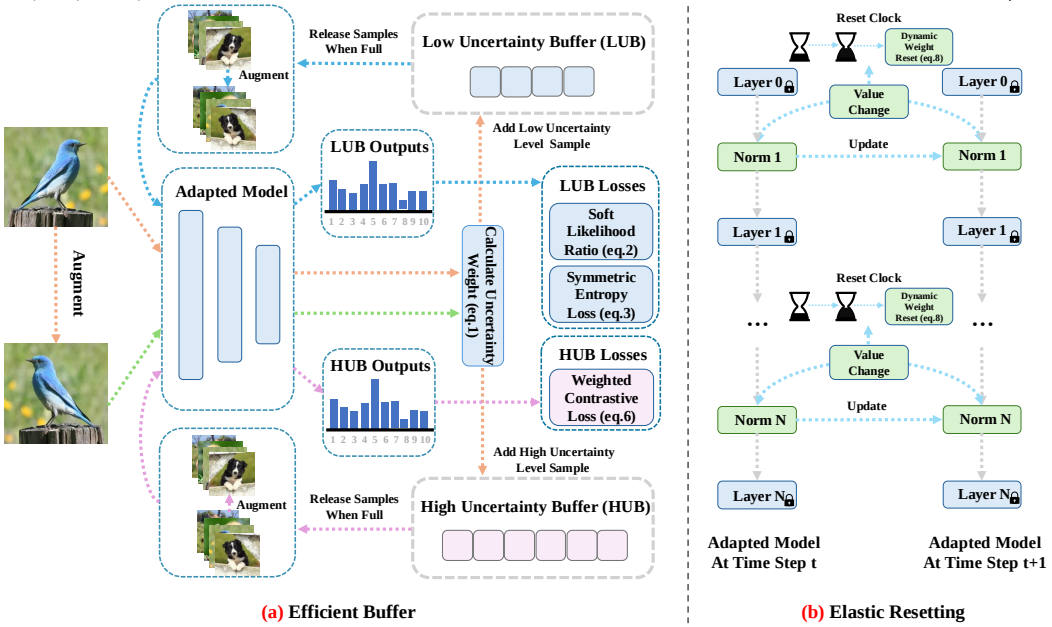
The overall architecture of our solution named **E**fficient **B**uffer **a**nd **R**esetting (EBaR) is illustrated in Fig.4. We begin with an initial source model, $f_{\Theta_S} = (F_S, H_S)$, and an adapted model, $f_{\Theta_t} = (F_t, H_t)$, at a time step t. Here, F and H represent a image encoder and a classification head with a softmax layer, respectively. At the time step 0, f_{Θ_0} is initialized with f_{Θ_S}.

At time step t, a test data batch $X_t = (x_t)$,consisting of a single sample x_t is presented to the adapted model $f_{\Theta_t} = (F_t, H_t)$ for classification. x_t belongs to one of the C categories, and the goal of f_{Θ_t} is to predict the category. First, x_t is fed into the image encoder F_t. We denote the encoded image as $o_t = F_t(x_t)$. Then, o_t is fed into the classification head H_t to generate the prediction $y_t = H_t(o_t)$. Finally, the classification accuracy is computed for x_t using y_t.

Meanwhile, the adapted model f_{Θ_t} is updated for the next time step $t+1$ with o_t and y_t being used. In EBaR, only the normalization layers are opened for updating, with other parameters fixed. The update procedure incorporates the **E**fficient **B**uffer (EB) and **E**lastic **R**esetting (ER) control unit, both are introduced with details in the following subsections.

3.3 Efficient Buffer for Stable Adaptation

We introduce a novel memory-efficient multilevel buffer named **E**fficient **B**uffer (EB). The purpose of EB is twofold: 1) stabilizing the adaptation by categorizing the samples based on their uncertainty level and applying different losses, and 2) reducing the memory overhead and enabling timely updates.



(a) Efficient Buffer (b) Elastic Resetting

Figure 4: EBaR, designed for the S-CoTTA task, consists of two key components: (a) an Efficient Buffer (EB) and (b) an Elastic Resetting (ER). As shown in (a), EB comprises a smaller Low Uncertainty Buffer (LUB) and a larger High Uncertainty Buffer (HUB). The original and augmented samples are first classified. Based on that, the uncertainty level for each sample is evaluated. The low-uncertainty and high-uncertainty samples are stored in LUB and HUB, respectively. Once a buffer is full, samples stored in the buffer are augmented and fed into the model for outputs. LUB and HUB losses are then computed for samples stored in the corresponding buffer and backpropagated to update the model. As shown in (b), ER features a reset clock that tracks parameter changes, enabling an asynchronous reset for each parameter based on its sensitivity to domain changes. When the clock exceeds the threshold, a dynamic weight reset is applied to set a parameter closer to its initial when it is sensitive.

Algorithm 1: Update the model with EB

Input: x_t, LUB, HUB

$x'_t \leftarrow$augmentation(x_t);
$y_t \leftarrow f_{\Theta_t}(x_t), y'_t \leftarrow f_{\Theta_t}(x'_t)$;
$\mathcal{L}_{Ut} \leftarrow \mathcal{L}_U(y_t, y'_t)$;
if $\mathcal{L}_{Ut} < \epsilon$ **then**
  if *number of samples in LUB equals to M* **then**
    $X_L \leftarrow$ LUB.stored_samples();
    $\mathcal{L}_{Lt} \leftarrow \mathcal{L}_L(X_L)$;
    LUB.clear_stored_samples();
  end
  LUB.add_sample(x_t);
else
  if *number of samples in HUB equals to N* **then**
    $X_H \leftarrow$ HUB.stored_samples();
    $\mathcal{L}_{Ht} \leftarrow \mathcal{L}_H(X_H)$;
    HUB.clear_stored_samples();
  end
  HUB.add_sample(x_t);
end
$\mathcal{L}_t \leftarrow \mathcal{L}_{Lt} + \mathcal{L}_{Ht}$;
$\mathcal{L}_t$.backpropagation();

Specifically, EB comprises a smaller **L**ow **U**ncertainty **B**uffer (LUB) and a larger **H**igh **U**ncertainty **B**uffer (HUB). LUB has a capacity of M samples, while HUB can store up to N samples. Samples with different uncertainty levels are stored in different buffers. Following [33, 40], we apply a symmetric entropy loss to indicate uncertainty levels for samples. Given a sample x_t, we obtain the prediction $y_t = f_{\Theta_t}(x_t)$. Then, the sample is augmented by applying color jitter, affine transformations, and horizontal flipping. The augmented sample, denoted as x'_t, is then fed into the model to obtain the prediction $y'_t = f_{\Theta_t}(x'_t)$. The uncertainty weight, denoted as $\mathcal{L}_U$, is computed as follows:

$$\mathcal{L}_U(y_t, y'_t) = -\Big(\sum_{c=1}^{C} y_{tc} \log y'_{tc} + \sum_{c=1}^{C} y'_{tc} \log y_{tc}\Big). \tag{1}$$

Next, we update the model using LUB and HUB, as outlined in Algo.1, where ϵ is a predefined threshold.

3.3.1 ***Low Uncertainty Buffer.*** The **L**ow **U**ncertainty **B**uffer (LUB) stores samples with low uncertainty levels. LUB has a smaller capacity M because low-uncertainty samples contain high-quality target knowledge with less noise [33] and can stably update the model without perturbation, even within a considerably small batch.

In Algo.1, LUB stores the samples whose $\mathcal{L}_U$ are below the threshold ϵ. Once the number of samples stored in LUB reaches M, the stored samples $X_L = (x_1, x_2, ...x_M)$ are released and processed to calculate the LUB loss $\mathcal{L}_L$. $\mathcal{L}_L$ consists of a soft likelihood ratio loss (SLR) $\mathcal{L}_{SLR}$, and a symmetric entropy loss (SEL) $\mathcal{L}_{SEL}$.

Specifically, we first obtain the predictions of X_L, denoted as $Y_L = (y_1, y_2, ..., y_M)$. $\mathcal{L}_{SLR}$ is calculated as follows:

$$\mathcal{L}_{SLR}(Y_L) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{c=1}^{C} y_{mc} \log\Big(\frac{y_{mc}}{\sum_{j\neq c}^{C} y_{mj} + \xi}\Big). \tag{2}$$

Meanwhile, we augment X_L to generate augmented samples $X'_L = (x'_1, x'_2, ...x'_M)$. X'_L is then fed into f_{Θ_t} to obtain the prediction $Y'_L = (y'_1, y'_2, ..., y'_M)$. $\mathcal{L}_{SEL}$ is then calculated as follows:

$$\mathcal{L}_{SEL}(Y_L, Y'_L) = -\frac{1}{2M}\sum_{m=1}^{M}(\sum_{c=1}^{C} y_{mc}\log y'_{mc} + \sum_{c=1}^{C} y'_{mc}\log y_{mc}). \quad (3)$$

$\mathcal{L}_{SLR}$ is less dominated by the predictions with low confidence [29, 31] and $\mathcal{L}_{SEL}$ is more robust to label noise compared to the entropy loss [7]. They both reduce the potential impact of pseudo-label perturbation in self-supervised adaptation. Jointly using two losses further stabilizes model updates when very limited samples are available at a time. Finally, the LUB loss $\mathcal{L}_L$ is calculated as:

$$\mathcal{L}_L = \mathcal{L}_{SLR} + \beta\mathcal{L}_{SEL}. \quad (4)$$

3.3.2 ***High Uncertainty Buffer.*** The **H**igh **U**ncertainty **B**uffer (HUB) stores the samples with high uncertainty levels. These samples disturb the adaptation, especially in a small batch [33]. However, we argue that completely discarding these samples, a strategy adopted by [33, 40], is suboptimal. Our rationale is twofold: 1) the model may initially overestimate uncertainty levels and can refine it after a few updates, and 2) high-uncertainty samples also contain valuable target-domain knowledge. To effectively exploit these samples, we allocate a larger capacity N to HUB. This reduces the immediate impact of these high-uncertainty samples and allows the model to reassess their uncertainty after a few more updates.

In Algo.1, HUB stores the samples whose $\mathcal{L}_U$ are higher than the threshold ϵ. Once the number of samples stored in HUB reaches N, the stored samples $X_H = (x_1, x_2, ...x_N)$ are released and processed to calculate the HUB loss $\mathcal{L}_H$. The loss $\mathcal{L}_H$ is calculated based on an entropy weight $\Delta_H = (\delta_1, \delta_2, ...\delta_n, ...\delta_N)$ and a weighted contrastive learning loss (CLL) $\mathcal{L}_{CLL}$.

Specifically, we first feed X_H into f_{Θ_t} to obtain the predictions $Y_H = (y_1, y_2, ...y_N)$. Next, augmentation is applied to X_H to generate augmented samples $X'_H = (x'_1, x'_2, ...x'_N)$. Then, X'_H is fed into f_{Θ_t} to generate predictions $Y'_H = (y'_1, y'_2, ...y'_N)$. The entropy weight δ_n for each stored sample x_n is computed as:

$$\delta_n = \exp\big(min(\sum_{c=1}^{C} y_{nc}\log y'_{nc} + \sum_{c=1}^{C} y'_{nc}\log y_{nc} + \epsilon, 0)\big). \quad (5)$$

Meanwhile, X_H and X'_H are fed into the encoder F_t to obtain the encoded images $O_H = (o_1, o_2, ...o_N)$ and $O'_H = (o'_1, o'_2, ...o'_N)$, respectively. $\mathcal{L}_{CLL}$ is then computed as:

$$\mathcal{L}_{CLL}(O_H, O'_H, \Delta_H) = -\frac{1}{N}\sum_{n=1}^{N}\delta_n(\sum_{c=1}^{C}\frac{\exp\big(\mathrm{sim}(o_{nc}, o'_{nc})/\tau\big)}{\sum_{k=1}^{N}\exp\big(\mathrm{sim}(o_{nc}, o'_{kc})/\tau\big)}), \quad (6)$$

where sim represents the cosine similarity, and τ is the temperature.

By weighting contrastive learning loss with the entropy weight, we achieve two key objectives: 1) extracting target-domain knowledge from samples with higher uncertainty without using potentially noisy pseudo-labels and 2) re-estimating sample uncertainty levels to modulate their contribution to model updates. This strategy mitigates the adverse effects of samples with high uncertainty while extracting useful target-domain knowledge. The HUB loss $\mathcal{L}_H$ is computed as: $\mathcal{L}_H = \mathcal{L}_{CLL}$.

3.4 Elastic Resetting for Anti-forgetting

Algorithm 2: Elastic resetting for parameters

Input: $\Theta_t, \Theta_{t-1}, R, \mu, \eta, \Upsilon, T$
for (θ_t, θ_{t-1}) *in* (Θ_t, Θ_{t-1}) **do**
  if θ_t *is updated* **then**
    $D \leftarrow \frac{||\theta_t - \theta_{t+1}||}{||\theta_t|| + ||\theta_{t+1}||}$;
    $R_\theta \leftarrow R_\theta + \eta D + \mu$;
  else
    $R_\theta \leftarrow R_\theta + \mu$;
  end
  $T_\theta \leftarrow T_\theta + \mu$;
  if $R_\theta \geq \Upsilon$ **then**
    θ.reset(T_θ, Υ) ;
    $R_\theta \leftarrow 0, T_\theta \leftarrow 0$;
  end
end

We introduce the **E**lastic **R**esetting (ER) control unit to mitigate catastrophic forgetting over long time steps. Unlike existing strategies [36, 42], which periodically reset the entire model back to the initial after a fixed number of time steps, ER employs an asynchronous resetting mechanism. In ER, different parameters are reset at varying frequencies based on their sensitivity to domain shifts. Our approach is motivated by two key insights: 1) the resetting operation effectively restores the source knowledge, and 2) not all parameters are equally susceptible to domain shifts: some are comparably robust to perturbations brought by domain shifts and effectively absorb rich target-domain knowledge. ER balances preserving source knowledge and retaining target knowledge by more frequently resetting only the parameters sensitive to perturbations.

To effectively implement ER, we introduce a reset clock R for each parameter θ, which operates based on Algo.2. In the algorithm, Θ_t and Θ_{t-1} represent the values of the updated parameters at the time steps t and $t-1$, respectively. The reset process begins at time step 1, with three hyper-parameters: μ, η, and Υ. In Algo.2, as the time step increases, the reset clock R_θ for each parameter θ increases by: 1) a fixed increment μ, and 2) an increment related proportionally to the magnitude of the changes in the parameter value during the update, adjusted by η. When R_θ reaches or exceeds the threshold Υ, the corresponding parameter θ is reset.

Unlike [36], which resets parameters to their initial values θ_0, ER employs a novel dynamic weight reset based on the sensitivity of each parameter to perturbations. We introduce a sensitive indicator T in Algo.2. Every time step, T_θ for each parameter θ increases by μ. The dynamic weight reset is performed as follows:

$$\theta'_t = (1-\alpha)\frac{T_\theta}{\Upsilon}\theta_t + (1 - \frac{T_\theta}{\Upsilon} + \alpha\frac{T_\theta}{\Upsilon})\theta_0, \quad (7)$$

where α is a hyper-parameter that adjusts the reset.

When reset occurs, a higher T_θ indicates that a parameter is reset less frequently and is less sensitive to perturbations caused by domain shifts. Consequently, parameters with higher T_θ are reset less close to their initial values θ_0, ensuring a balance between retraining target knowledge and mitigating forgetting.

Table 1: The averaged classification accuracy (%) of different methods. All the methods except EBaR have a moving window of size 32 added for better convergence. 'Source' represents the backbone model without adaptation. 'CMAE' only supports Transformer baselines. EBaR achieves state-of-the-art performance across multiple datasets with different baselines.

Dataset	Backbone	SOURCE	TENT [45]	COTTA [46]	ETA [33]	ViDA [25]	REALM [40]	SAR [34]	IST [27]	CMAE [24]	ROTTA [51]	RDUMB [36]	BDG [50]	ROID [29]	EBaR
CIFAR10	WideResNet	56.4	73.8	76.6	77.4	79.9	79.8	77.1	**80.8**	-	79.0	79.9	80.4	80.6	**84.3** (+3.5)
CIFAR100	ResNeXT-29	53.6	29.8	61.6	59.7	66.5	65.3	67.1	66.2	-	64.8	66.0	**68.0**	**67.6**	**71.9** (+3.9)
IN-C	ResNet-50	18.1	33.1	33.7	39.6	43.8	41.7	36.6	36.2	-	30.1	40.3	**44.1**	43.7	**47.6** (+3.5)
	ViT-B-16	39.9	23.2	13.7	43.6	50.5	44.9	49.8	50.4	51.3	44.1	46.1	52.4	**52.5**	**56.7** (+4.2)
	Swin-B	45.3	19.8	9.9	54.8	56.8	52.7	47.8	60.2	**62.1**	44.8	53.2	**62.1**	61.8	**67.3** (+5.2)
CCC	ResNet-50	33.5	5.7	7.6	6.8	16.8	31.5	39.3	15.8	-	10.7	41.3	45.2	**47.6**	**50.6** (+3.0)
	ViT-B-16	53.5	4.8	5.3	9.0	36.9	62.8	**65.8**	39.7	30.9	9.7	64.9	64.2	65.6	**70.3** (+4.5)
	Swin-B	55.8	3.9	4.1	7.9	39.8	63.9	65.4	39.6	30.2	9.8	64.5	64.1	**65.5**	**70.9** (+5.4)

Table 2: The classification accuracy (%) of different methods using ResNet-50 on IN-C across 15 corruptions. EBaR achieves the highest accuracy on 10 out of 15 corruptions.

Method	gaussian	shot	impulse	defocus	glass	motion	zoom	snow	frost	fog	bright	contrast	elsatic	pixelate	jpeg	Avg.
SOURCE	2.2	3.0	1.9	18.6	10.2	14.8	22.0	16.5	23.2	24.1	58.7	6.0	17.5	20.7	31.5	18.1
TENT	15.1	21.4	23.2	17.8	23.1	29.9	40.8	33.5	32.0	43.4	53.4	24.0	45.8	49.1	43.3	33.1
COTTA	12.8	17.1	19.7	17.0	17.6	24.2	37.5	34.1	32.6	47.7	59.9	30.6	48.6	55.4	51.2	33.7
ETA	21.4	31.9	34.6	21.7	26.9	35.1	44.0	39.8	37.4	50.3	59.6	36.4	50.6	53.6	50.7	39.6
ViDA	18.8	23.3	24.7	22.0	21.7	33.6	42.4	38.7	36.1	48.2	59.5	32.0	47.6	53.1	50.7	36.8
REALM	22.4	30.1	35.0	24.1	25.2	36.2	44.3	44.3	43.2	45.2	**73.1**	**52.8**	40.2	55.1	54.2	41.7
SAR	17.9	23.8	27.7	20.1	23.6	33.9	43.1	37.0	37.1	47.0	60.2	29.8	49.1	50.2	48.1	36.6
IST	27.3	28.6	26.9	20.0	18.8	29.8	41.9	39.7	35.9	49.7	60.8	15.8	47.2	51.3	49.0	36.2
ROTTA	10.7	15.5	14.7	7.8	15.3	23.8	38.2	32.1	33.6	43.1	60.5	21.2	41.7	49.2	44.4	30.1
RDUMB	21.8	33.2	35.8	22.4	25.8	37.0	44.8	41.4	37.1	49.8	59.6	38.1	51.5	54.2	51.4	40.3
BDG	24.8	33.4	35.7	28.9	**39.5**	38.9	50.6	46.3	**45.1**	56.4	62.5	39.6	54.9	54.5	50.8	44.1
ROID	26.3	**37.8**	36.9	29.7	32.1	42.9	46.6	43.8	43.7	53.2	63.8	37.4	52.7	56.3	52.0	43.7
EBaR (ours)	**30.9**	37.5	**41.5**	**32.8**	33.2	**43.9**	**53.4**	**49.8**	43.9	**59.6**	69.9	43.6	**56.9**	**59.8**	**56.8**	**47.6**

4 Experiments

4.1 Settings

4.1.1 ***Datasets.*** EBaR is evaluated on four CoTTA datasets: CIFAR10-C, CIFAR100-C, ImageNet-C (IN-C) [15, 16, 35, 39], and CCC [36]. CIFAR10-C, CIFAR100-C, and IN-C contain 15 types of image corruption, each with 5 severity levels. The experiments are conducted at severity level 5. The CCC dataset (Continuously Changing Corruptions) comprises 7.5 million images with smooth transitions between different IN-C corruptions [36]. CCC is designed to assess the stability of test-time adaptation methods over extended time steps. Our experiments use CCC-40, where the source ResNet-50 model achieves 35% to 40% accuracy without adaptation. The results on CCC datasets are averaged over 3 different random seeds.

4.1.2 ***Implementation Details.*** EBaR is evaluated on five backbones: WideResNet-28 (WRN-28) [52], ResNeXt-29 [49], ResNet-50 [13], ViT Base (Vit-B-16) [8], and Swin Transformer Base (Swin-B-16) [26]. Following [29], the evaluation on CIFAR10-C is performed with WRN-28, and CIFAR100-C with ResNeXt-29. ResNet-50, Vit-B-16, and Swin-B-16 are evaluated for IN-C and CCC. WRN-28 and ResNeXt-29 are pretrained with CIFAR datasets [20], while ResNet-50, Vit-B-16 and Swin-B-16 are pretrained with ImageNet [6], following [5, 43, 44, 53, 59]. We set the learning rate to 1×10^{-4}, LUB size M to 2, HUB size N to 8, ϵ (Algo.1) to 6.2, ξ (Eq.2) to 0.01, β (Eq.4) to 1, and τ (Eq.6) to 0.05, μ (Algo.2) to 1, η (Algo.2) to 100, Υ (Algo.2) to 8000, and α (Eq.7) to 0.99 as default.

4.2 Effectiveness of EBaR

We compare EBaR with state-of-the-art CoTTA and STTA methods in Tab.1 and Tab.2. For a fair comparison, we apply a moving window of size 32 to all counterpart methods, following [7, 29, 33].

The results show that under S-CoTTA, EBaR outperforms state-of-the-art methods by clear margins: (1) across multiple datasets with various backbone architectures (Tab.1), and (2) on 10 of 15 corruptions for the IN-C dataset using the ResNet-50 backbone (Tab.2). For instance, EBaR achieves +3.5%, +4.2%, and +5.2% on IN-C with ResNet-50, ViT-B-16, and Swin-B-16.

4.3 Effectiveness of Key Components

4.3.1 ***Efficiency Buffer.*** We evaluate the effectiveness of the Efficient Buffer, including LUB, HUB, and their associated loss functions. For comparison, we choose the baseline method ETA [33] with a moving window size of 32. As shown in Tab.3, both LUB and HUB, along with their loss functions, boost accuracy. Furthermore, combining LUB and HUB yields a more significant accuracy gain (+5.6%) compared to using them separately (+2.5% and +1.8%).

Table 3: Evaluation of the key components: efficient buffer and elastic resetting on IN-C with ResNet-50. We list the averaged accuracy (%). 'DWR' represents dynamic weight reset (Eq.7). The result proves the effectiveness of both.

Method	$\mathcal{L}_{SLR}$	$\mathcal{L}_{SEL}$	$\mathcal{L}_{CLL}$	*Clock*	*DWR*	Average Acc.
Baseline (ETA)	✗	✗	✗	✗	✗	39.6
LUB + HUB	✓	✓	✓	✗	✗	45.2 (+5.6%)
LUB only	✓	✓	✗	✗	✗	42.1 (+2.5%)
HUB only	✗	✗	✓	✗	✗	41.4 (+1.8%)
LUB only -$\mathcal{L}_{SLR}$	✓	✗	✗	✗	✗	41.1 (+1.5%)
LUB only -$\mathcal{L}_{SEL}$	✗	✓	✗	✗	✗	40.2 (+0.6%)
ER only	✗	✗	✗	✓	✓	44.3 (+4.7%)
ER only - DWR	✗	✗	✗	✓	✗	42.7 (+3.1%)
DWR only	✗	✗	✗	✗	✓	42.2 (+2.6%)
EBaR	✓	✓	✓	✓	✓	**47.6** (+8.0%)

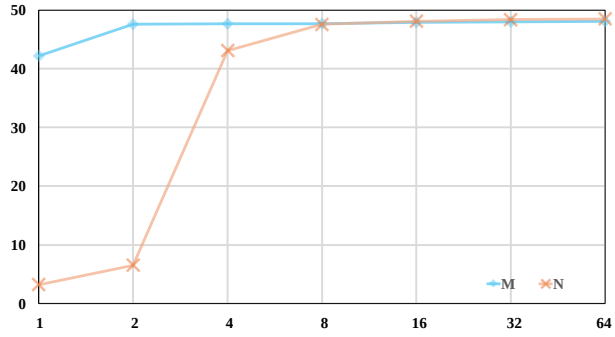


Figure 5: Effect of buffer sizes M and N. Averaged accuracy with different LUB size M and HUB size N on IN-C using ResNet-50 is reported. The optimum setting is $M = 2, N = 8$.

4.3.2 ***Elastic Resetting.*** We evaluate the effectiveness of the Elastic Resetting and its two key components, the reset clock and the dynamic weight reset. As shown in Tab.3, both clock-based reset and dynamic weight reset boost performance. Moreover, combining ER and EB yields a more significant increase in accuracy of +8.0%.

4.4 Ablation

We evaluate the influence of several key hyper-parameters. More hyper-parameter studies are detailed in the supplement.

4.4.1 ***Buffer Size.*** We evaluate the impact of the LUB and HUB sizes M and N (Algo.1). In the LUB experiment group, we set $N = 8$; in the HUB experiment group, we set $M = 2$. As shown in Fig.5, EBaR achieves optimal performance when M reaches 2, with minimal improvement when $M > 2$. A similar trend is observed for N, where performance stabilizes after N reaches 8. In particular, when N is smaller than 4, EBaR collapses due to the influence of high-uncertainty samples. These samples cause a misestimation of distribution and disrupt the updates. To balance memory efficiency and accuracy, we set $M = 2$ and $N = 8$ as default.

4.4.2 ***Uncertainty Level Threshold.*** We evaluate the impact of the uncertainty level threshold ϵ (Algo.1). The result is shown in Fig.6. As ϵ increases, the proportion of samples classified as high uncertainty gradually decreases, drastically dropping when $\epsilon \geq 6$. The high- and low-uncertainty samples are balanced when $\epsilon \in (6.1, 6.4)$.

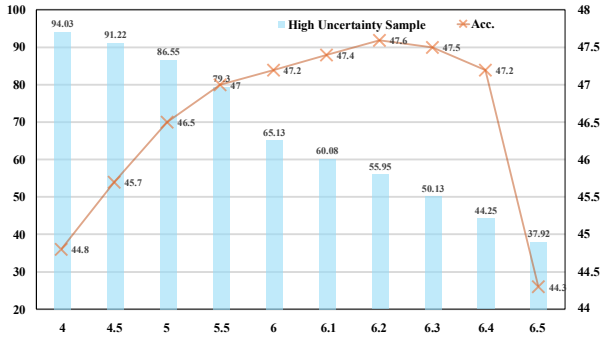


Figure 6: Effect of uncertainty level threshold ϵ. Experiments are conducted on IN-C with ResNet-50. The percentage of samples categorized as high uncertainty and the corresponding accuracy are reported. $\epsilon = 6.2$ is the optimum.

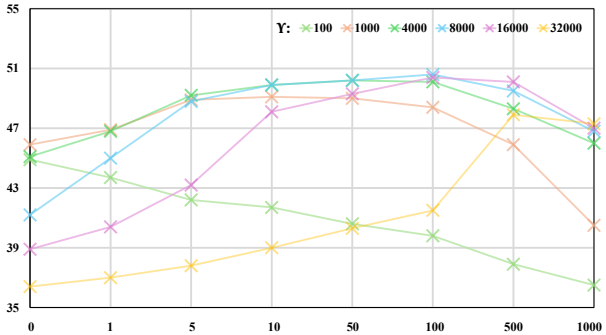


Figure 7: Effect of the hyper-parameters that control the reset frequency: η, and Υ (Algo.2). We fix $\mu = 1$. The experiment is conducted on the CCC with ResNet-50. Averaged accuracy is reported. The result shows that $\eta = 100, \Upsilon = 8000$ is optimum.

When most samples are categorized as high uncertainty, LUB becomes ineffective. In contrast, HUB is ineffective when most are categorized as low uncertainty. EBaR achieves the highest accuracy $\epsilon = 6.2$, where samples from both levels are balanced. Thus, we set $\epsilon = 6.2$ as the default.

4.4.3 ***Resetting Frequency.*** We evaluate the influence of the hyper-parameters that control the reset frequency, namely μ, η, and υ (Algo.2). We fix $\mu = 1$ in the experiments and adjust the other two parameters. As shown in Fig.7, when Υ increases, the optimal η increases. When Υ is small while η is large, the model is reset excessively frequently, harming the adaptation. Conversely, when Υ is large while η is small, the model is not reset in time, leading to catastrophic forgetting. $\eta = 100, \Upsilon = 8000$ achieve the highest accuracy and are set as default.

4.5 Observation

4.5.1 ***Computational Efficiency.*** We evaluate the computational efficiency of EBaR. As shown in Tab.4, we compare the computational overhead of ETA, ROID, and EBaR on a single RTX A5500 GPU. Without a moving window, ETA and ROID both collapse. Added a moving window of size 32, ETA and ROID converge. Still, EBaR achieves a higher classification accuracy, requiring only 60% of the memory overhead. Furthermore, EBaR improves accuracy by

Table 4: Computational efficiency of EBaR. Experiment is conducted on IN-C with ResNet-50 backbone. 'W32' represents a moving window [29, 33] of size 32. The experiment is conducted on a single RTX A5500 GPU. EBaR yields higher accuracy boost with less extra computational overhead.

Method	Memory Cost	Time Per Prediction	Average Acc.
ETA	0.93G	3.4 ms	4.1
ETA+W32	6.22G	2.1 ms	39.6
ROID	1.13G	41.6 ms	5.2
ROID+W32	6.42G	5.3 ms	43.7
EBaR	3.72G	4.1 ms	**47.6**

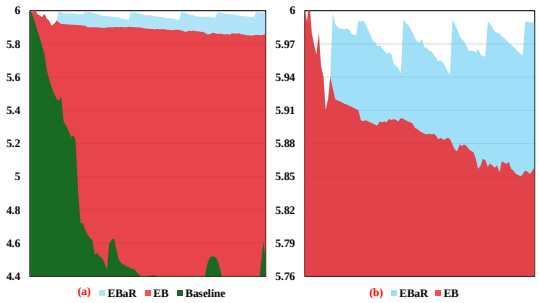


Figure 8: The adaptation stability of EBaR. EBaR is compared with EBaR with Efficient Buffer Only (EB), and ETA with window size 32 (Baseline) on the CCC dataset across 3×10^4 steps with ResNet-50. The averaged weight value ($\times 10^{-3}$) of a normalization layer is depicted. The result shows that EB significantly enhances adaptation stability while elastic resetting maintains stability over the long run.

Table 5: EBaR against other CoTTA counterparts under the standard CoTTA setting on IN-C using ResNet-50 backbone. Averaged accuracy is reported. EBaR demonstrates more substantial advantages under small-batch settings while achieving comparable performance under large-batch settings.

Batch Size	ETA	ViDA	IST	ROID	BDG	EBaR
4	19.9	20.1	33.1	**40.5**	40.4	**47.9**
8	22.1	30.4	35.4	42.8	**42.9**	**48.2**
16	31.8	36.9	36.8	43.5	**43.6**	**48.6**
32	40.1	44.3	37.6	44.2	**44.5**	**48.8**
64	40.2	45.7	38.1	48.1	**48.9**	48.8

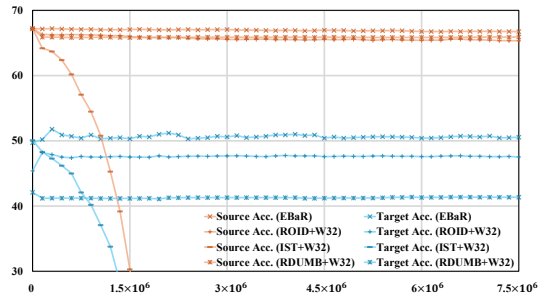


Figure 9: Anti-forgetting ability of EBaR. EBaR is compared with IST, ROID, and RDUMB (all three of them are with a moving window of size 32) on ImageNet (Source) and CCC (Target) using ResNet-50. The average accuracy per 1.5×10^5 steps across 7.5×10^6 steps is reported. The result demonstrates that EBaR effectively prevents catastrophic forgetting while retaining useful target knowledge.

+8% with only 1.9 ms of additional time per prediction compared to the ETA baseline. In comparison, ROID achieves lower accuracy while incurring 3.2 ms of extra time per prediction. Interestingly, we observe that the moving window reduces the prediction time by decreasing the frequency of model updates.

4.5.2 ***Stable Adaptation.*** We examine the ability of EBaR to enhance adaptation stability by analyzing the fluctuation in parameter values of normalization layers over time. Fig.8 depicts the averaged weight values of a batch normalization layer in the ResNet-50 backbone. The baseline model (ETA with window size 32) suffers from perturbations of domain shift, resulting in severe fluctuations in parameter values. The efficient buffer increases robustness to perturbations and reduces fluctuations significantly. Furthermore, elastic resetting maintains adaptation stability in the long run.

4.5.3 ***Anti-forgetting.*** We compare the ability of EBaR to prevent catastrophic forgetting with IST, ROID, and RDUMB (three counterparts have an additional moving window of size 32). As shown in Fig.9, the average accuracies per 1.5×10^5 time steps for CCC (denoted as Target) and original ImageNet (denoted as Source) are reported across 7.5×10^6 time steps. EBaR achieves the highest average source accuracy (66.9%) and target accuracy (50.6%). The result demonstrates that EBaR can effectively prevent catastrophic forgetting while retaining more useful target knowledge.

4.5.4 ***Under CoTTA Setting.*** We compare EBaR with recent CoTTA methods on the CoTTA task under different batch size settings. In this experiment, no moving window is applied to the counterparts. As shown in Tab.5, EBaR demonstrates a significant advantage with the small-batch settings. Even with large-batch settings, EBaR achieves performance comparable to the state-of-the-art methods on standard CoTTA tasks, highlighting the robustness and versatility of our approach.

5 Conclusion

In this paper, we propose a single-sample continual test-time domain adaptation (S-CoTTA) task, which combines single-sample adaptation, where only one sample is available at a time, and continual adaptation, where the target domains constantly change. The key challenge is to achieve stable adaptation with a single sample per time step and prevent the model from forgetting the source knowledge over time. To address this, we propose a novel Efficient Buffer and Resetting (EBaR). EBaR consists of 1) a memory-efficient buffer that efficiently mitigates the perturbations of a sample with high uncertainty and ensures stable adaptations and 2) an elastic resetting unit that prevents catastrophic forgetting while retaining useful target knowledge. Experiments demonstrate the effectiveness of our method: EBaR effectively enhances the adaptation stability and prevents the model from forgetting the source knowledge.

References

[1] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. 2022. Mt3: Meta test-time training for self-supervised test-time adaption. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3080–3090.

[2] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. 2021. Adamatch: A unified approach to semi-supervised learning and domain adaptation. *arXiv preprint arXiv:2106.04732* (2021).

[3] Goirik Chakrabarty, Manogna Sreenivas, and Soma Biswas. 2023. SANTA: Source Anchoring Network and Target Alignment for Continual Test Time Adaptation. *Transactions on Machine Learning Research* (2023).

[4] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. 2022. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 295–305.

[5] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. 2021. When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations. *arXiv preprint arXiv:2106.01548* (2021).

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[7] Mario Döbler, Robert A Marsden, and Bin Yang. 2023. Robust mean teacher for continual and gradual test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7704–7714.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Yulu Gan, Yan Bai, Yihang Lou, Xianzheng Ma, Renrui Zhang, Nian Shi, and Lin Luo. 2023. Decorate the newcomers: Visual domain prompt for continual test time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 7595–7603.

[10] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. 2023. Back to the source: Diffusion-driven adaptation to test-time corruption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11786–11796.

[11] Michael Goetz, Christian Weber, Franciszek Binczyk, Joanna Polanska, Rafal Tarnawski, Barbara Bobek-Billewicz, Ullrich Koethe, Jens Kleesiek, Bram Stieltjes, and Klaus H Maier-Hein. 2015. DALSA: Domain adaptation for supervised learning from sparsely annotated MR images. *IEEE transactions on medical imaging* 35, 1 (2015), 184–196.

[12] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. 2022. NOTE: Robust continual test-time adaptation against temporal correlation. *Advances in Neural Information Processing Systems* 35 (2022), 27253–27266.

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

[14] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. 2023. Parameter-efficient model adaptation for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 817–825.

[15] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. 2021. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*. 8340–8349.

[16] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *Proceedings of the International Conference on Learning Representations* (2019).

[17] Minguk Jang, Sae-Young Chung, and Hye Won Chung. 2022. Test-time adaptation via self-training with nearest neighbor information. *arXiv preprint arXiv:2207.10792* (2022).

[18] Shibo Jie and Zhi-Hong Deng. 2023. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 1060–1068.

[19] Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. 2023. Robustifying Vision Transformer Without Retraining from Scratch Using Attention-Based Test-Time Adaptation. *New Generation Computing* 41, 1 (2023), 5–24.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[21] Wen Li, Lixin Duan, Dong Xu, and Ivor W Tsang. 2013. Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation. *IEEE Transactions on Pattern analysis and machine intelligence* 36, 6 (2013), 1134–1148.

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

[23] Jiachen Liang, Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. 2023. Generalized semi-supervised learning via self-supervised feature adaptation. *Advances in Neural Information Processing Systems* 36 (2023), 60791–60803.

[24] Jiaming Liu, Ran Xu, Senqiao Yang, Renrui Zhang, Qizhe Zhang, Zehui Chen, Yandong Guo, and Shanghang Zhang. 2024. Continual-mae: Adaptive distribution masked autoencoders for continual test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 28653–28663.

[25] Jiaming Liu, Senqiao Yang, Peidong Jia, Renrui Zhang, Ming Lu, Yandong Guo, Wei Xue, and Shanghang Zhang. 2023. Vida: Homeostatic visual domain adapter for continual test time adaptation. *arXiv preprint arXiv:2306.04344* (2023).

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[27] Jing Ma. 2024. Improved self-training for test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 23701–23710.

[28] Tianyi Ma, Yifan Sun, Zongxin Yang, and Yi Yang. 2023. ProD: Prompting-To-Disentangle Domain Knowledge for Cross-Domain Few-Shot Image Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19754–19763.

[29] Robert A Marsden, Mario Döbler, and Bin Yang. 2024. 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*. 2555–2565.

[30] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. 2022. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14765–14775.

[31] Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. 2021. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999* (2021).

[32] Fahim Faisal Niloy, Sk Miraj Ahmed, Dripta S Raychaudhuri, Samet Oymak, and Amit K Roy-Chowdhury. 2024. Effective Restoration of Source Knowledge in Continual Test Time Adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2091–2100.

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

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

[35] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. 2019. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1406–1415.

[36] Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. 2023. Rdumb: A simple approach that questions our progress in continual test-time adaptation. *Advances in Neural Information Processing Systems* 36 (2023), 39915–39935.

[37] Xiaoxue Qian, Weiguo Lu, and You Zhang. 2024. Adaptive wavelet-VNet for single-sample test time adaptation in medical image segmentation. *Medical physics* 51, 12 (2024), 8865–8881.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV] https://arxiv.org/abs/2103.00020

[39] Evgenia Rusak, Steffen Schneider, Peter Vincent Gehler, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. 2022. ImageNet-D: A new challenging robustness dataset inspired by domain adaptation. In *ICML 2022 Shift Happens Workshop*.

[40] Skyler Seto, Barry-John Theobald, Federico Danieli, Navdeep Jaitly, and Dan Busbridge. 2024. REALM: Robust entropy adaptive loss minimization for improved single-sample test-time adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2062–2071.

[41] Damian Sójka, Sebastian Cygert, Bartłomiej Twardowski, and Tomasz Trzciński. 2023. AR-TTA: A Simple Method for Real-World Continual Test-Time Adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3491–3495.

[42] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. 2023. EcoTTA: Memory-Efficient Continual Test-time Adaptation via Self-distilled Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11920–11929.

[43] Andreas Steiner, Alexander Kolesnikov, , Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. 2021. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. *arXiv preprint arXiv:2106.10270* (2021).

[44] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. MLP-Mixer: An all-MLP Architecture for Vision. *arXiv preprint arXiv:2105.01601* (2021).

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

[46] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. 2022. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7201–7211.

[47] Yanshuo Wang, Jie Hong, Ali Cheraghian, Shafin Rahman, David Ahmedt-Aristizabal, Lars Petersson, and Mehrtash Harandi. 2024. Continual Test-time Domain Adaptation via Dynamic Sample Selection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1701–1710.

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

[49] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1492–1500.

[50] Xu Yang, Xuan Chen, Moqi Li, Kun Wei, and Cheng Deng. 2024. A Versatile Framework for Continual Test-Time Domain Adaptation: Balancing Discriminability and Generalizability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 23731–23740.

[51] Longhui Yuan, Binhui Xie, and Shuang Li. 2023. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15922–15932.

[52] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).

[53] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. 2022. LiT: Zero-Shot Transfer with Locked-image Text Tuning. *CVPR* (2022).

[54] Dong Zhang, Daniel Gatica-Perez, Samy Bengio, and Iain McCowan. 2005. Semi-supervised adapted hmms for unusual event detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1. IEEE, 611–618.

[55] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. 2023. Domainadaptor: A novel approach to test-time adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 18971–18981.

[56] Jiaxin Zhang, Yiqi Wang, Xihong Yang, and En Zhu. 2024. A Fully Test-Time Training Framework for Semi-Supervised Node Classification on Out-of-Distribution Graphs. *ACM Transactions on Knowledge Discovery from Data* 18, 7 (2024), 1–19.

[57] Marvin Zhang, Sergey Levine, and Chelsea Finn. 2022. Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems* 35 (2022), 38629–38642.

[58] Yifan Zhang, Xue Wang, Kexin Jin, Kun Yuan, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. 2023. Adanpc: Exploring non-parametric classifier for test-time adaptation. In *International Conference on Machine Learning*. PMLR, 41647–41676.

[59] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. 2022. Surrogate Gap Minimization Improves Sharpness-Aware Training. *ICLR* (2022).