# WHEN AND WHERE TO RESET MATTERS FOR LONG-TERM TEST-TIME ADAPTATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

When continual test-time adaptation (TTA) persists over the long term, errors accumulate in a model and further lead it to predict only a few classes regardless of the input, known as *model collapse*. Recent studies have explored reset strategies that erase these accumulated errors completely. However, their periodic resets lead to suboptimal adaptation, as they occur independently of collapse. Also, their full resets cause the catastrophic loss of knowledge acquired over time, even though it could be beneficial in future. To this end, we propose 1) an **A**daptive and **S**elective **R**eset (**ASR**) scheme that dynamically determines when and where to reset, 2) an importance-aware regularizer to recover essential knowledge lost from reset, and 3) an on-the-fly adaptation adjustment scheme to enhance adaptability under challenging domain shifts. Extensive experiments across long-term TTA benchmarks demonstrate the effectiveness of our approach, particularly under challenging conditions. Our code will be released.

## 1 INTRODUCTION

Test-time adaptation (TTA) (Liang et al., 2020; Sun et al., 2020; Wang et al., 2021) aims to address the growing challenge of distribution shifts in real-world applications by enabling model adaptation at test time. Recently, TTA research has expanded to continual scenarios (Wang et al., 2022; Döbler et al., 2023), allowing models to adapt to a non-stationary stream of domains, where updates progress continuously, while errors accumulate over time. However, when domain shifts persist over the long term, these errors further result in *model collapse* (Niu et al., 2023; Shumailov et al., 2024), in which models converge to generate incorrect predictions concentrated on only a few classes across inputs.

To address this, recent studies have explored methods seeking to preserve knowledge from the source domain when adapting to target domains (Wang et al., 2022; Marsden et al., 2024; Press et al., 2023). A straightforward yet effective method involves periodically resetting model parameters to those of the source model (Press et al., 2023), which erases accumulated updates and errors, thereby rescuing the model from irreversible collapse. However, such a mechanism forces resets to depend on a single pre-defined reset interval across all situations, leading to too frequent or infrequent resets. Moreover, this completely erases knowledge acquired during adaptation, thereby disrupting forward knowledge transfer within the continuously adapting model (Díaz-Rodríguez et al., 2018).

To this end, we propose an **A**daptive and **S**elective **R**eset (**ASR**) scheme that dynamically determines when and where to reset based on the concentration of predicted classes, which is utilized to estimate the risk of model collapse. We trigger a reset once the risk is deemed significant, and adjust its scope based on how significant the risk is. Several studies (Bai et al., 2021; Yang et al., 2024) showed that corruption from label noise begins at the end of the network. Since this corruption results in collapse, we prioritize layers closer to the output for reset. Fig. 1 illustrates how our ASR scheme differs from the aforesaid naive reset approach. Besides, we introduce an importance-aware regularizer to recover essential knowledge lost from reset. We estimate parameter importance through a newly formulated Fisher information. Based on this, parameters regarded as crucial to previous tasks are aligned with their accumulated state, which incorporates all prior target knowledge. Finally, we propose to adjust our adapting mechanism on the fly based on domain discrepancy. We define prediction inconsistency to quantify this discrepancy, and then use it to update model hyperparameters via reparameterization, improving our adaptability under challenging domain shifts. Our contributions are as follows:

- We propose an *Adaptive and Selective Reset (ASR)* scheme that dynamically determines when and where to reset, effectively preventing model collapse while mitigating knowledge loss.

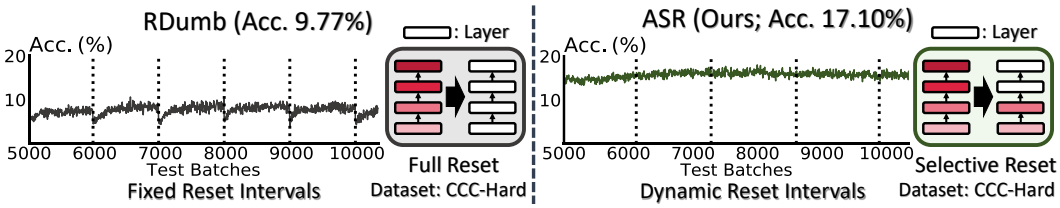


Figure 1: Illustrative comparison between a naive reset approach (RDumb; Press et al. (2023)) and our Adaptive and Selective Reset (ASR) on the same model (ETA; Niu et al. (2022)). RDumb fully resets parameters at fixed intervals (e.g., every 1000 steps), whereas ASR dynamically decides when and where to reset, achieving more stable (smaller fluctuations) and higher (+7.33%p) performance. Dotted vertical lines indicate when resets occur.

- Beyond the reset strategy, we introduce an *importance-aware regularizer* to recover parameters that are inevitably reset but deemed crucial to prior tasks, and *on-the-fly adaptation adjustment* that updates model hyperparameters according to domain discrepancy to enhance adaptability.
- Extensive experimental results across various long-term TTA benchmarks demonstrate the effectiveness of our method. Remarkably, our method yields a substantial 44.12% improvement over the state of the art on the challenging CCC-Hard (Press et al., 2023).

## 2 Related Work

**Test-time adaptation.** TTA enables a model to adapt to unknown target environments without any target assumptions. Since true labels are unavailable at test time, early works have explored effective unsupervised adaptation (Kundu et al., 2020; Li et al., 2020; Liang et al., 2020). Initial TTA research proposed to adjust batch normalization statistics (Schneider et al., 2020; Mirza et al., 2022), which evolved toward integrating self-training schemes (Zhang et al., 2022; Goyal et al., 2022), such as entropy minimization, improving predictive confidence on target data (Wang et al., 2021), which has been developed to prevent wrong confidence intensification (Zhang et al., 2025a; Han et al., 2025).

**Continual test-time adaptation.** Self-training methods face a critical challenge in a non-stationary domain stream, where their performance gradually deteriorates over time with noisy pseudo-labeling repeated (Wang et al., 2022; Niu et al., 2023). It accumulates errors, enhancing predictive confidence in incorrect predictions, eventually leading them to converge to suboptimal solutions, a phenomenon known as model collapse (Niu et al., 2023; Shumailov et al., 2024). Several studies (Niu et al., 2023; Hoang et al., 2024) empirically illustrated that once collapsing, a model assigns all inputs into a few dominant classes. CoTTA (Wang et al., 2022) addresses this collapse by stabilizing its self-training scheme using augmentation-averaged pseudo-labels and preventing source knowledge forgetting via stochastic parameter restoration. On the one hand, to handle error accumulation, recent research has explored reliable adaptation, such as using adaptive learning rates (Park et al., 2024; Maharana et al., 2025) or adaptive loss functions (Liu et al., 2024a).

**Long-term test-time adaptation.** While effective at preventing collapse in standard continual settings, TTA methods struggle under more realistic environments, such as gradual (Döbler et al., 2023) or smooth (Press et al., 2023) domain shifts that persist over the long term. To overcome these challenges, ROID (Marsden et al., 2024) introduces weight ensembling as a smooth restoration scheme, where the adapting model is updated by combining with the weighted pre-trained model. CMF (Lee & Chang, 2024) improves it by updating the pre-trained model based on the adapting model, inspired by the Kalman filter (Särkkä & Svensson, 2023). On the one hand, more aggressive alternatives have also been proposed. One such alternative is to periodically reset all parameters to their original state (Press et al., 2023). Others trigger such a reset only when extremely high predictive confidence (Niu et al., 2023) or a significant distribution discrepancy from the source (Wang et al., 2024) is identified. Another line of research has developed regularization techniques to constrain the deviation between pre-trained and adapting parameters, such as weighting regularization with Fisher information (Niu et al., 2022) or adjusting the regularization coefficient based on parameter divergence from the original state (Hoang et al., 2024). This coefficient can also be dynamically assigned for each single layer based on its location (Yang et al., 2024) or its sensitivity to distribution shifts (Choi et al., 2022). In this study, our research aligns with the emerging trend of *long-term TTA* (denoting TTA under more realistic environments where domain shifts persist over the long term), addressing the drawbacks of conventional reset mechanisms that reset too often or too rarely and completely erase the knowledge

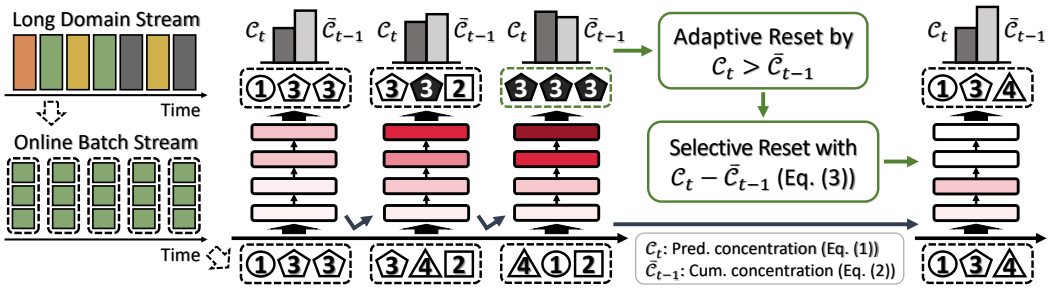


Figure 2: Overview of our **A**daptive and **S**elective **R**eset (**ASR**) scheme, which compares prediction concentration $\mathcal{C}_t$ with its cumulative counterpart $\bar{\mathcal{C}}_{t-1}$ for each test batch from a long domain stream, triggers a reset when $\mathcal{C}_t > \bar{\mathcal{C}}_{t-1}$, indicating that the model is corrupted severely enough to collapse, and determines layers to reset based on $\mathcal{C}_t - \bar{\mathcal{C}}_{t-1}$, which reflects how severely the model is corrupted. On the upper side, icons inside dashed boxes, labeled with numbers, denote class labels. White icons represent correct predictions, while black icons represent incorrect predictions.

accumulated for extended periods. However, our approach dynamically determines when and where to reset, while recovering significant knowledge lost.

# 3 Method

## 3.1 Problem Definition

Given a pre-trained source model $f_{\theta_0}$, our goal is to improve its performance at test time over a long sequence of test domains without access to source data. A handful of test samples arrive in sequence and are then inaccessible once processed via the model. At step $t$, the current model $f_{\theta_{t-1}}$ is given a test sample $x_t^i$ and generate a prediction $\hat{y}_t^i = \sigma(f_{\theta_{t-1}}(x_t^i))$, where $f_*$ yields logit outputs and $\sigma$ is the softmax function. The model is evaluated using its predictions $\hat{y}_t$, and is then adapted as $\theta_{t-1} \to \theta_t$ using unsupervised objective functions. Besides, we also aim to achieve stable adaptation, ensuring that performance does not deteriorate over time under collapse-prone scenarios such as perpetually changing or cyclically recurring domain streams. To this end, we address the limitations of existing reset approaches, such as suboptimal reset timing and the catastrophic erasure of knowledge, through the following three main components: (1) Adaptive and Selective Reset (ASR; illustrated in Fig. 2), (2) importance-aware knowledge recovery, and (3) on-the-fly adaptation adjustment.

## 3.2 Motivation

First, we observed that RDumb (Press et al., 2023)'s fixed periodic reset is only fit for standard TTA benchmarks where domain shifts occur at a regular interval. In real-world settings, however, domain shifts do not follow a fixed schedule and their timing can vary significantly. In these settings, RDumb resets either too early or too late, misaligned with the actual risk of collapse, leading to suboptimal or unstable adaptation. Second, as shown in Fig. 1, RDumb suffers from a substantial performance drop immediately after each reset. This is primarily due to its full-parameter recovery, which discards all adaptation knowledge accumulated so far, while causing significant recovery delays as well. These observations motivated our reset strategy, which triggers resets only when the model is at risk and mitigates knowledge erasure from the reset. We further support the second motivation by quantifying post-reset performance drops and recovery delays in Appendix F.1.

## 3.3 Adaptive and Selective Reset

**When to reset.** We introduce an adaptive reset scheme that triggers a reset only when a high risk of collapse is detected. To achieve it, we define prediction concentration $\mathcal{C}_t$, leveraging the notion that entropy reflects the uniformity of a distribution, where `Softmax(Mean(Logits))` serves as the underlying measure, as follows:

$$\mathcal{C}_t = \sum_{c=1}^{C} \hat{p}_{t_c} \log(\hat{p}_{t_c}) \quad \text{where} \quad \hat{p}_t = \sigma\left(\frac{1}{|\mathcal{B}_t|}\sum_{i=1}^{|\mathcal{B}_t|} f_{\theta_{t-1}}(x_t^i)\right), \tag{1}$$

$C$ is the total number of classes, and $\hat{p}_{t_c}$ indicates the probability of the $c$-th class in $\hat{p}_t$, obtained by applying the softmax function $\sigma$ to the average logits of the batch $\mathcal{B}_t$ at time step $t$. Although we can

measure the concentration of predicted classes, it remains unclear when it is high enough to suggest that the model is on the verge of collapse. We argue that when the concentration $\mathcal{C}_t$ deviates from its long-term normal behavior, it can be regarded as an indication that collapse is likely to emerge, and define cumulative concentration $\bar{\mathcal{C}}_t$, computed via exponential moving average (EMA), as follows:

$$\bar{\mathcal{C}}_t = \mu_{\mathcal{C}} \cdot \bar{\mathcal{C}}_{t-1} + (1 - \mu_{\mathcal{C}}) \cdot \mathcal{C}_t, \tag{2}$$

where $\mu_{\mathcal{C}}$ is the momentum coefficient, and $\bar{\mathcal{C}}_0$ is initialized as $-\log(\alpha_0 \cdot C)$ using a pre-defined $\alpha_0$. We compare the concentration $\mathcal{C}_t$ with its cumulative counterpart $\bar{\mathcal{C}}_{t-1}$ to judge whether to trigger a reset at each step $t$. $\bar{\mathcal{C}}_{t-1}$ is reinitialized as $-\log(\alpha_0 \cdot C)$ if the model is reset; otherwise it is updated via Eq. (2). We choose $\alpha_0$ such that the initial cumulative value is always sufficiently larger than $\mathcal{C}_t$ for any $t$ (see top-right of Fig. 2). $\bar{\mathcal{C}}_{t-1}$ is guaranteed with time to approximate the long-term normal behavior of $\mathcal{C}_t$. We render a reset triggered right after $\mathcal{C}_t > \bar{\mathcal{C}}_{t-1}$ is detected to prevent accumulating corrupted Fisher information, which will be described in Sec. 3.4. To demonstrate that our prediction concentration $\mathcal{C}_t$ is an effective metric for detecting a high collapse risk, we evaluate its correlation with accuracy in Fig. 3, where low accuracy represents a higher risk of collapse. We observe a strong Pearson correlation of 0.88, confirming the reliability of our $\mathcal{C}_t$. A detailed setup and additional analysis are provided in Appendix C.1.

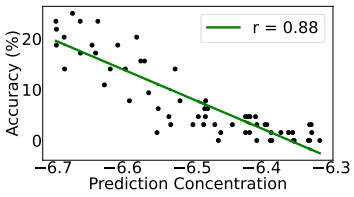


Figure 3: **Corr. of $\mathcal{C}_t$ and Acc.**

**Where to reset.** The critical drawback of reset is the catastrophic loss of knowledge acquired over time. To alleviate this, we exploit the hierarchical nature of deep neural networks. In the early stages of collapse, layers closer to the input tend to be more robust to corruption than those closer to the output, since corruption from label noise begins at the end of the network (Bai et al., 2021; Yang et al., 2024). Inspired by this insight, we propose a selective reset strategy that decides which layers to reset according to how likely the model is to collapse, prioritizing those closer to the output. Since collapse progresses with the number of corrupted layers increasing, the model facing a higher risk of collapse tends to have more corrupted layers. As a result, reset targets should scale with the risk of collapse. We can measure this risk via how far our concentration metric deviates from its normal behavior, denoted as $\mathcal{C}_t - \bar{\mathcal{C}}_{t-1}$. We define a selective reset factor $r_t$ that specifies which layers to reset, as follows:

$$r_t = r_0 + \lambda_r \cdot (\mathcal{C}_t - \bar{\mathcal{C}}_{t-1}), \tag{3}$$

where $r_0$ and $\lambda_r$ are pre-defined as the minimum size of reset targets and the risk scaling factor. The factor $r_t$ is always greater than $r_0$, as the model is reset only when $\mathcal{C}_t > \bar{\mathcal{C}}_{t-1}$, and is also subject to an upper bound of 1, indicating a full reset. It specifies target layers to reset starting from the output, such that the last $r_t$ proportion of layers are reset, while the remaining $1 - r_t$ are preserved[1].

### 3.4 Importance-Aware Knowledge Recovery

Although we attempt to mitigate the catastrophic knowledge loss from reset, some highly important knowledge is still inevitably erased. To further address this issue, we introduce an importance-aware regularizer designed to recover essential knowledge lost. At every iteration, we accumulate learnable parameters and their importance matrices computed via Fisher information (Kirkpatrick et al., 2017; Zenke et al., 2017; Schwarz et al., 2018). We then apply the regularizer to strongly guide parameters deemed significant for previous tasks toward alignment with the accumulated ones, as follows:

$$\mathcal{L}(\mathcal{B}_t; \theta_{t-1}) = \mathcal{L}_u(\mathcal{B}_t; \theta_{t-1}) + \lambda_{\mathcal{F}} \sum_{i=1}^{|\theta_{t-1}|} \bar{\mathcal{F}}^i \left(\theta^i_{t-1} - \bar{\theta}^i\right)^2, \tag{4}$$

where $\mathcal{L}$ and $\mathcal{L}_u$ are total and unsupervised losses, $\bar{\mathcal{F}}^i$ and $\bar{\theta}^i$ are the $i$-th accumulated Fisher matrix and accumulated parameter, $\theta^i_{t-1} \in \theta_{t-1}$ is the $i$-th learnable parameter from $\theta_{t-1}$, and $\lambda_{\mathcal{F}}$ is the regularization coefficient.

In the accumulation phase, the following dilemma arises: *While parameters and their Fisher matrices increasingly align with the current domain, their proximity to reset makes them more vulnerable to corruption.* Conceptually, proximity to reset indicates that, as a model has adapted for a long time, errors have also accumulated substantially, compromising its integrity and signaling that it requires

[1]For example, with 15 layers and $r_t = 0.5$, the 8 deepest layers are reset (rounded off).

a reset. We further provide empirical evidence to support it in Appendix F.2. EMA is a widely used accumulation technique, but it is not an ideal choice here, as it inherently prioritizes recent information. To address this, we propose a hybrid accumulation scheme that combines cumulative moving average (CMA) with EMA. At every iteration, CMA accumulates learnable parameters and their Fisher matrices equally. EMA then aggregates the CMA-accumulated values at each reset-triggered point, after which CMA is reinitialized to zero. The EMA-accumulated parameters and Fisher matrices correspond to $\bar{\theta}$ and $\bar{\mathcal{F}}$ in Eq. (4). More details about this knowledge accumulation scheme are provided in Appendix C.2, and its computational efficiency is analyzed in Appendix C.4. Moreover, we provide both theoretical and empirical evidence for the view that our regularizer effectively recovers essential knowledge erased by resets in Appendix E.6.

### 3.5 On-the-Fly Adaptation Adjustment

While we assume domain-evolving settings, we have not yet taken account of how evolution unfolds when designing our method. Under challenging domain shifts, our adaptability may struggle to keep pace, as resets are occasionally required. In such cases, strong guidance from the Fisher regularizer becomes crucial to exploit additional knowledge about target domains, and source–target discrepancies are also amplified, thereby worsening label noise. Pseudo-labels are more likely to be randomly assigned (Semenova et al., 2023), which complicates robust inference of prediction concentration $\mathcal{C}_t$ and renders stable updates of $\bar{\mathcal{C}}_{t-1}$ in Eq. (2) particularly challenging. To address this, we propose to adjust model adaptation on the fly based on domain discrepancy. We define prediction inconsistency $\phi_t$ to quantify domain discrepancy, as follows:

$$\phi_t = \frac{1}{|\mathcal{B}_t|} \sum_{i=1}^{|\mathcal{B}_t|} \mathbb{I}\left(\pi(\breve{y}_t^i) \neq \pi(\hat{y}_t^i)\right), \tag{5}$$

where $\mathbb{I}$ is the indicator function, $\pi$ is the argmax operation, and $\breve{y}_t^i$ and $\hat{y}_t^i$ are the softmax probabilities of the source $f_{\theta_0}$ and current $f_{\theta_{t-1}}$ models for the $i$-th test sample in $\mathcal{B}_t$, respectively. Higher $\phi_t$ values (i.e., closer to 1) indicate greater domain discrepancy. Based on this, we adjust adaptation on the fly by updating the regularization coefficient $\lambda_{\mathcal{F}}$ in Eq. (4) and the momentum coefficient $\mu_{\mathcal{C}}$ in Eq. (2) through reparameterization as follows:

$$\lambda_{\mathcal{F}} = \lambda_0 \cdot \phi_t^2, \tag{6}$$

$$\mu_{\mathcal{C}} = 1 - \mu_0 \cdot (1 - \phi_t), \tag{7}$$

where $\lambda_0$ and $\mu_0$ are pre-defined. As $\phi_t$ increases, $\lambda_{\mathcal{F}}$ grows exponentially within $[0, \lambda_0]$ for stronger regularization in Eq. (4), and $\mu_{\mathcal{C}}$ grows linearly within $[1 - \mu_0, 1]$ to minimize unstable updating of $\bar{\mathcal{C}}_{t-1}$ in Eq. (2). If $\lambda_0 = 0$, no knowledge recovery occurs; if $\mu_0 = 0$, no update of $\bar{\mathcal{C}}_{t-1}$ occurs.

## 4 Experiments

### 4.1 Setup

**Datasets.** As discussed in Press et al. (2023), standard TTA benchmarks are inadequate for validating the stability of continual TTA methods in long-term scenarios that are prone to model collapse. To address this, we adopt recently introduced benchmarks (1, 2) specifically designed for collapse, and modify the existing TTA benchmarks (3, 4) to better reflect long-term collapse-prone scenarios. We conduct experiments on the following four benchmarks: 1) **Continually Changing Corruptions (CCC)** (Press et al., 2023) is a benchmark systematically processed from ImageNet-C (Hendrycks & Dietterich, 2019). This assumes smooth domain shifts over the long term, where one fades gradually as another emerges, with the two overlapping. It is also divided into three adaptation difficulty levels (Easy / Medium / Hard), each incorporating three corruption orderings and three corruption evolving speeds, resulting in nine variations in total. 2) **Concatenated ImageNet-C (CIN-C)** is an extended version of ImageNet-C, containing 50K images per corruption, ten times larger than the original, in which 15 corruption types are sequenced under the highest corruption condition (level 5). It is often used by several studies (Wang et al., 2022; Niu et al., 2022; Gong et al., 2022; Brahma & Rai, 2023) to demonstrate their adaptation stability, while exposing collapse in Tent (Wang et al., 2021). Lastly, the following two standard TTA benchmarks, 3) **ImageNet-C (IN-C)** and 4) **ImageNet-D109 (IN-D109)** (Peng et al., 2019) are processed to reflect model collapse, following prior works (Press et al., 2023; Hoang et al., 2024). IN-C cyclically repeats the sequence of corruptions 20 times, consisting of only four types on which the source model achieves less than 10% accuracy, indicating hard-level

| Method | CCC | | | CIN-C | | IN-C | | IN-D109 | |
|---|---|---|---|---|---|---|---|---|---|
| | Easy | Medium | Hard | *i.i.d.* | *non-i.i.d.* | Visit 1 / 20 | Mean | Visit 1 / 20 | Mean |
| Source | 33.89±0.2 | 16.87±0.2 | 1.27±0.0 | 18.01±0.0 | 18.01±0.0 | 3.08 / 3.08 | 3.08±0.0 | 32.52 / 32.52 | 32.52±0.0 |
| RoTTA (CVPR'23) | 2.28±0.6 | 1.76±0.6 | 0.69±0.2 | 29.05±2.0 | 29.71±1.7 | 12.45 / 12.96 | 17.60±2.8 | 39.89 / 34.34 | 40.61±3.1 |
| ViDA (ICLR'24) | 12.68±0.8 | 5.75±0.5 | 0.42±0.0 | 17.76±0.1 | 17.76±0.1 | 3.09 / 2.84 | 2.99±0.1 | 0.01 / 0.01 | 0.01±0.0 |
| PALM (AAAI'25) | 1.56±0.2 | 0.74±0.3 | 0.13±0.0 | 12.69±6.3 | 12.08±6.1 | 24.66 / 30.98 | 30.70±1.4 | 13.86 / 1.42 | 2.06±2.7 |
| EATA (ICML'22) | 49.52±0.9 | 39.19±1.7 | 0.82±0.4 | 47.81±0.2 | 47.54±0.2 | 31.31 / 36.35 | 36.32±1.2 | 41.62 / 41.32 | 41.61±0.3 |
| + COME (ICLR'25) | 46.67±3.3 | 36.63±1.6 | 0.80±0.4 | 44.14±0.3 | 44.09±0.3 | 30.20 / 32.06 | 33.02±1.1 | 42.94 / 44.91 | 45.11±0.6 |
| CoTTA (CVPR'22) | 17.50±1.0 | 9.83±0.9 | 1.52±0.5 | 35.51±2.6 | 35.29±2.4 | 18.78 / 37.22 | 34.39±4.8 | 41.76 / 40.55 | 43.91±2.1 |
| SAR (ICLR'23) | 37.94±1.2 | 22.25±1.9 | 2.03±0.5 | 40.35±1.8 | 40.07±0.6 | 24.38 / 34.93 | 34.09±2.4 | 40.86 / 33.11 | 39.09±3.4 |
| + COME (ICLR'25) | 48.42±0.4 | 37.06±1.2 | 2.08±0.7 | 42.96±0.3 | 42.56±0.3 | 23.67 / 35.24 | 34.28±2.7 | 40.59 / 34.96 | 42.10±3.1 |
| CMF (ICLR'24) | 49.31±0.9 | 40.61±1.6 | 0.89±0.6 | 48.61±0.1 | 48.28±0.2 | 35.07 / 39.40 | 39.35±1.0 | 44.69 / 45.46 | 45.25±0.3 |
| PeTTA (NeurIPS'24) | 36.89±2.2 | 22.64±2.8 | 6.00±0.8 | 31.55±0.1 | 31.61±0.1 | 11.91 / 12.40 | 12.65±0.3 | 39.56 / 42.69 | 42.76±0.8 |
| ETA (ICML'22) | 43.24±1.0 | 19.03±6.9 | 0.32±0.1 | 43.61±0.4 | 43.63±0.4 | 30.64 / 35.80 | 35.88±1.2 | 41.24 / 34.21 | 37.22±2.1 |
| + RDumb (NeurIPS'23) | 49.47±0.8 | 39.42±1.5 | 9.77±1.8 | 46.39±0.2 | 46.13±0.2 | 30.71 / 30.94 | 34.66±2.2 | 40.93 / 41.59 | 41.45±0.4 |
| + ASR (Ours) | 51.20±0.8 | 41.88±1.6 | 17.10±2.1 | 47.17±0.2 | 46.83±0.2 | 28.68 / 39.10 | 36.90±2.9 | 40.61 / 41.32 | 41.53±0.3 |
| ROID (WACV'24) | 49.88±0.8 | 40.47±1.4 | 12.48±2.6 | 48.58±0.1 | 48.25±0.1 | 35.32 / 38.02 | 37.96±0.6 | 46.02 / 46.17 | 46.16±0.1 |
| + RDumb (NeurIPS'23) | 49.69±0.8 | 40.05±1.4 | 15.41±1.5 | 48.00±0.1 | 47.67±0.1 | 35.60 / 35.75 | 37.18±1.2 | 46.07 / 45.62 | 45.99±0.2 |
| + ASR (Ours) | **51.41±0.8** | **42.80±1.5** | **22.21±1.2** | **49.50±0.2** | **49.14±0.2** | **35.66 / 42.96** | **41.56±1.7** | **46.13 / 46.32** | **46.49±0.1** |

Table 1: Comparison with state-of-the-art continual TTA methods across four datasets using **Accuracy (%)**. Results for each level of CCC (Easy / Medium / Hard) are averaged over nine variations, considering three different corruption orderings and three corruption evolving speeds. CIN-C results are averaged over ten runs. In the *non-i.i.d.* setting, we use a Dirichlet parameter $\delta = 0.1$, following prior works (Gong et al., 2022; Yuan et al., 2023). For IN-C and IN-D109, we report averages across domains at the initial and last (20th) visits, as well as overall averages across all visits. Gray denotes model collapse, defined as performance worse than the source model (Press et al., 2023).

corruptions. IN-D109 is processed in the same way as IN-C, but it selects four hard-level corruptions according to less than 50% accuracy.

**Baselines.** We compare our approach with state-of-the-art continual TTA approaches. We categorize them into two groups based on whether they incorporate an explicit mechanism to prevent collapse. The first group, which lacks an explicit safeguard against collapse, consists of ETA (Niu et al., 2022), RoTTA (Yuan et al., 2023), ViDA (Liu et al., 2024b), C-MAE (Liu et al., 2024a), PALM (Maharana et al., 2025), and REM (Han et al., 2025). The second group, which integrates an explicit safeguard against collapse, is composed of EATA (Niu et al., 2022), CoTTA (Wang et al., 2022), RDumb (Press et al., 2023), SAR (Niu et al., 2023), ROID (Marsden et al., 2024), CMF (Lee & Chang, 2024), and PeTTA (Hoang et al., 2024). COME (Zhang et al., 2025a) does not belong to either group because it can be combined with any method using an entropy minimization objective. RDumb was originally implemented on ETA, but as a naive reset strategy, we apply it to other methods to ensure a reliable evaluation for our reset method.

**Implementation details.** We re-implement all methods in `PyTorch` (Paszke et al., 2019) within a unified TTA repository (Marsden et al., 2024), and all reported results are obtained by re-running these methods for a fair and consistent comparison. Experiments are conducted on ResNet-50 (He et al., 2016), provided by either `torchvision` or `RobustBench` (Croce et al., 2021). We also test on ViT-B-16 (Dosovitskiy et al., 2021) for CCC to further assess generalization. For ASR, we follow the implementation details of ETA (Niu et al., 2022) and ROID (Marsden et al., 2024), since we use them as our TTA baselines. We determine hyperparameters using only 5% of a holdout split (transition speed 2000, random seed 44) out of the nine available from CCC-Hard, and apply them to all datasets and settings. We also evaluate robustness to hyperparameter variations across all CCC levels in Appendix E.5. The loss $L_u$ in Eq. (4) is defined based on what our TTA baseline uses as its final loss. More details of our implementation are available in Appendix C.3. For analysis on CCC, we consistently use a single split (transition speed 2000, random seed 44).

### 4.2 Main Results

**a) CCC.** Table 1 presents the limitations of existing continual TTA methods on CCC. All methods (except for the source model) in the first row collapse across all CCC levels. Following Press et al. (2023), model collapse is defined as performance worse than the source model. Most methods in the second row achieve stable adaptation, but some fail on CCC-Hard and lack competitive performance. In the last row, RDumb (Press et al., 2023) effectively avoids collapse and further enhances ETA (Niu et al., 2022); however, it degrades ROID (Marsden et al., 2024) on CCC-Easy/-Medium. Our method demonstrates its effectiveness by achieving stable and improved performance across all baselines. It particularly achieves 22.21% (average) accuracy on the most challenging CCC-Hard, outperforming the best state-of-the-art by 44.12%. We further assess the generalization of our method on ViT-B-16

| Method | Easy | Medium | Hard | Mean |
|---|---|---|---|---|
| Source | 54.92±0.2 | 41.74±0.6 | 14.83±0.6 | 37.16±16.7 |
| CMF | 61.52±0.7 | 51.50±6.3 | 1.79±1.7 | 38.27±26.4 |
| C-MAE | 51.15±2.3 | 43.48±3.9 | 26.92±2.3 | 40.52±10.5 |
| REM | 66.16±0.3 | 57.99±0.9 | 10.97±9.9 | 45.04±25.0 |
| ETA | 45.07±10.4 | 33.71±4.6 | 1.22±0.5 | 26.67±19.7 |
| + RDumb | 59.99±0.6 | 50.50±1.4 | 23.27±1.1 | 44.58±15.6 |
| + ASR | 60.58±0.7 | 51.63±1.6 | 24.45±0.9 | 45.55±15.4 |
| ROID | 60.85±0.7 | 52.19±1.3 | 14.30±8.2 | 42.45±20.8 |
| + RDumb | 60.60±0.7 | 51.68±1.3 | 25.72±1.4 | 46.00±14.8 |
| + ASR | 61.48±0.7 | 53.55±1.3 | 28.09±0.6 | **47.71±14.3** |

Table 2: Acc. (%) comparison on **ViT**.

| $\mathcal{C}_t$ (Eq. (1)) | $r_t$ (Eq. (3)) | $\bar{\mathcal{F}}$ (Eq. (4)) | $\lambda_0$ (Eq. (6)) | $\mu_0$ (Eq. (7)) | CCC Easy | CCC Medium | CCC Hard | CCC Mean |
|---|---|---|---|---|---|---|---|---|
| ✗ | ✗ | ✗ | ✗ | ✗ | 49.74 | 40.19 | 11.81 | 33.91 |
| ✗ | ✓ | ✓ | ✓ | ✗ | 49.83 | 40.58 | 17.16 | 35.86 |
| ✓ | ✗ | ✓ | ✓ | ✓ | 49.83 | 40.35 | 15.99 | 35.39 |
| ✓ | ✓ | ✗ | ✓ | ✓ | 51.04 | 42.19 | 20.18 | 37.80 |
| ✓ | ✓ | ✓ | ✗ | ✓ | 51.07 | 42.33 | 20.27 | 37.89 |
| ✓ | ✓ | ✓ | ✓ | ✗ | 50.82 | 41.86 | 20.70 | 37.79 |
| ✓ | ✓ | ✓ | ✓ | ✓ | **51.19** | **42.42** | **21.36** | **38.32** |

Table 3: **Effect of components in ASR on ROID**.

using CCC, as reported in Table 2. We compare with baselines that have reported their performance on the ViT. While CMF (Lee & Chang, 2024) and REM (Han et al., 2025) achieve strong results on CCC-Easy and -Medium, they fail to prevent collapse on CCC-Hard. In contrast, C-MAE (Liu et al., 2024a) demonstrates its effectiveness on CCC-Hard, but does not generalize well to other levels. Our approach, however, not only maintains strong performance on CCC-Hard but also achieves the best average performance.

**b) CIN-C.** Table 1 presents results on CIN-C, reporting average accuracy over ten permutations, in which 15 corruption types are shuffled. Methods that achieve stable adaptation on CCC also perform well on CIN-C. Weight ensembling (Marsden et al., 2024; Lee & Chang, 2024), often referred to as smooth parameter restoration, demonstrates its effectiveness, achieving the top two ranks among the baselines. Our method still attains the best performance even in CIN-C that is less prone to collapse. Most existing studies assume label-i.i.d. test environments, but such assumptions do not always hold in real-world applications. Recently, increasing attention has been given to non-i.i.d. settings where labels are temporally correlated. Following Gong et al. (2022); Yuan et al. (2023), we use a Dirichlet parameter $\delta = 0.1$ to adjust the class distribution of test samples. Our method consistently improves our baselines (ETA, ROID) and achieves the best performance on ROID.

**c) IN-C.** We report average accuracy over a sequence of corruptions at the first and last (20th) visits, as well as the overall average across all visits for IN-C, as shown in Table 1. Most baselines succeed in avoiding collapse and achieve substantial improvements over the source model. IN-C is less prone to collapse; however, our method, originally designed to address such risks, also proves effective in enhancing adaptability, showing the best results consistently across the first, last, and overall visits.

**d) IN-D109.** Results for IN-D109 are reported in the same manner as for IN-C (Table 1). Several of the methods exhibit decreased performance when comparing visit 1 and 20. This indicates the early stages of collapse, which may be due to the reduced number of classes. IN-D109 contains only 109 classes, roughly ten times fewer than other datasets. Consequently, a skewed prediction distribution is more clearly observed in IN-D109 than in the other datasets. In contrast, our method demonstrates stable and superior performance on IN-D109.

### 4.3 Ablation Studies

We ablate each component from our approach to validate its individual effectiveness. Table 3 shows that dynamically determining when and where to reset is the most critical factor, as demonstrated by the first and second component-ablated results. To ablate our adaptive reset, we replace it with a fixed-interval reset scheme using $T = 20000$. In this case, $\mu_0$ is omitted as $\mathcal{C}_t$ is no longer computed. To ablate our selective reset, we adopt a full reset mechanism. The remaining components (i.e., the importance-aware regularizer and hyperparameter reparameterization) have relatively small individual impact, but when combined, they yield meaningful performance gains. When $\lambda_0$ is ablated, $\lambda_{\mathcal{F}}$ is fixed to 5.0 in Eq. (4). When $\mu_0$ is ablated, $\mu_{\mathcal{C}}$ is fixed to 0.995 in Eq. (2). More experiments for the ablation study is provided in Appendix E

### 4.4 Empirical Studies on Model Collapse

Model collapse refers to a terminal state where long-term error accumulation has severely degraded performance, eventually leading the model to predict only a few classes for all inputs. It is therefore crucial to anticipate collapse. However, it is a non-trivial task because true labels are inaccessible at test time, making such accumulation undetectable. The only reliable signal for detecting collapse is a biased prediction distribution, even though it does not hold under non-i.i.d. or imbalanced class priors. We will discuss a way to address these class priors in Sec. 4.5. `Mean(Softmax(Logits))` is the most straightforward way to measure the bias of a prediction distribution. However, what we suggest is `Softmax(Mean(Logits))`.

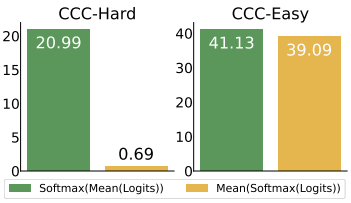


Figure 4: $\sigma(\mu)$ vs. $\mu(\sigma)$.

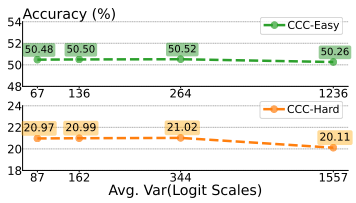


Figure 5: **Robust to Var(|logit|).**

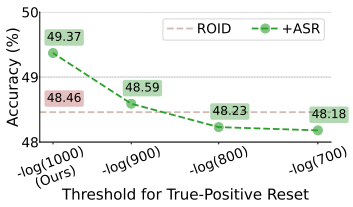


Figure 6: **False vs. True-Pos.**

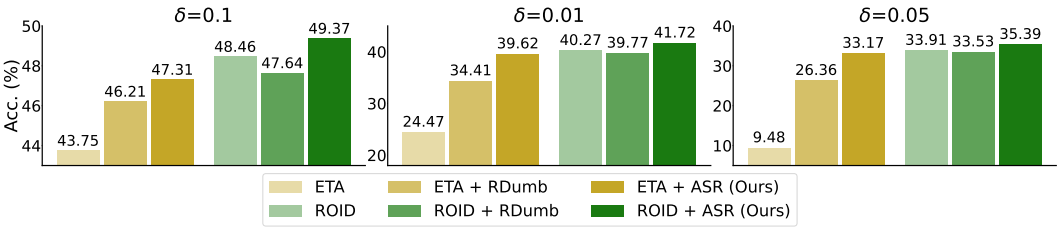


Figure 7: Comparison of ETA / ROID and its variants with RDumb and ASR over different Dirichlet parameters $\delta$ on *non-i.i.d.* CIN-C. The lower the $\delta$, the more imbalanced the label distribution.

**Q: Why is `Softmax(Mean(Logits))` effective to detect collapse?**

**A:** Models tend to update predominantly based on high-confidence predictions. Collapse is similarly driven by these predictions. Its early sign emerges when they begin to concentrate on a small subset of classes. Since large-scale logits reflect high-confidence predictions (Wei et al., 2022), averaging raw logits highlights these predictions. However, `Mean(Softmax(Logits))` normalizes logits, so it discards confidence information. In contrast, `Softmax(Mean(Logits))` is sensitive to the growing concentration of high-confidence predictions, thereby enabling more reliable detection of early collapse signs. Fig. 4 demonstrates that using `Softmax(Mean(Logits))` enables reliable adaptation in collapse-prone scenarios (e.g., CCC), whereas using `Mean(Softmax(Logits))` leads to degraded performance and fails to adapt. In the figure, the green ones represent our method with ROID, which will be described later.

**Q: Is `Softmax(Mean(Logits))` invariant to the logit-scale variance?**

**A:** We empirically verify that the logit-scale variance within a batch is not a significant concern. We adjust this variance by modifying logits within each batch as follows. For each sample, we subtract the mean of its logits to obtain deviations, scale these deviations by a factor, and then add the mean back. This scales the logit-scale variance, while preserving the logit-scale mean. As a result, large-scale logits become amplified and small-scale logits become compressed, or vice versa, depending on the factor. For this experiment, we use a single split (transition speed 1000; random seed 43) of CCC-Easy and -Hard with ROID (Marsden et al., 2024) as our base model. Fig. 5 shows that our method based on `Mean(Softmax(Logits))` is highly stable across a wide range of logit-scale variances. Even when we increase the variance by more than 15×, accuracy keeps nearly unchanged (<0.3%p on CCC-Easy and <1%p on CCC-Hard). This shows that `Mean(Softmax(Logits))` remains reliable even when logits of substantially different scales occur within a batch.

### 4.5 Risk of False-Positive Reset

One may question *"whether our method still works well under label imbalance, even though predictions are typically highly concentrated"*. The answer is that the imbalanced setting does not actually disrupt our method. As predictions are more concentrated, the cumulative prediction concentration $\bar{\mathcal{C}}_{t-1}$ rises accordingly, then a high risk of collapse is favorably captured when a much higher $\mathcal{C}_t$ is detected. We show that imbalanced class priors do not undermine our method by evaluating it under various label-imbalanced settings, as shown in Fig. 7.

Following this, one may ask *"if the much higher $\mathcal{C}_t$ could arise temporarily from extremely label-imbalanced inputs"*. In response, we argue that performing a reset at a high $\mathcal{C}_t$ is beneficial, regardless of what label distribution incoming inputs follow. Regardless of whether predictions are correct or incorrect, highly concentrated predictions produce biased update signals, ultimately leading the model to collapse. To test whether *false-positive resets*, triggered by temporarily high concentration in correctly adapting models, are beneficial, we conduct a controlled experiment under a non-i.i.d. label scenario, where such resets are common. We prepare a batch with i.i.d. labels to ensure that any triggered reset would be considered a *true-positive*. We use a single split of CIN-C (the first split in

| CCC-Easy | Original | Gain (%) | Modified | Gain (%) |
|---|---|---|---|---|
| ETA | 43.46 | - | 43.17 | - |
| + RDumb | 49.53 | +13.9% | 47.36 | +9.7% |
| + ASR (Ours) | 51.27 | +17.9% | 51.15 | +18.4% |
| ROID | 49.95 | - | 49.54 | - |
| + RDumb | 49.76 | -0.3% | 49.33 | -0.4% |
| + ASR (Ours) | 51.47 | +3.0% | 51.46 | +3.8% |

Table 4: Acc. (%) of original and modified CCC-Easy using seed 43. Gains (%) are relative to each corresponding baseline.

| CCC-Hard | Original | Gain (%) | Modified | Gain (%) |
|---|---|---|---|---|
| ETA | 0.41 | - | 1.83 | - |
| + RDumb | 9.46 | +2207% | 11.88 | +549% |
| + ASR (Ours) | 15.95 | +3790% | 17.61 | +862% |
| ROID | 9.63 | - | 16.51 | - |
| + RDumb | 14.03 | +45.6% | 15.99 | -3.1% |
| + ASR (Ours) | 21.22 | +120% | 21.56 | +30.5% |

Table 5: Acc. (%) of original and modified CCC-Hard using seed 43. Gains (%) are relative to each corresponding baseline.

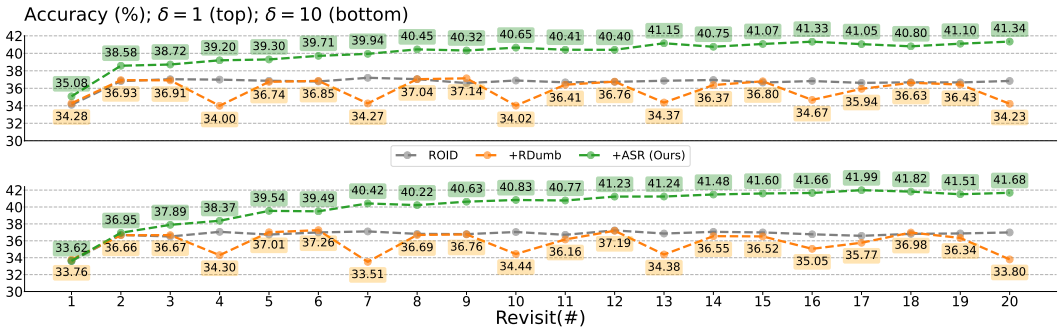


Figure 8: Performance comparison over revisits on IN-C with CDC settings.

Table D.5) with ROID as our baseline. For each reset, we compute $\mathcal{C}_t$ in Eq. (1) for that i.i.d. batch and apply a threshold to determine whether the reset is truly necessary. We initialize the threshold as described below Eq. (2). A higher threshold reduces false-positive ones, while allowing for more true-positive ones. Fig. 6 demonstrates that allowing false-positive resets (i.e., small threshold) leads to improved performance. This confirms that interrupting biased parameter updates, even when the model appears to adapt correctly, helps maintain stable long-term adaptation.

### 4.6 Dynamically Changing Corruptions: A Variant of CCC

Although we noted in our motivation (Sec. 3.2) that real-world domain shifts do not follow a fixed schedule, our benchmarks do not include varying domain-shift intervals. To better evaluate robustness under such conditions, we construct modified CCC variants, which we refer to as *Dynamically Changing Corruptions*, where the length of each corruption is randomly sampled from 1,000, 2,000, or 5,000 batches. In the original CCC setting, each corruption persists for a fixed length (e.g., always 2,000 batches). This modification introduces a stochastic corruption-transition schedule that allows us to evaluate robustness under real-world-like data streams. For a reliable evaluation, we compare results on our modified CCC variants with those on the original CCC benchmarks, as summarized in Table 4–5. In CCC-Easy, performance gains seen in the original setting are similarly reproduced in the modified setting across all methods. In contrast, CCC-Hard reveals a difference. ROID+RDumb exhibits degraded performance under the modified setting, and we conjecture that RDumb's fixed reset schedule is unable to adapt when challenging corruptions evolve unpredictably. However, our method consistently preserves performance gains, demonstrating that it adapts effectively even when corruptions are severe and evolve irregularly.

### 4.7 CDC Setting for Dynamic Domain-Shift Schedule

We demonstrate the robustness of our approach under dynamic domain shifts by applying the Continual Dynamic Change (CDC; Zhang et al. (2025b)) protocol to IN-C. This IN-C variant explicitly introduces *fast switching between domains* and *stochastic domain durations*, controlled via the Dirichlet parameter $\delta$. We evaluate our approach under both a standard CDC setting ($\delta = 1.0$) and a more dynamic setting ($\delta = 10.0$) to further emphasize its robustness. We show the results in Fig. 8. For $\delta = 1.0$, RDumb experiences repeated drops, e.g., accuracy falls from 36.91 to 34.00 at the 4th transition. In contrast, ASR steadily improves over time, rising from 35.08 to 41.34 across 20 transitions and maintaining more stable performance than RDumb. Similarly, under $\delta = 10.0$, RDumb again suffers repeated drops, whereas ASR gradually improves and remains stable, reaching 41.68 at the 20th transition. These results demonstrate that our method reliably maintains high and stable performance, even under rapid and stochastic domain shifts in real-world dynamic settings. We also provide full experimental results under CDC settings in Appendix D.5.

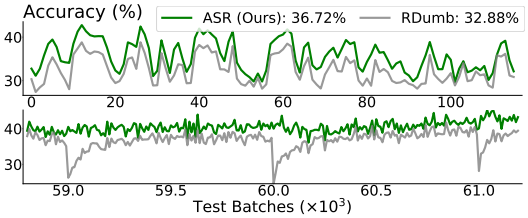


Figure 9: Comparison between ASR and RDumb on ETA using **Accuracy (%)** from a global view (top) covering 0 to 110K batches and a local view (bottom) ranging from 59K to 61K batches.

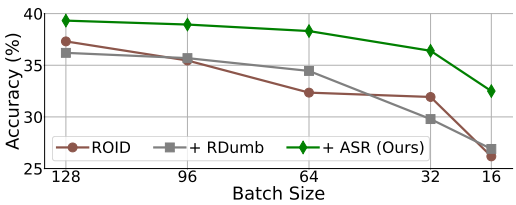


Figure 10: Accuracy (%) of ROID and its variants with RDumb and ASR over different batch sizes, averaged across all CCC levels.

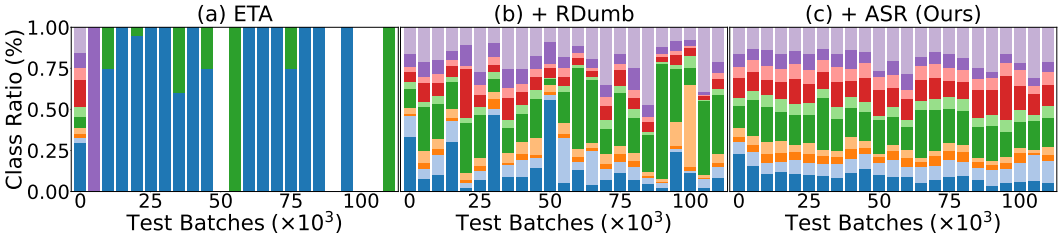


Figure 11: Histogram of predictions on CCC-Hard for ten fixed, randomly selected class labels, comparing ETA, RDumb, and ASR to evaluate robustness against model collapse. Results are measured every $10^3$ batches, with class labels color-coded consistently.

### 4.8 Analysis

**Stability analysis over time.** Beyond quantitative results, we examine whether our approach consistently maintains strong performance over time, as stabilization is crucial for reliable use in real-world applications. Fig. 9 illustrates accuracy (%) over time for ASR and RDumb on ETA[2]. For each step, we compute the average accuracy over $10^3$ batches across all CCC levels. Finally, ASR consistently outperforms RDumb from the global view (top), and the stability of ASR is demonstrated by smaller performance fluctuations from the local view (bottom).

**Robustness to batch size.** We assess the robustness of our method to batch size, as illustrated in Fig. 10. We report the average accuracy across all CCC levels, varying the batch size from 128 down to 16. As expected, performance generally decreases with smaller batch sizes. However, our method demonstrates more graceful degradation than ROID and RDumb. Moreover, in the extreme case of sequential single-sample inputs, this can be effectively addressed by stacking samples over time and adapting only when a sufficient number is obtained, following Gong et al. (2023); Niu et al. (2024). We further present results for truly small batch sizes (i.e., fewer than 16) in Appendix F.4.

**Collapse analysis.** We analyze how models are affected by collapse. Experiments are conducted on CCC-Hard under the common assumption that class labels follow a *uniform* distribution. We select ten fixed class labels and track how models generate predictions over time. ETA (Niu et al., 2022) is used as our baseline since it is highly vulnerable to collapse, allowing a clear analysis. Fig. 11 shows that ETA initially predicts a variety of classes, but its label diversity abruptly decreases afterward. It sometimes fails to assign any of the ten fixed class labels. RDumb (Press et al., 2023) helps prevent collapse, but its class distribution remains unstable and biased. In contrast, our method demonstrates superior robustness against collapse by maintaining a uniform class distribution until the end.

## 5 Conclusion

In this paper, we mitigate model collapse in long-term TTA via Adaptive and Selective Reset (ASR), combined with importance-aware knowledge recovery and on-the-fly adaptation adjustment. Experimental results demonstrate the effectiveness of our proposed method across long-term TTA benchmarks, particularly in challenging settings. Specifically, our method outperforms the state-of-the-art by 44.12% on CCC-Hard. We hope that our work motivates further exploration into advanced reset mechanisms for long-term TTA, aiming at robust and stable adaptation while preventing collapse.

[2]Two methods are identical at $t = 0$, but the initial point in Fig. 9 (top) denotes the average over $t \in [0, 999]$.

## REFERENCES


Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. Understanding and improving early stopping for learning with noisy labels. In *NeurIPS*, 2021.

Dhanajit Brahma and Piyush Rai. A probabilistic framework for lifelong test-time adaptation. In *CVPR*, 2023.

Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *ECCV*, 2022.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

Mario Döbler, Robert A Marsden, and Bin Yang. Robust mean teacher for continual and gradual test-time adaptation. In *CVPR*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Yulu Gan, Yan Bai, Yihang Lou, Xianzheng Ma, Renrui Zhang, Nian Shi, and Lin Luo. Decorate the newcomers: Visual domain prompt for continual test time adaptation. In *AAAI*, 2023.

Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. Note: Robust continual test-time adaptation against temporal correlation. In *NeurIPS*, 2022.

Taesik Gong, Yewon Kim, Taeckyung Lee, Sorn Chottananurak, and Sung-Ju Lee. Sotta: Robust test-time adaptation on noisy data streams. In *NeurIPS*, 2023.

Sachin Goyal, Mingjie Sun, Aditi Raghunanthan, and Zico Kolter. Test-time adaptation via conjugate pseudo-labels. In *NeurIPS*, 2022.

Jisu Han, Jaemin Na, and Wonjun Hwang. Ranked entropy minimization for continual test-time adaptation. In *ICML*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Trung-Hieu Hoang, Duc Minh Vo, and Minh N Do. Persistent test-time adaptation in recurring testing scenarios. In *NeurIPS*, 2024.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

Jogendra Nath Kundu, Naveen Venkat, Rahul M V, and R. Venkatesh Babu. Universal source-free domain adaptation. In *CVPR*, 2020.

Jae-Hong Lee and Joon-Hyuk Chang. Continual momentum filtering on parameter space for online test-time adaptation. In *ICLR*, 2024.

Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *CVPR*, 2020.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020.

Jiaming Liu, Ran Xu, Senqiao Yang, Renrui Zhang, Qizhe Zhang, Zehui Chen, Yandong Guo, and Shanghang Zhang. Continual-mae: Adaptive distribution masked autoencoders for continual test-time adaptation. In *CVPR*, 2024a.

Jiaming Liu, Senqiao Yang, Peidong Jia, Renrui Zhang, Ming Lu, Yandong Guo, Wei Xue, and Shanghang Zhang. Vida: Homeostatic visual domain adapter for continual test time adaptation. In *ICLR*, 2024b.

Sarthak Kumar Maharana, Baoming Zhang, and Yunhui Guo. Palm: Pushing adaptive learning rate mechanisms for continual test-time adaptation. In *AAAI*, 2025.

Robert A Marsden, Mario Döbler, and Bin Yang. Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction. In *WACV*, 2024.

M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *CVPR*, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, 2022.

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

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

Junyoung Park, Jin Kim, Hyeongjun Kwon, Ilhoon Yoon, and Kwanghoon Sohn. Layer-wise auto-weighting for non-stationary test-time adaptation. In *WACV*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 1992.

Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. Rdumb: A simple approach that questions our progress in continual test-time adaptation. In *NeurIPS*, 2023.

Simo Särkkä and Lennart Svensson. Bayesian filtering and smoothing. *Cambridge university press*, 2023.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018.

Lesia Semenova, Harry Chen, Ronald Parr, and Cynthia Rudin. A path to simpler models starts with noise. In *NeurIPS*, 2023.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 2024.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

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

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022.

Ziqiang Wang, Zhixiang Chi, Yanan Wu, Li Gu, Zhi Liu, Konstantinos Plataniotis, and Yang Wang. Distribution alignment for fully test-time adaptation with dynamic online data streams. In *ECCV*, 2024.

Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization. In *ICML*, 2022.

Xu Yang, Xuan Chen, Moqi Li, Kun Wei, and Cheng Deng. A versatile framework for continual test-time domain adaptation: Balancing discriminability and generalizability. In *CVPR*, 2024.

Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *CVPR*, 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *NeurIPS*, 2022.

Qingyang Zhang, Yatao Bian, Xinke Kong, Peilin Zhao, and Changqing Zhang. Come: Test-time adaption by conservatively minimizing entropy. In *ICLR*, 2025a.

Yunbei Zhang, Akshay Mehra, Shuaicheng Niu, and Jihun Hamm. Dpcore: Dynamic prompt coreset for continual test-time adaptation. In *ICML*, 2025b.

# APPENDICES

# A DISCUSSIONS

**Q: Does your method rely on incremental and heuristic solutions for long-term TTA?**

**A:** Our method is not a collection of small fixes. We reframe long-term TTA through a reset-based view, in which preventing collapse is considered as *a continuous decision-making task* rather than following a fixed schedule. Prior work typically adopts resets at fixed intervals (Press et al., 2023) or only after collapse occurs (Niu et al., 2023). In contrast, our approach continuously estimates the risk of collapse. Moreover, we integrate several components (Sec. 3.3–3.4) under a single principle: *balancing the forgetting and retaining of knowledge*. This unified framing has not been explored in prior TTA research. We describe how these components work together in Appendix E.6.

**Q: Does your method fail to overcome the need for reset in long-term TTA?**

**A:** Reset is an essential and widely recognized mechanism to prevent collapse in long-term TTA. Neural networks typically converge to sharp minima, making it difficult to escape and find better solutions through standard gradient updates (Keskar et al., 2017). Collapse is an even more challenging state than a sharp minimum, making recovery *nearly impossible* without reset (Hoang et al., 2024). Despite its importance, reset has been largely unexplored: existing approaches simply adopt resets at fixed intervals with full-parameter recovery. We tackle these fundamental limitations, effectively exploring the potential of reset and proposing a strategy that dynamically adjusts both its timing and extent based on the model's state.

**Q: Are the marginal gains worth the engineering effort, or would simpler variants suffice?**

**A:** Designed to tackle model collapse in long-term TTA, our method is highly effective in challenging and realistic scenarios. CCC-Hard best reflects such scenarios, where we achieve a substantial 44.12% improvement over the state of the art, demonstrating that our approach effectively handles difficult tasks. In contrast, other benchmarks, such as IN-C or IN-D109, are easier, and the modest improvements are what any method could achieve in such simple settings. This shows that the smaller gains on easy tasks do not imply that simpler variants would be sufficient for the more challenging benchmarks. As more benchmarks prone to collapse are available, we expect the benefits of our approach to become even clearer.

## B More Details on Datasets

In this paper, we evaluate the stable adaptability of continual TTA methods across the following four benchmarks known for their susceptibility to collapse.

**1) Continually Changing Corruptions (CCC)** is introduced by RDumb (Press et al., 2023), which is systematically processed using the ImageNet-C dataset. This converts ImageNet-C's abrupt corruption transitions into smooth ones by interpolating integer corruption levels (1–5) to floating-point values between 0 and 5 in steps of 0.25, where one fades gradually (e.g., $1 \rightarrow 0$) as another emerges (e.g., $0 \rightarrow 1$), with the two overlapping. A smooth transition path consists of two key aspects: levels at which two corruptions start, and how they gradually fade and emerge. They are determined by the source model's accuracy (0% / 20% / 40%), which reflects the adaptation difficulty (Easy / Medium / Hard). Different corruption types are incorporated into each level's path, as reported in Table B.1. A transition speed, defined as the number of images per step along the path, has three variations (1000 / 2000 / 5000), and a corruption ordering also has three variations, determined randomly using seeds (43 / 44 / 45). CCC contains 7.5M images for each combination of path, speed, and ordering. Lastly, CCC incorporates widely-recognized contributors to model collapse, including long-term corruption transitions (Wang et al., 2022), consistent adaptation difficulty across corruptions (Press et al., 2023), and repeated corruption occurrences (Hoang et al., 2024).

| Level | Corruption Types |
|---|---|
| Easy | `Gaussian_noise, Shot_noise, Impulse_noise, Contrast` |
| Medium | `Gaussian_noise, Shot_noise, Impulse_noise, Defocus_blur, Glass_blur, Motion_blur, Zoom_blur, Snow, Frost, Fog, Contrast, Elastic, Pixelate` |
| Hard | `Gaussian_noise, Shot_noise, Impulse_noise, Defocus_blur, Glass_blur, Motion_blur, Zoom_blur, Snow, Frost, Fog, Contrast, Elastic, Pixelate, JPEG` |

Table B.1: Corruption types per smooth transition path for each level of adaptation difficulty.

**2) Concatenated ImageNet-C (CIN-C)** consists of image samples from the ImageNet-C validation set with 15 corruption types—*Gaussian noise*, *Shot noise*, *Impulse noise*, *Defocus blur*, *Glass blur*, *Motion blur*, *Zoom blur*, *Snow*, *Frost*, *Fog*, *Contrast*, *Brightness*, *Elastic*, *Pixelate*, *JPEG*—at the highest severity (level 5). CIN-C contains 50K images for each corruption type, which is totally ten times larger than the original set.

**3) ImageNet-C (IN-C)** is processed to evaluate stability against model collapse. It consists of only four corruption types at the highest severity (level 5), including *Gaussian noise*, *Shot noise*, *Impulse noise*, *Contrast*, for which the source model achieves less than 10% accuracy, ensuring consistent adaptation difficulty across corruptions. Each type contains 5K images, and IN-C contains a total of 400K images by repeating the corruption sequence 20 times, satisfying another known contributor to model collapse. Finally, IN-C uses the following ordering: *Gaussian noise* $\rightarrow$ *Shot noise* $\rightarrow$ *Impulse noise* $\rightarrow$ *Contrast*.

**4) ImageNet-D109 (IN-D109)** is also processed to evaluate stability against collapse. It consists of only four domains—*Clipart*, *Infograph*, *Painting*, *Sketch*—out of six available, for which the source model achieves less than 50% accuracy, ensuring consistent adaptation difficulty across domains. It uses the ordering of the domain sequence as *Clipart* $\rightarrow$ *Infograph* $\rightarrow$ *Painting* $\rightarrow$ *Sketch*, and repeats the sequence 20 times to account for another key contributor to collapse. Finally, it has only classes that are shared with the DomainNet dataset, resulting in 109 classes.

# C Additional Details of ASR

## C.1 More Details on Prediction Concentration

To compute the correlation in Fig. 3, we use ETA as a TTA model and CCC-Hard as a benchmark, because they exhibits explicit collapse and are therefore suitable for demonstrating the link between collapse and prediction concentration $\mathcal{C}_t$. Moreover, Fig. 3 does not include temporal information, so points corresponding to single batches toward the right do not represent later adaptation steps.

One may question that *"could the pattern in Fig. 3 be an artifact of logit averaging from Eq. (1)?"* To address it, we measure prediction concentration $\mathcal{C}_t$ after excluding the largest-scale logit in each batch, and also measure it after excluding the top 10% of logits by scale. We compute their Pearson correlations, as shown in Table C.1. Although slightly lower than the original value of 0.88 (refer to Fig. 3), the variant values of 0.85 and 0.77 are also meaningful. As a result, the effect of extremely large-scale logits is minimal, and the pattern in Fig. 3 cannot be attributed entirely to an artifact. In addition, as a model approaches collapse, its predictions assign increasingly large logit values to a few dominant classes, causing the overall logit scale to grow as well. Consequently, the pattern in Fig. 3 reflects contributions from many logits, not just a few extreme ones.

| Excluded logits | Pearson correlation |
|---|---|
| None | 0.88 |
| Top-1 | 0.85 |
| Top-10% | 0.77 |

Table C.1: Effect of large-scale logits on the correlation in Fig. 3.

## C.2 More Details on Knowledge Accumulation

We achieve knowledge recovery by guiding parameters through regularization using their accumulated values and importance, as described in Sec. 3.4. Moreover, particular caution is required during the accumulation phase, as a trade-off exists: achieving better representations for the current domain comes at the cost of increased vulnerability to corruption, as errors accumulate over time. To address this, we propose a hybrid accumulation strategy that combines cumulative moving average (CMA) with exponential moving average (EMA). First of all, at every iteration, we accumulate the squared loss derivatives with respect to each parameter, $\left(\nabla_{\theta^i_{t-1}}\mathcal{L}(\mathcal{B}_t;\theta_{t-1})\right)^2$, defined as the diagonal of the Fisher information matrix, as well as learnable parameters $\theta^i_{t-1}$ via CMA, as follows:

$$\tilde{\mathcal{F}}^i_t = \frac{(t-1-t^*_{\text{latest}})\cdot\tilde{\mathcal{F}}^i_{t-1} + \left(\nabla_{\theta^i_{t-1}}\mathcal{L}(\mathcal{B}_t;\theta_{t-1})\right)^2}{t-t^*_{\text{latest}}}, \tag{C.1}$$

$$\tilde{\theta}^i_t = \frac{(t-1-t^*_{\text{latest}})\cdot\tilde{\theta}^i_{t-1} + \theta^i_{t-1}}{t-t^*_{\text{latest}}}, \tag{C.2}$$

where $t^*_{\text{latest}}$ is the latest step of reset prior to step $t$, and $\tilde{\mathcal{F}}^i_t$ and $\tilde{\theta}^i_t$ represent the CMA-accumulated Fisher matrix and parameter for the $i$-th parameter $\theta^i$, both initialized to zero at $t=0$. We then accumulate the CMA-accumulated Fisher matrices and parameters via EMA at each reset, as follows:

$$\bar{\mathcal{F}}^i \leftarrow \mu_{\mathcal{F}}\cdot\bar{\mathcal{F}}^i + (1-\mu_{\mathcal{F}})\cdot\tilde{\mathcal{F}}^i_t, \tag{C.3}$$

$$\bar{\theta}^i \leftarrow \mu_{\theta}\cdot\bar{\theta}^i + (1-\mu_{\theta})\cdot\tilde{\theta}^i_t, \tag{C.4}$$

where $\mu_{\mathcal{F}}$ and $\mu_{\theta}$ are the momentum coefficients, both of which are pre-defined as 0.9, and $\bar{\mathcal{F}}^i$ and $\bar{\theta}^i$ are the EMA-accumulated Fisher matrix and parameter for the $i$-th parameter $\theta^i$, both initialized to zero. After the EMA update, $\tilde{\mathcal{F}}^i_t$ and $\tilde{\theta}^i_t$ are reinitialized to zero.

### C.3 Implementation Details

Detailed hyperparameters are listed in Table C.2.

| Hyperparameter | Description | Reference | ResNet-50 | ViT-B-16 |
|---|---|---|---|---|
| $\alpha_0$ | Initialization factor for $\bar{\mathcal{C}}_{t-1}$ | Below Eq. (2) (Sec. 3.3) | 0.5 | $5.0 \times 10^{-4}$ |
| $\mu_{\mathcal{C}}$ | EMA update momentum for $\bar{\mathcal{C}}_{t-1}$ | Eq. (2) (Sec. 3.3) | 0.995 | 0.995 |
| $r_0$ | Minimum reset proportion | Eq. (3) (Sec. 3.3) | 0.5 | 0.5 |
| $\lambda_r$ | Reset proportion scaling factor | Eq. (3) (Sec. 3.3) | 20.0 | 0.1 |
| $\lambda_{\mathcal{F}}$ | Fisher regularization coefficient | Eq. (4) (Sec. 3.4) | 5.0 | 5.0 |
| $\lambda_0$ | Initialization factor for $\lambda_{\mathcal{F}}$ | Eq. (6) (Sec. 3.5) | 5.0 | 5.0 |
| $\mu_0$ | Initialization factor for $\mu_{\mathcal{C}}$ | Eq. (7) (Sec. 3.5) | 0.15 | $1.0 \times 10^{-3}$ |

Table C.2: Hyperparameters used for ResNet-50 and ViT-B-16 across all benchmarks.

### C.4 Computational Efficiency

Table C.3 compares baselines, ASR and its ablations in terms of # trainable/total parameters, computation time (secs per batch) and average accuracy (%) across all CCC levels. Parameter restoration methods (i.e., ROID, RDumb, and ASR) double the memory to retain the initial state, and the additional cost for our extra parameters (mostly Fisher information) is negligible compared to a total model size of 25.5M. Specifically, $\bar{\theta}$ and $\tilde{\theta}$ have a size of $|\theta|$, respectively. Each of $\bar{\mathcal{F}}$ and $\tilde{\mathcal{F}}$ also has a size of $|\theta|$, as they store only the diagonal elements of the Fisher matrix, following the standard practice in Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017). This indicates that each of the four occupies just 0.025M parameters (i.e., 0.098% of the total). Regarding the computational cost, Fisher information is computed once per batch (with size 64), adding only less than 0.001s per batch. Therefore, the computation and memory overhead of our extra parameters is minimal, making our method highly efficient in practice.

| Method | # Trainable | # Param | Time | Acc. |
|---|---|---|---|---|
| ETA | 53.1K | 25.5M | .083 | 21.72 |
| ROID | 53.1K | 51.1M | .125 | 33.91 |
| + RDumb | 53.1K | 51.1M | .125 | 35.39 |
| + ASR (Ours) | 53.1K | 51.2M | .200 | 38.32 |
| + w/o recovery (Sec. 3.4) | 53.1K | 51.1M | .200 | 37.80 |
| + w/o on-the-fly (Sec. 3.5) | 53.1K | 51.2M | .181 | 37.89 |

Table C.3: Computational analysis on CCC. # Trainable denotes the number of learnable parameters; # Param denotes the total number of parameters; Time denotes seconds per batch of 64 samples; and Acc. denotes the average accuracy (%) across all CCC levels.

### C.5 ASR UNDER ABRUPT DOMAIN CHANGES

Gan et al. (2023) find that prediction confidence rapidly changes along with domain shifts. Similarly, we observe that prediction concentration exhibits abrupt dynamics together with domain changes, as illustrated in Fig. C.1. Since ASR relies on prediction concentration, we check whether such abrupt behavior negatively impacts it. An abrupt decline in prediction concentration may be interpreted as random predictions. In reality, it is not severe enough to cause such predictions. However, an abrupt rise in prediction concentration often results in $\mathcal{C}_t > \bar{\mathcal{C}}_{t-1}$, thereby unintentionally triggering a reset. Zhang et al. (2025b) point out that negative knowledge transfer may occur along with a domain shift and should thus be addressed. In this regard, such unintended resets can serve as a safeguard against this transfer. Finally, the abrupt dynamics of prediction concentration along with domain shifts pose no risk of disrupting ASR.

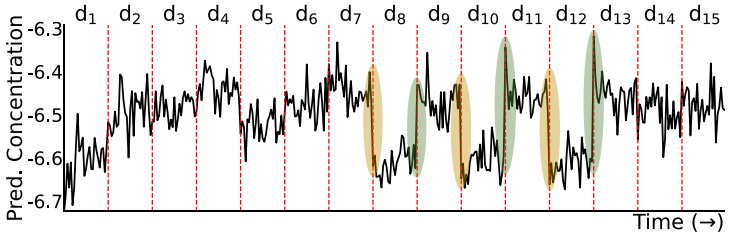


Figure C.1: Prediction concentration (Eq. (1)) over time under fifteen corruptions in CIN-C. Dashed vertical lines (Red) denote corruption (domain) boundaries. Colored ellipses indicate abrupt dynamics along with domain shifts ( Yellow : abrupt decline, Green : abrupt rise).

### C.6 ALGORITHM

The complete ASR workflow is presented in Algorithm 1.

**Algorithm 1:** Adaptive and Selective Reset (ASR)

**Input:** Test batches $\{\mathcal{B}_t\}_{t=1}^{T}$, adapting model $f_{\theta_*}$, source model $f_{\theta_0}$, cumulative concentration initialization factor $\alpha_0$, regularization coefficient initialization factor $\lambda_0$, EMA update momentum initialization factor $\mu_0$, minimum reset proportion $r_0$, reset proportion scaling factor $\lambda_r$, and EMA update momentums $\{\mu_{\mathcal{F}}, \mu_\theta\}$.

Initialize $\bar{\mathcal{C}}_0 \leftarrow -\log(\alpha_0 \cdot C)$, $\tilde{\mathcal{F}}_0 \leftarrow 0$ and $\tilde{\theta}_0 \leftarrow 0$;

**for** $t \in \{1, \ldots, T\}$ **do**

- `// 1) Model Adaptation`
- Generate logits $z_t = f_{\theta_{t-1}}(\mathcal{B}_t)$;
- Compute loss $\mathcal{L}(\mathcal{B}_t; \theta_{t-1})$ in Eq. (4);
- `//  CMA-based Knowledge Accumulation`
- Update $\tilde{\mathcal{F}}_t$ and $\tilde{\theta}_t$ via Eq. (C.1) and Eq. (C.2);
- Update $\theta_t \leftarrow \underset{\theta_{t-1}}{\texttt{Optim}}\, \mathcal{L}(\mathcal{B}_t; \theta_{t-1})$;
- `// 2) On-the-fly Adaptation Adjustment`
- Compute prediction inconsistency $\phi_t = \frac{1}{|\mathcal{B}_t|} \sum_{i=1}^{|\mathcal{B}_t|} \mathbb{I}\left(\pi(\breve{y}_t^i) \neq \pi(\hat{y}_t^i)\right)$
  where $\pi(\breve{y}_t^i) = \text{argmax}_c[\sigma(f_{\theta_0}(x_t^i))]_c$ and $\pi(\hat{y}_t^i) = \text{argmax}_c[\sigma(z_t^i)]_c$;
- Adjust regularization coefficient $\lambda_{\mathcal{F}} = \lambda_0 \cdot \phi_t^2$
  and momentum coefficient $\mu_{\mathcal{C}} = 1 - \mu_0 \cdot (1 - \phi_t)$;
- `// 3) Adaptive and Selective Reset`
- Compute prediction concentration $\mathcal{C}_t = \sum_{c=1}^{C} \hat{p}_{t_c} \log(\hat{p}_{t_c})$ where $\hat{p}_t = \sigma\left(\frac{1}{|\mathcal{B}_t|} \sum_{i=1}^{|\mathcal{B}_t|} z_t^i\right)$;
- **if** $\mathcal{C}_t - \bar{\mathcal{C}}_{t-1} \leq 0$ **then**
  - Update $\bar{\mathcal{C}}_t \leftarrow \mu_{\mathcal{C}} \cdot \bar{\mathcal{C}}_{t-1} + (1 - \mu_{\mathcal{C}}) \cdot \mathcal{C}_t$;
- **end**
- **else**
  - Compute selective reset factor $r_t = r_0 + \lambda_r \cdot (\mathcal{C}_t - \bar{\mathcal{C}}_{t-1})$ where $r_t \in [r_0, 1]$;
  - Reset only the last $r_t$ proportion of total layers;
  - Initialize $\bar{\mathcal{C}}_t \leftarrow -\log(\alpha_0 \cdot C)$;
  - `//  EMA-based Knowledge Accumulation`
  - Update $\bar{\mathcal{F}} \leftarrow \texttt{EMA}(\bar{\mathcal{F}}, \tilde{\mathcal{F}}_t, \mu_{\mathcal{F}})$ in Eq. (C.3);
  - Update $\bar{\theta} \leftarrow \texttt{EMA}(\bar{\theta}, \tilde{\theta}_t, \mu_\theta)$ in Eq. (C.4);
  - Initialize $\tilde{\mathcal{F}}_t \leftarrow 0$ and $\tilde{\theta}_t \leftarrow 0$;
- **end**

**end**

# D ADDITIONAL RESULTS

## D.1 FULL RESULTS ON RESNET

In Tables D.1–D.7, we present the full evaluation results on ResNet-50, extending Table 1.

| Transition speed | 1000 | | | 2000 | | | 5000 | | | Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Corruption ordering | 43 | 44 | 45 | 43 | 44 | 45 | 43 | 44 | 45 | Mean |
| Source | 33.89 | 33.97 | 33.95 | 33.69 | 33.90 | 33.99 | 33.34 | 34.06 | 34.23 | 33.89±0.2 |
| RMT (CVPR'23) | 48.15 | 46.78 | 47.38 | 46.70 | 46.44 | 47.80 | 48.61 | 45.03 | 48.07 | 47.22±1.0 |
| RoTTA (CVPR'23) | 1.76 | 1.49 | 1.88 | 2.43 | 1.84 | 2.28 | 2.50 | 3.06 | 3.28 | 2.28±0.6 |
| SANTA (TMLR'23) | 47.33 | 47.47 | 47.49 | 47.87 | 47.68 | 47.77 | 48.32 | 47.11 | 48.10 | 47.68±0.4 |
| LAW (WACV'24) | 2.71 | 2.50 | 2.92 | 2.99 | 2.44 | 3.25 | 2.79 | 2.20 | 3.55 | 2.82±0.4 |
| ViDA (ICLR'24) | 13.52 | 12.28 | 12.74 | 13.68 | 12.32 | 11.89 | 13.81 | 12.29 | 11.61 | 12.68±0.8 |
| DPLOT (CVPR'24) | 36.68 | 35.71 | 35.84 | 36.18 | 34.02 | 35.79 | 33.55 | 32.94 | 34.61 | 35.04±1.2 |
| PALM (AAAI'25) | 1.55 | 1.34 | 1.66 | 1.67 | 1.29 | 1.70 | 1.84 | 1.23 | 1.73 | 1.56±0.2 |
| EATA (ICML'22) | 48.53 | 48.65 | 48.48 | 49.52 | 49.47 | 49.35 | 51.00 | 50.07 | 50.64 | 49.52±0.9 |
| + COME (ICLR'25) | 46.99 | 47.04 | 47.00 | 37.50 | 47.72 | 47.63 | 49.11 | 48.26 | 48.80 | 46.67±3.3 |
| CoTTA (CVPR'22) | 17.01 | 15.98 | 16.24 | 18.05 | 17.02 | 17.13 | 19.33 | 18.10 | 18.60 | 17.50±1.0 |
| SAR (ICLR'23) | 36.65 | 36.24 | 36.47 | 39.21 | 37.55 | 38.75 | 39.92 | 37.84 | 38.83 | 37.94±1.2 |
| + COME (ICLR'25) | 47.99 | 48.16 | 48.01 | 48.57 | 48.36 | 48.25 | 49.23 | 48.32 | 48.87 | 48.42±0.4 |
| PETAL (CVPR'23) | 2.57 | 2.52 | 2.64 | 2.62 | 2.54 | 2.71 | 2.66 | 2.43 | 2.64 | 2.59±0.1 |
| CMF (ICLR'24) | 48.29 | 48.33 | 48.20 | 49.38 | 49.25 | 49.12 | 50.87 | 49.95 | 50.40 | 49.31±0.9 |
| DATTA (ECCV'24) | 9.87 | 18.26 | 23.48 | 28.49 | 24.46 | 20.79 | 25.26 | 29.36 | 23.65 | 22.62±5.5 |
| PeTTA (NeurIPS'24) | 34.61 | 34.43 | 34.56 | 36.45 | 36.26 | 36.40 | 40.43 | 38.90 | 40.01 | 36.89±2.2 |
| ETA (ICML'22) | 42.13 | 42.23 | 42.12 | 43.46 | 43.13 | 42.87 | 45.25 | 43.86 | 44.07 | 43.24±1.0 |
| + RDumb (NeurIPS'23) | 48.55 | 48.57 | 48.49 | 49.53 | 49.42 | 49.35 | 50.79 | 49.97 | 50.57 | 49.47±0.8 |
| + ASR (Ours) | 50.33 | 50.31 | 50.13 | 51.27 | 51.12 | 50.92 | 52.73 | 51.78 | 52.21 | 51.20±0.8 |
| ROID (WACV'24) | 49.02 | 49.03 | 48.92 | 49.95 | 49.81 | 49.74 | 51.15 | 50.37 | 50.94 | 49.88±0.8 |
| + RDumb (NeurIPS'23) | 48.82 | 48.85 | 48.74 | 49.76 | 49.63 | 49.56 | 50.91 | 50.15 | 50.75 | 49.69±0.8 |
| + ASR (Ours) | 50.50 | 50.58 | 50.42 | 51.47 | 51.36 | 51.19 | 52.86 | 51.94 | 52.35 | 51.41±0.8 |

Table D.1: Performance comparison with state-of-the-art methods on **CCC-Easy**, containing nine variations with three corruption transition speeds (1000 / 2000 / 5000) and three corruption orderings determined by random seeds (43 / 44 / 45).

| Transition speed | 1000 | | | 2000 | | | 5000 | | | Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Corruption ordering | 43 | 44 | 45 | 43 | 44 | 45 | 43 | 44 | 45 | Mean |
| Source | 16.95 | 16.78 | 16.95 | 16.59 | 16.87 | 16.97 | 16.57 | 16.91 | 17.20 | 16.87±0.2 |
| RMT (CVPR'23) | 35.48 | 35.38 | 35.60 | 36.07 | 35.65 | 34.08 | 35.41 | 31.42 | 36.09 | 35.02±1.4 |
| RoTTA (CVPR'23) | 1.23 | 1.00 | 1.36 | 1.84 | 1.31 | 1.70 | 2.76 | 2.08 | 2.54 | 1.76±0.6 |
| SANTA (TMLR'23) | 33.75 | 33.77 | 34.17 | 35.65 | 34.18 | 34.26 | 35.94 | 33.79 | 34.57 | 34.45±0.8 |
| LAW (WACV'24) | 1.56 | 1.09 | 1.50 | 1.66 | 0.64 | 1.57 | 1.38 | 0.76 | 1.57 | 1.30±0.4 |
| ViDA (ICLR'24) | 6.16 | 6.10 | 6.20 | 5.73 | 6.19 | 5.78 | 4.95 | 5.65 | 5.01 | 5.75±0.5 |
| DPLOT (CVPR'24) | 14.64 | 10.70 | 18.70 | 12.05 | 7.58 | 18.07 | 9.50 | 6.83 | 20.08 | 13.13±4.7 |
| PALM (AAAI'25) | 0.76 | 0.50 | 0.98 | 0.63 | 0.37 | 1.47 | 0.51 | 0.62 | 0.83 | 0.74±0.3 |
| EATA (ICML'22) | 37.36 | 36.91 | 37.40 | 39.98 | 38.66 | 38.80 | 41.52 | 40.47 | 41.58 | 39.19±1.7 |
| + COME (ICLR'25) | 34.81 | 34.63 | 35.02 | 37.47 | 36.03 | 36.15 | 38.86 | 37.87 | 38.79 | 36.63±1.6 |
| CoTTA (CVPR'22) | 9.62 | 8.65 | 9.10 | 10.04 | 9.06 | 10.30 | 10.56 | 9.52 | 11.63 | 9.83±0.9 |
| SAR (ICLR'23) | 19.98 | 21.20 | 20.01 | 23.25 | 20.91 | 21.19 | 23.89 | 24.91 | 24.89 | 22.25±1.9 |
| + COME (ICLR'25) | 35.75 | 35.99 | 35.48 | 37.95 | 36.75 | 36.58 | 38.28 | 37.68 | 39.11 | 37.06±1.2 |
| PETAL (CVPR'23) | 2.14 | 1.92 | 2.18 | 2.06 | 1.84 | 2.18 | 2.02 | 1.69 | 1.98 | 2.00±0.2 |
| CMF (ICLR'24) | 38.77 | 38.41 | 38.79 | 41.32 | 40.28 | 40.28 | 42.84 | 41.93 | 42.85 | 40.61±1.6 |
| DATTA (ECCV'24) | 9.42 | 10.96 | 9.19 | 12.78 | 13.28 | 12.85 | 19.46 | 14.37 | 17.49 | 13.31±3.2 |
| PeTTA (NeurIPS'24) | 19.34 | 19.30 | 19.65 | 23.41 | 21.92 | 22.13 | 27.34 | 25.19 | 25.47 | 22.64±2.8 |
| ETA (ICML'22) | 22.28 | 13.05 | 18.40 | 25.36 | 17.55 | 22.01 | 20.87 | 3.37 | 28.41 | 19.03±6.9 |
| + RDumb (NeurIPS'23) | 37.72 | 37.45 | 37.83 | 40.21 | 39.05 | 39.15 | 41.55 | 40.46 | 41.34 | 39.42±1.5 |
| + ASR (Ours) | 40.10 | 39.78 | 40.04 | 42.34 | 41.47 | 41.49 | 44.13 | 43.35 | 44.25 | 41.88±1.6 |
| ROID (WACV'24) | 38.79 | 38.64 | 38.91 | 41.24 | 40.16 | 40.19 | 42.44 | 41.44 | 42.41 | 40.47±1.4 |
| + RDumb (NeurIPS'23) | 38.42 | 38.26 | 38.56 | 40.85 | 39.75 | 39.77 | 42.00 | 40.94 | 41.90 | 40.05±1.4 |
| + ASR (Ours) | 41.13 | 40.91 | 41.20 | 43.40 | 42.49 | 42.42 | 44.77 | 43.98 | 44.91 | 42.80±1.5 |

Table D.2: Performance comparison with state-of-the-art methods on **CCC-Medium**, including nine variations with three corruption transition speeds (1000 / 2000 / 5000) and three corruption orderings determined by random seeds (43 / 44 / 45).

| Transition speed | 1000 | | | 2000 | | | 5000 | | | Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Corruption ordering | 43 | 44 | 45 | 43 | 44 | 45 | 43 | 44 | 45 | Mean |
| Source | 1.29 | 1.23 | 1.31 | 1.31 | 1.23 | 1.30 | 1.33 | 1.19 | 1.25 | 1.27±0.0 |
| RMT (CVPR'23) | 12.13 | 13.18 | 13.12 | 9.43 | 10.74 | 12.83 | 7.73 | 0.86 | 9.27 | 9.92±3.7 |
| RoTTA (CVPR'23) | 0.50 | 0.77 | 0.74 | 0.66 | 0.77 | 0.96 | 0.79 | 0.17 | 0.87 | 0.69±0.2 |
| SANTA (TMLR'23) | 9.28 | 9.93 | 9.16 | 9.08 | 9.89 | 9.14 | 8.96 | 9.77 | 9.97 | 9.46±0.4 |
| LAW (WACV'24) | 0.34 | 0.17 | 0.22 | 0.31 | 0.17 | 0.22 | 0.27 | 0.16 | 0.20 | 0.23±0.1 |
| ViDA (ICLR'24) | 0.44 | 0.39 | 0.45 | 0.45 | 0.40 | 0.44 | 0.48 | 0.38 | 0.38 | 0.42±0.0 |
| DPLOT (CVPR'24) | 0.53 | 0.88 | 0.77 | 0.64 | 0.24 | 0.31 | 1.22 | 0.12 | 0.36 | 0.56±0.3 |
| PALM (AAAI'25) | 0.14 | 0.12 | 0.10 | 0.14 | 0.11 | 0.17 | 0.13 | 0.12 | 0.16 | 0.13±0.0 |
| EATA (ICML'22) | 1.25 | 0.80 | 0.57 | 1.11 | 0.49 | 0.64 | 1.67 | 0.32 | 0.51 | 0.82±0.4 |
| + COME (ICLR'25) | 0.74 | 0.96 | 0.79 | 1.07 | 0.29 | 0.85 | 1.51 | 0.21 | 0.79 | 0.80±0.4 |
| CoTTA (CVPR'22) | 1.73 | 1.98 | 1.95 | 1.43 | 1.71 | 2.09 | 1.44 | 0.20 | 1.19 | 1.52±0.5 |
| SAR (ICLR'23) | 1.54 | 1.64 | 1.52 | 1.61 | 2.29 | 1.67 | 2.90 | 2.50 | 2.56 | 2.03±0.5 |
| + COME (ICLR'25) | 2.90 | 1.94 | 1.94 | 1.22 | 1.98 | 2.04 | 2.18 | 1.06 | 3.50 | 2.08±0.7 |
| PETAL (CVPR'23) | 0.68 | 0.64 | 0.74 | 0.81 | 0.56 | 0.80 | 0.96 | 0.14 | 0.55 | 0.65±0.2 |
| CMF (ICLR'24) | 1.06 | 0.62 | 0.46 | 1.08 | 0.40 | 0.73 | 2.41 | 0.29 | 0.95 | 0.89±0.6 |
| DATTA (ECCV'24) | 3.00 | 2.48 | 2.57 | 1.49 | 1.51 | 1.56 | 1.61 | 1.70 | 1.53 | 1.94±0.5 |
| PeTTA (NeurIPS'24) | 4.93 | 5.44 | 4.70 | 5.88 | 6.53 | 5.71 | 6.58 | 7.12 | 7.15 | 6.00±0.8 |
| ETA (ICML'22) | 0.67 | 0.28 | 0.26 | 0.41 | 0.18 | 0.29 | 0.34 | 0.19 | 0.24 | 0.32±0.1 |
| + RDumb (NeurIPS'23) | 7.58 | 9.64 | 6.90 | 9.46 | 11.08 | 8.74 | 10.33 | 12.67 | 11.57 | 9.77±1.8 |
| + ASR (Ours) | 15.01 | 18.18 | 13.36 | 15.95 | 18.57 | 15.83 | 18.07 | 18.32 | 20.59 | 17.10±2.1 |
| ROID (WACV'24) | 12.64 | 15.79 | 13.28 | 9.63 | 12.65 | 11.81 | 10.66 | 8.72 | 17.12 | 12.48±2.6 |
| + RDumb (NeurIPS'23) | 14.13 | 15.92 | 13.74 | 14.03 | 16.05 | 14.06 | 15.48 | 17.34 | 17.98 | 15.41±1.5 |
| + ASR (Ours) | 20.99 | 22.51 | 20.40 | 21.22 | 22.93 | 21.36 | 22.37 | 23.84 | 24.25 | 22.21±1.2 |

Table D.3: Performance comparison with state-of-the-art methods on **CCC-Hard**, containing nine variations with three corruption transition speeds (1000 / 2000 / 5000) and three corruption orderings determined by random seeds (43 / 44 / 45).

| Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Source | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01±0.0 |
| RMT (CVPR'23) | 47.68 | 45.48 | 44.09 | 43.87 | 46.48 | 45.70 | 44.68 | 44.86 | 43.61 | 43.64 | 45.01±1.3 |
| + Source-free | 42.33 | 39.13 | 33.49 | 36.63 | 40.32 | 38.21 | 37.75 | 34.24 | 33.02 | 33.63 | 36.88±3.1 |
| RoTTA (CVPR'23) | 27.21 | 31.85 | 27.23 | 24.99 | 28.56 | 30.99 | 30.55 | 29.61 | 30.70 | 28.85 | 29.05±2.0 |
| SANTA (TMLR'23) | 40.00 | 39.85 | 39.83 | 39.77 | 39.84 | 39.53 | 39.83 | 39.85 | 39.63 | 39.98 | 39.81±0.1 |
| LAW (WACV'24) | 22.91 | 17.65 | 1.14 | 14.63 | 24.72 | 17.70 | 10.91 | 11.27 | 11.66 | 2.06 | 13.47±7.4 |
| ViDA (ICLR'24) | 17.87 | 17.81 | 17.62 | 17.76 | 17.78 | 17.83 | 17.77 | 17.80 | 17.79 | 17.60 | 17.76±0.1 |
| DPLOT (CVPR'24) | 37.52 | 33.86 | 30.34 | 31.38 | 33.64 | 29.72 | 32.60 | 30.58 | 29.99 | 30.38 | 32.00±2.3 |
| PALM (AAAI'25) | 21.14 | 15.12 | 3.47 | 16.37 | 23.57 | 14.06 | 8.57 | 11.02 | 8.86 | 4.75 | 12.69±6.3 |
| EATA (ICML'22) | 48.03 | 47.60 | 47.85 | 47.42 | 48.18 | 47.87 | 47.75 | 47.78 | 48.01 | 47.62 | 47.81±0.2 |
| + COME (ICLR'25) | 44.34 | 43.66 | 44.13 | 43.59 | 44.48 | 44.24 | 44.04 | 44.16 | 44.62 | 44.13 | 44.14±0.3 |
| CoTTA (CVPR'22) | 39.59 | 36.76 | 31.44 | 35.60 | 38.70 | 36.71 | 36.41 | 34.66 | 33.18 | 32.03 | 35.51±2.6 |
| SAR (ICLR'23) | 41.62 | 41.13 | 40.77 | 40.04 | 41.61 | 40.71 | 41.48 | 40.63 | 40.44 | 35.11 | 40.35±1.8 |
| + COME (ICLR'25) | 43.47 | 42.97 | 42.62 | 42.69 | 43.46 | 43.12 | 43.00 | 42.95 | 42.79 | 42.50 | 42.96±0.3 |
| PETAL (CVPR'23) | 40.87 | 38.09 | 33.08 | 38.92 | 40.03 | 38.73 | 37.50 | 36.94 | 34.65 | 34.03 | 37.28±2.5 |
| CMF (ICLR'24) | 48.74 | 48.41 | 48.67 | 48.35 | 48.83 | 48.61 | 48.57 | 48.58 | 48.80 | 48.56 | 48.61±0.1 |
| DATTA (ECCV'24) | 35.97 | 37.58 | 33.45 | 37.68 | 35.69 | 32.80 | 31.88 | 36.86 | 28.88 | 34.81 | 34.56±2.7 |
| PeTTA (NeurIPS'24) | 31.57 | 31.57 | 31.44 | 31.59 | 31.56 | 31.36 | 31.60 | 31.40 | 31.65 | 31.76 | 31.55±0.1 |
| ETA (ICML'22) | 43.68 | 43.69 | 42.97 | 42.91 | 44.19 | 44.12 | 43.84 | 43.79 | 43.88 | 43.03 | 43.61±0.4 |
| + RDumb (NeurIPS'23) | 46.44 | 46.09 | 46.48 | 46.06 | 46.54 | 46.39 | 46.40 | 46.46 | 46.75 | 46.31 | 46.39±0.2 |
| + ASR (Ours) | 47.50 | 46.89 | 47.10 | 46.89 | 47.51 | 47.43 | 47.26 | 47.22 | 47.15 | 46.79 | 47.17±0.2 |
| ROID (WACV'24) | 48.66 | 48.53 | 48.57 | 48.47 | 48.66 | 48.56 | 48.56 | 48.53 | 48.66 | 48.56 | 48.58±0.1 |
| + RDumb (NeurIPS'23) | 48.01 | 47.92 | 48.07 | 47.90 | 48.02 | 47.96 | 48.04 | 48.02 | 48.10 | 48.00 | 48.00±0.1 |
| + ASR (Ours) | 49.76 | 49.31 | 49.40 | 49.20 | 49.78 | 49.63 | 49.42 | 49.60 | 49.54 | 49.32 | 49.50±0.2 |

Table D.4: Accuracy (%) on **CIN-C** over ten random permutations of the corruption order.

| Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Source | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01 | 18.01±0.0 |
| RMT (CVPR'23) | 46.53 | 44.99 | 42.80 | 44.17 | 45.95 | 44.01 | 43.88 | 43.59 | 42.99 | 42.92 | 44.18±1.2 |
| + Source-free | 42.07 | 38.68 | 32.79 | 36.20 | 40.16 | 37.47 | 37.30 | 34.04 | 32.34 | 33.08 | 36.41±3.2 |
| RoTTA (CVPR'23) | 29.33 | 32.30 | 27.21 | 27.16 | 29.09 | 32.17 | 30.24 | 30.19 | 30.41 | 28.96 | 29.71±1.7 |
| SANTA (TMLR'23) | 39.60 | 39.38 | 39.28 | 39.28 | 39.54 | 39.52 | 39.37 | 39.44 | 39.42 | 39.16 | 39.40±0.1 |
| LAW (WACV'24) | 21.82 | 15.81 | 1.47 | 15.34 | 24.20 | 16.41 | 9.00 | 13.86 | 13.06 | 2.74 | 13.37±6.9 |
| ViDA (ICLR'24) | 17.86 | 17.82 | 17.60 | 17.77 | 17.77 | 17.85 | 17.78 | 17.80 | 17.79 | 17.60 | 17.76±0.1 |
| DPLOT (CVPR'24) | 36.98 | 34.19 | 30.29 | 30.54 | 34.04 | 27.87 | 33.05 | 30.02 | 29.84 | 29.38 | 31.62±2.7 |
| PALM (AAAI'25) | 19.09 | 14.37 | 3.18 | 16.71 | 22.72 | 13.95 | 7.52 | 10.76 | 8.30 | 4.23 | 12.08±6.1 |
| EATA (ICML'22) | 47.70 | 47.29 | 47.63 | 47.12 | 47.89 | 47.63 | 47.51 | 47.52 | 47.71 | 47.41 | 47.54±0.2 |
| + COME (ICLR'25) | 44.26 | 43.69 | 44.11 | 43.62 | 44.41 | 44.11 | 43.86 | 44.24 | 44.55 | 44.05 | 44.09±0.3 |
| CoTTA (CVPR'22) | 39.10 | 36.39 | 31.57 | 35.61 | 38.40 | 36.41 | 36.15 | 34.40 | 32.14 | 32.73 | 35.29±2.4 |
| SAR (ICLR'23) | 40.75 | 40.57 | 40.08 | 39.04 | 40.89 | 39.68 | 40.30 | 39.52 | 39.44 | 40.40 | 40.07±0.6 |
| + COME (ICLR'25) | 42.98 | 42.66 | 42.36 | 42.17 | 43.00 | 42.72 | 42.57 | 42.58 | 42.48 | 42.10 | 42.56±0.3 |
| PETAL (CVPR'23) | 26.41 | 23.71 | 17.45 | 22.96 | 24.88 | 22.74 | 22.97 | 20.34 | 17.88 | 19.07 | 21.84±2.9 |
| CMF (ICLR'24) | 48.44 | 48.06 | 48.28 | 48.03 | 48.57 | 48.33 | 48.15 | 48.27 | 48.44 | 48.19 | 48.28±0.2 |
| DATTA (ECCV'24) | 7.94 | 3.30 | 2.42 | 1.81 | 3.75 | 2.46 | 1.66 | 4.07 | 2.72 | 2.37 | 3.25±1.7 |
| PeTTA (NeurIPS'24) | 31.49 | 31.62 | 31.60 | 31.55 | 31.57 | 31.69 | 31.66 | 31.47 | 31.69 | 31.74 | 31.61±0.1 |
| ETA (ICML'22) | 43.75 | 43.61 | 43.29 | 43.09 | 44.32 | 43.95 | 43.90 | 43.56 | 43.72 | 43.08 | 43.63±0.4 |
| + RDumb (NeurIPS'23) | 46.21 | 45.92 | 46.34 | 45.68 | 46.20 | 46.12 | 46.19 | 46.18 | 46.42 | 46.00 | 46.13±0.2 |
| + ASR (Ours) | 47.31 | 46.62 | 46.93 | 46.47 | 47.04 | 47.00 | 46.72 | 46.87 | 46.81 | 46.50 | 46.83±0.2 |
| ROID (WACV'24) | 48.46 | 48.32 | 48.25 | 48.11 | 48.28 | 48.27 | 48.17 | 48.24 | 48.24 | 48.16 | 48.25±0.1 |
| + RDumb (NeurIPS'23) | 47.64 | 47.60 | 47.77 | 47.60 | 47.64 | 47.72 | 47.58 | 47.72 | 47.75 | 47.66 | 47.67±0.1 |
| + ASR (Ours) | 49.37 | 48.98 | 49.07 | 48.84 | 49.45 | 49.27 | 49.04 | 49.27 | 49.16 | 48.99 | 49.14±0.2 |

Table D.5: Accuracy (%) on *non-i.i.d.* **CIN-C** over ten random permutations of the corruption order.

| Method | *Recurring visit* → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08 | 3.08±0.0 |
| RMT | 27.63 | 33.91 | 37.28 | 39.08 | 39.99 | 40.78 | 41.18 | 41.36 | 41.72 | 41.76 | 41.76 | 41.92 | 42.01 | 42.02 | 41.96 | 42.05 | 42.10 | 42.08 | 42.08 | 42.09 | 40.24±3.5 |
| + Source-free | 27.63 | 34.15 | 37.14 | 38.11 | 38.70 | 38.87 | 39.22 | 39.43 | 39.40 | 39.60 | 39.60 | 39.60 | 39.71 | 39.80 | 39.76 | 39.68 | 39.74 | 39.79 | 39.76 | 39.76 | 38.47±2.8 |
| RoTTA | 12.45 | 17.22 | 19.19 | 20.77 | 19.92 | 21.29 | 21.88 | 21.23 | 19.71 | 19.25 | 18.70 | 18.00 | 17.33 | 16.91 | 16.20 | 15.58 | 14.97 | 14.44 | 13.97 | 12.96 | 17.60±2.8 |
| SANTA | 27.28 | 27.75 | 27.30 | 27.20 | 27.10 | 26.94 | 27.04 | 26.81 | 26.91 | 26.59 | 26.49 | 26.42 | 26.53 | 26.39 | 26.25 | 26.38 | 26.18 | 26.17 | 26.25 | 25.94 | 26.70±0.5 |
| LAW | 23.83 | 30.62 | 31.98 | 32.03 | 31.60 | 31.11 | 30.75 | 30.34 | 30.06 | 29.76 | 29.65 | 29.52 | 29.44 | 29.36 | 29.37 | 29.35 | 29.41 | 29.37 | 29.31 | 29.24 | 29.81±1.6 |
| ViDA | 3.09 | 3.09 | 3.08 | 3.08 | 3.07 | 3.05 | 3.02 | 3.03 | 3.02 | 3.02 | 3.02 | 2.99 | 2.97 | 2.95 | 2.92 | 2.91 | 2.89 | 2.89 | 2.88 | 2.84 | 2.99±0.1 |
| DPLOT | 30.16 | 33.83 | 35.76 | 36.61 | 36.94 | 37.07 | 37.18 | 37.14 | 37.28 | 37.41 | 37.35 | 37.33 | 37.36 | 37.38 | 37.35 | 37.34 | 37.35 | 37.36 | 37.37 | 37.39 | 36.65±1.7 |
| PALM | 24.66 | 31.70 | 32.18 | 31.71 | 31.29 | 30.79 | 30.71 | 30.74 | 30.76 | 30.76 | 30.81 | 30.78 | 30.78 | 30.86 | 30.82 | 30.91 | 30.93 | 30.92 | 30.96 | 30.98 | 30.70±1.4 |
| EATA | 31.31 | 36.38 | 36.70 | 36.90 | 36.98 | 36.67 | 36.56 | 36.60 | 36.73 | 36.79 | 36.52 | 36.56 | 36.47 | 36.40 | 36.46 | 36.52 | 36.54 | 36.45 | 36.48 | 36.35 | 36.32±1.2 |
| + COME | 30.20 | 34.59 | 34.90 | 34.52 | 34.17 | 33.88 | 33.82 | 33.44 | 33.25 | 32.99 | 32.97 | 32.82 | 32.57 | 32.44 | 32.40 | 32.48 | 32.52 | 32.35 | 32.10 | 32.06 | 33.02±1.1 |
| CoTTA | 18.78 | 24.90 | 29.02 | 31.39 | 33.47 | 34.66 | 35.47 | 35.96 | 36.28 | 36.55 | 36.70 | 36.94 | 37.05 | 37.17 | 37.24 | 37.25 | 37.20 | 37.25 | 37.24 | 37.22 | 34.39±4.8 |
| SAR | 24.38 | 31.54 | 33.42 | 34.06 | 34.40 | 34.52 | 34.70 | 34.85 | 35.00 | 35.08 | 35.11 | 35.02 | 35.02 | 35.03 | 34.94 | 34.98 | 34.93 | 34.99 | 34.97 | 34.93 | 34.09±2.4 |
| + COME | 23.67 | 30.97 | 33.02 | 33.97 | 34.50 | 35.10 | 35.15 | 35.20 | 35.30 | 35.41 | 35.36 | 35.40 | 35.38 | 35.39 | 35.33 | 35.27 | 35.27 | 35.30 | 35.28 | 35.24 | 34.28±2.7 |
| PETAL | 18.74 | 25.64 | 29.12 | 30.91 | 31.76 | 32.36 | 32.80 | 33.28 | 33.55 | 33.75 | 33.89 | 34.04 | 34.10 | 34.18 | 34.21 | 34.24 | 34.22 | 34.24 | 34.24 | 34.24 | 32.18±3.7 |
| CMF | 35.07 | 38.66 | 39.22 | 39.52 | 39.58 | 39.62 | 39.90 | 39.95 | 39.92 | 39.76 | 39.70 | 39.73 | 39.28 | 39.61 | 39.54 | 39.65 | 39.52 | 39.84 | 39.52 | 39.40 | 39.35±1.0 |
| DATTA | 20.11 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.67 | 19.69±0.1 |
| PeTTA | 11.91 | 12.63 | 12.61 | 12.83 | 12.65 | 13.16 | 12.96 | 12.76 | 12.61 | 12.72 | 12.72 | 12.27 | 12.72 | 12.74 | 12.80 | 12.60 | 12.88 | 12.33 | 12.73 | 12.40 | 12.65±0.3 |
| ETA | 30.64 | 35.80 | 36.56 | 36.67 | 36.76 | 36.58 | 36.45 | 36.47 | 36.28 | 36.16 | 36.08 | 36.00 | 36.06 | 36.01 | 35.96 | 35.91 | 35.86 | 35.76 | 35.82 | 35.80 | 35.88±1.2 |
| + RDumb | 30.71 | 35.95 | 36.80 | 30.73 | 35.66 | 36.30 | 31.97 | 35.98 | 36.97 | 32.71 | 34.87 | 36.66 | 34.06 | 33.88 | 36.60 | 35.83 | 33.07 | 36.51 | 36.92 | 30.94 | 34.66±2.2 |
| + ASR (Ours) | 28.68 | 33.09 | 34.65 | 33.52 | 33.00 | 34.86 | 36.24 | 37.32 | 38.02 | 38.60 | 38.79 | 38.86 | 38.89 | 38.86 | 38.97 | 39.23 | 39.07 | 39.16 | 39.07 | 39.10 | 36.90±2.9 |
| ROID | 35.32 | 37.74 | 38.21 | 37.96 | 38.00 | 38.16 | 38.02 | 38.02 | 38.10 | 38.08 | 38.43 | 38.51 | 37.95 | 38.20 | 38.15 | 38.16 | 37.98 | 38.16 | 37.97 | 38.02 | 37.96±0.6 |
| + RDumb | 35.60 | 38.28 | 38.34 | 35.08 | 38.02 | 38.34 | 35.21 | 37.61 | 37.76 | 35.12 | 37.72 | 38.48 | 36.00 | 37.49 | 38.25 | 37.16 | 37.32 | 38.34 | 37.70 | 35.75 | 37.18±1.2 |
| + ASR (Ours) | 35.66 | 39.42 | 39.64 | 40.42 | 41.03 | 41.40 | 41.74 | 41.83 | 41.87 | 42.20 | 42.46 | 42.48 | 42.76 | 42.12 | 42.06 | 42.60 | 42.67 | 42.86 | 43.08 | 42.96 | 41.56±1.7 |

Table D.6: Accuracy (%) on **IN-C** across 20 recurring visits of the domain sequence.

| Method | *Recurring visit* → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52 | 32.52±0.0 |
| RMT | 43.16 | 45.54 | 46.26 | 46.51 | 46.74 | 46.78 | 46.81 | 46.83 | 46.88 | 46.88 | 46.87 | 46.88 | 46.90 | 46.89 | 46.90 | 46.90 | 46.89 | 46.88 | 46.86 | 46.87 | 46.56±0.8 |
| + Source-free | 42.24 | 43.73 | 43.82 | 43.92 | 43.94 | 43.98 | 43.81 | 43.86 | 43.87 | 43.88 | 43.77 | 43.81 | 43.80 | 43.76 | 43.75 | 43.71 | 43.68 | 43.68 | 43.68 | 43.65 | 43.72±0.4 |
| RoTTA | 39.89 | 43.06 | 44.03 | 44.36 | 44.34 | 43.94 | 43.66 | 43.26 | 42.71 | 42.07 | 41.41 | 40.63 | 40.04 | 39.38 | 38.66 | 37.85 | 37.04 | 36.20 | 35.23 | 34.34 | 40.61±3.1 |
| SANTA | 41.52 | 41.68 | 41.74 | 41.66 | 41.75 | 41.80 | 41.68 | 41.59 | 41.65 | 41.54 | 41.44 | 41.47 | 41.39 | 41.58 | 41.50 | 41.41 | 41.42 | 41.34 | 41.40 | 41.29 | 41.54±0.1 |
| LAW | 40.19 | 35.70 | 32.19 | 30.78 | 30.25 | 30.01 | 29.85 | 29.78 | 29.75 | 29.72 | 29.68 | 29.67 | 29.65 | 29.67 | 29.67 | 29.67 | 29.66 | 29.66 | 29.66 | 29.67 | 30.74±2.6 |
| ViDA | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01±0.0 |
| DPLOT | 42.09 | 42.46 | 42.35 | 42.26 | 42.24 | 42.12 | 42.17 | 42.16 | 42.14 | 42.12 | 42.12 | 42.12 | 42.12 | 42.10 | 42.10 | 42.11 | 42.11 | 42.10 | 42.09 | 42.10 | 42.16±0.1 |
| PALM | 13.86 | 1.74 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 1.42 | 2.06±2.7 |
| EATA | 41.62 | 42.42 | 42.21 | 41.77 | 41.99 | 41.96 | 41.96 | 41.58 | 41.57 | 41.34 | 41.30 | 41.46 | 41.50 | 41.54 | 41.10 | 41.23 | 41.37 | 41.57 | 41.43 | 41.32 | 41.61±0.3 |
| + COME | 42.94 | 45.13 | 45.46 | 45.36 | 45.73 | 45.46 | 45.30 | 45.24 | 45.52 | 45.30 | 45.11 | 45.48 | 45.29 | 45.16 | 45.04 | 45.25 | 44.89 | 44.87 | 44.78 | 44.91 | 45.11±0.6 |
| CoTTA | 41.76 | 45.70 | 46.79 | 46.90 | 46.72 | 46.30 | 45.85 | 45.42 | 44.99 | 44.62 | 44.30 | 43.91 | 43.43 | 42.87 | 42.49 | 41.92 | 41.54 | 41.22 | 40.82 | 40.55 | 43.91±2.1 |
| SAR | 40.86 | 42.94 | 43.26 | 43.15 | 42.93 | 42.50 | 42.06 | 41.53 | 40.87 | 40.17 | 39.47 | 38.76 | 38.01 | 37.23 | 36.49 | 35.70 | 34.97 | 34.22 | 33.59 | 33.11 | 39.09±3.4 |
| + COME | 40.59 | 43.57 | 44.50 | 44.98 | 45.19 | 45.22 | 45.12 | 44.91 | 44.65 | 44.16 | 43.58 | 43.02 | 42.31 | 41.53 | 40.78 | 39.89 | 38.91 | 37.74 | 36.48 | 34.96 | 42.10±3.1 |
| PETAL | 0.03 | 0.03 | 0.02 | 0.03 | 0.03 | 0.09 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.39 | 0.28±0.2 |
| CMF | 44.69 | 45.21 | 45.42 | 45.25 | 45.38 | 45.59 | 44.92 | 45.14 | 45.31 | 45.55 | 45.72 | 45.52 | 45.18 | 45.26 | 44.94 | 45.00 | 45.02 | 45.13 | 45.27 | 45.46 | 45.25±0.3 |
| DATTA | 33.75 | 3.50 | 1.33 | 0.88 | 0.85 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 2.65±7.2 |
| PeTTA | 39.56 | 42.05 | 42.84 | 43.02 | 43.12 | 43.27 | 43.26 | 43.23 | 43.12 | 43.08 | 42.94 | 42.95 | 42.91 | 42.93 | 42.98 | 42.96 | 42.81 | 42.75 | 42.70 | 42.69 | 42.76±0.8 |
| ETA | 41.24 | 40.92 | 40.30 | 39.86 | 39.07 | 38.49 | 38.26 | 37.64 | 37.28 | 36.96 | 36.51 | 36.14 | 36.28 | 35.94 | 35.67 | 35.47 | 35.09 | 34.74 | 34.37 | 34.21 | 37.22±2.1 |
| + RDumb | 40.93 | 41.36 | 42.11 | 41.66 | 41.18 | 41.46 | 41.28 | 41.84 | 41.31 | 40.78 | 41.90 | 42.09 | 41.70 | 40.85 | 41.60 | 41.44 | 41.59 | 41.52 | 40.75 | 41.59 | 41.45±0.4 |
| + ASR (Ours) | 40.61 | 41.46 | 41.49 | 41.74 | 41.79 | 42.04 | 41.82 | 41.70 | 41.84 | 41.88 | 41.74 | 41.76 | 41.60 | 41.38 | 41.27 | 41.36 | 41.16 | 41.28 | 41.36 | 41.32 | 41.53±0.3 |
| ROID | 46.02 | 46.22 | 46.03 | 46.33 | 46.22 | 46.29 | 46.14 | 45.94 | 46.32 | 46.04 | 46.12 | 46.26 | 46.03 | 46.13 | 46.16 | 46.20 | 46.23 | 46.22 | 46.19 | 46.17 | 46.16±0.1 |
| + RDumb | 46.07 | 46.34 | 46.32 | 46.25 | 46.24 | 46.23 | 46.16 | 46.04 | 46.07 | 46.14 | 45.86 | 45.86 | 45.86 | 45.94 | 45.79 | 45.75 | 45.80 | 45.81 | 45.68 | 45.62 | 45.99±0.2 |
| + ASR (Ours) | 46.13 | 46.50 | 46.53 | 46.63 | 46.52 | 46.52 | 46.49 | 46.61 | 46.70 | 46.55 | 46.56 | 46.63 | 46.53 | 46.54 | 46.48 | 46.39 | 46.48 | 46.33 | 46.40 | 46.32 | 46.49±0.1 |

Table D.7: Accuracy (%) on **IN-D109** across 20 recurring visits of the domain sequence.

### D.2 Full Results on ViT

In Tables D.8–D.10, we present the full evaluation results on ViT-B-16, extending Table 2.

| Transition speed | 1000 | | | 2000 | | | 5000 | | | Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Corruption ordering | 43 | 44 | 45 | 43 | 44 | 45 | 43 | 44 | 45 | Mean |
| Source | 54.74 | 55.18 | 54.47 | 54.97 | 55.03 | 54.77 | 55.09 | 54.88 | 55.12 | 54.92±0.2 |
| CMF (ICLR'24) | 60.68 | 60.96 | 60.59 | 61.51 | 61.74 | 61.51 | 62.66 | 62.52 | 61.52 | 61.52±0.7 |
| CMAE (CVPR'24) | 48.11 | 49.94 | 48.19 | 50.68 | 50.44 | 52.57 | 51.04 | 54.89 | 54.50 | 51.15±2.3 |
| REM (ICML'25) | 65.82 | 66.12 | 65.79 | 66.14 | 66.23 | 66.05 | 66.93 | 66.02 | 66.36 | 66.16±0.3 |
| ETA (ICML'22) | 47.95 | 47.68 | 47.47 | 48.71 | 15.74 | 48.44 | 49.92 | 49.78 | 49.93 | 45.07±10.4 |
| + RDumb (NeurIPS'23) | 59.25 | 59.56 | 59.13 | 59.74 | 60.07 | 59.74 | 60.76 | 60.75 | 60.91 | 59.99±0.6 |
| + ASR (Ours) | 59.62 | 59.99 | 59.59 | 60.34 | 60.69 | 60.46 | 61.57 | 61.33 | 61.62 | 60.58±0.7 |
| ROID (WACV'24) | 60.01 | 60.32 | 59.88 | 60.67 | 60.99 | 60.68 | 61.66 | 61.64 | 61.82 | 60.85±0.7 |
| + RDumb (NeurIPS'23) | 59.78 | 60.09 | 59.65 | 60.44 | 60.75 | 60.43 | 61.35 | 61.39 | 61.56 | 60.60±0.7 |
| + ASR (Ours) | 60.61 | 60.86 | 60.51 | 61.28 | 61.54 | 61.30 | 62.46 | 62.21 | 62.52 | 61.48±0.7 |

Table D.8: Performance comparison with state-of-the-art methods on **CCC-Easy**, containing nine variations with three corruption transition speeds (1000 / 2000 / 5000) and three corruption orderings determined by random seeds (43 / 44 / 45).

| Transition speed | 1000 | | | 2000 | | | 5000 | | | Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Corruption ordering | 43 | 44 | 45 | 43 | 44 | 45 | 43 | 44 | 45 | Mean |
| Source | 41.76 | 41.47 | 40.49 | 42.32 | 41.10 | 41.49 | 42.15 | 42.39 | 42.51 | 41.74±0.6 |
| CMF (ICLR'24) | 52.17 | 51.63 | 51.87 | 34.21 | 53.49 | 53.59 | 55.41 | 55.33 | 55.82 | 51.50±6.3 |
| CMAE (CVPR'24) | 41.76 | 36.63 | 41.88 | 43.16 | 40.52 | 44.40 | 45.02 | 46.41 | 51.52 | 43.48±3.9 |
| REM (ICML'25) | 57.34 | 57.23 | 56.92 | 58.36 | 57.28 | 57.68 | 58.94 | 58.41 | 59.71 | 57.99±0.9 |
| ETA (ICML'22) | 34.62 | 23.74 | 28.04 | 34.04 | 34.54 | 36.05 | 35.36 | 38.07 | 38.97 | 33.71±4.6 |
| + RDumb (NeurIPS'23) | 49.02 | 48.88 | 48.57 | 50.77 | 50.16 | 50.28 | 51.97 | 52.25 | 52.58 | 50.50±1.4 |
| + ASR (Ours) | 49.86 | 49.69 | 49.57 | 51.96 | 51.39 | 51.46 | 53.31 | 53.55 | 53.92 | 51.63±1.6 |
| ROID (WACV'24) | 50.72 | 50.67 | 50.39 | 52.61 | 51.91 | 52.00 | 53.62 | 53.72 | 54.08 | 52.19±1.3 |
| + RDumb (NeurIPS'23) | 50.23 | 50.20 | 49.81 | 52.11 | 51.45 | 51.39 | 53.09 | 53.26 | 53.62 | 51.68±1.3 |
| + ASR (Ours) | 52.14 | 52.08 | 51.98 | 53.91 | 53.07 | 53.26 | 55.03 | 54.95 | 55.57 | 53.55±1.3 |

Table D.9: Performance comparison with state-of-the-art methods on **CCC-Medium**, including nine variations with three corruption transition speeds (1000 / 2000 / 5000) and three corruption orderings determined by random seeds (43 / 44 / 45).

| Transition speed | 1000 | | | 2000 | | | 5000 | | | Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Corruption ordering | 43 | 44 | 45 | 43 | 44 | 45 | 43 | 44 | 45 | Mean |
| Source | 14.40 | 15.44 | 15.40 | 14.16 | 15.38 | 14.10 | 13.90 | 15.31 | 15.40 | 14.83±0.6 |
| CMF (ICLR'24) | 1.22 | 0.30 | 2.74 | 2.17 | 0.14 | 0.85 | 3.23 | 0.13 | 5.34 | 1.79±1.7 |
| CMAE (CVPR'24) | 26.47 | 26.78 | 22.70 | 25.95 | 28.33 | 24.96 | 26.60 | 30.27 | 30.20 | 26.92±2.3 |
| REM (ICML'25) | 3.80 | 8.53 | 7.03 | 5.58 | 5.94 | 9.39 | 10.67 | 38.45 | 9.31 | 10.97±9.9 |
| ETA (ICML'22) | 1.34 | 0.33 | 1.66 | 0.99 | 1.34 | 1.02 | 1.97 | 1.58 | 0.79 | 1.22±0.5 |
| + RDumb (NeurIPS'23) | 22.41 | 24.43 | 22.01 | 23.52 | 25.54 | 23.39 | 22.16 | 23.52 | 22.42 | 23.27±1.1 |
| + ASR (Ours) | 24.67 | 25.88 | 24.14 | 23.93 | 25.21 | 24.26 | 22.76 | 23.96 | 25.20 | 24.45±0.9 |
| ROID (WACV'24) | 11.74 | 23.23 | 1.00 | 25.75 | 9.08 | 25.10 | 12.49 | 6.75 | 13.55 | 14.30±8.2 |
| + RDumb (NeurIPS'23) | 24.17 | 25.40 | 23.76 | 25.05 | 26.62 | 24.84 | 26.25 | 27.37 | 28.01 | 25.72±1.4 |
| + ASR (Ours) | 27.69 | 28.76 | 27.31 | 27.62 | 28.82 | 27.52 | 27.58 | 28.67 | 28.85 | 28.09±0.6 |

Table D.10: Performance comparison with state-of-the-art methods on **CCC-Hard**, containing nine variations with three corruption transition speeds (1000 / 2000 / 5000) and three corruption orderings determined by random seeds (43 / 44 / 45).

### D.3 RESULTS FOR CIFAR10-C/100-C

We are interested in more challenging yet realistic environments, as proposed by Press et al. (2023). Standard CIFAR without repeating corruptions is relatively simple and less realistic. Thus, we group corruption types into three levels (Easy / Medium / Hard) for consistent adaptation difficulty across corruptions, and repeat them cyclically, following Press et al. (2023); Hoang et al. (2024). We report corruption types for each level in Table D.11 for CIFAR10-C and Table D.12 for CIFAR100-C. We also provide experimental results for CIFAR10-C/100-C, as shown in Fig. D.1–D.2.

| Level | Corruption Types |
|---|---|
| Easy | `Motion_blur, Snow, Fog, Elastic, JPEG` |
| Medium | `Defocus_blur, Glass_blur, Zoom_blur, Frost, Contrast, Pixelate` |
| Hard | `Gaussian_noise, Shot_noise, Impulse_noise` |

Table D.11: Corruption types for each level of CIFAR10-C.

| Level | Corruption Types |
|---|---|
| Easy | `Impulse_noise, Defocus_blur, Motion_blur, Zoom_blur, Snow, Brightness, Elastic` |
| Medium | `Glass_blur, Frost, Fog, Contrast, JPEG` |
| Hard | `Gaussian_noise, Shot_noise, Pixelate` |

Table D.12: Corruption types for each level of CIFAR100-C.

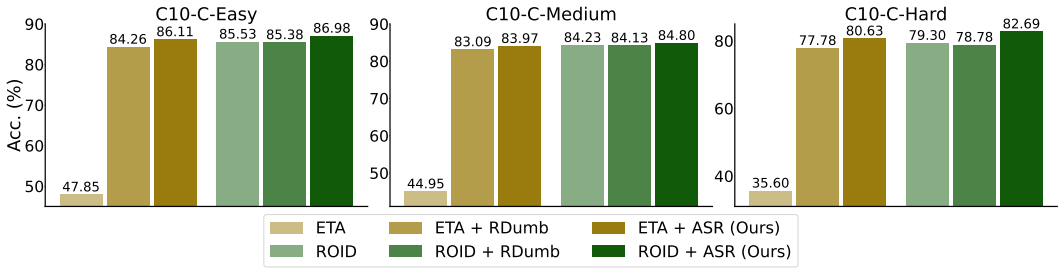


Figure D.1: Comparison of ETA / ROID and its variants with RDumb and ASR across three levels of CIFAR10-C using accuracy (%), averaged over 1000 recurring visits of the corruption sequence.

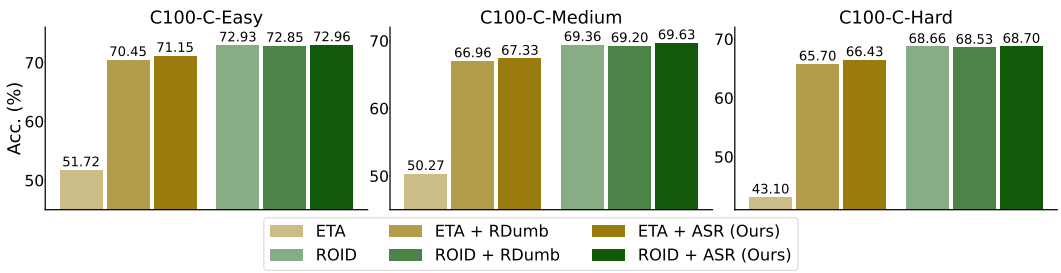


Figure D.2: Comparison of ETA / ROID and its variants with RDumb and ASR across three levels of CIFAR100-C using accuracy (%), averaged over 1000 recurring visits of the corruption sequence.

### D.4 Results on ViT-Tiny

We evaluate our method on one of the lightweight backbones (i.e., ViT-Tiny). Table D.13 shows that our method consistently improves over baselines on CCC-Medium and -Easy. Because the backbone capacity is extremely limited, adapting to CCC-Hard is particularly challenging, which is reflected in the table where all ROID variants achieve only 0.1% accuracy. Even with such low accuracies, our method achieves performance gains similar to those in Table 2, demonstrating its effectiveness despite severe capacity constraints.

| ViT-Tiny | CCC-Hard | CCC-Medium | CCC-Easy |
|---|---|---|---|
| ETA | 2.29 | 34.20 | 47.09 |
| + RDumb | 4.45 | 32.51 | 45.51 |
| + ASR (Ours) | 5.30 | 36.48 | 47.23 |
| ROID | 0.10 | 32.14 | 45.29 |
| + RDumb | 0.10 | 31.61 | 44.92 |
| + ASR (Ours) | 0.10 | 34.47 | 45.64 |

Table D.13: Accuracy (%) on ViT-Tiny across CCC benchmarks.

### D.5 Results under CDC Settings

We present full experimental results under CDC settings, extending Fig. 8.

| | *Recurring visit* → | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta = 1.0$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | Mean |
| ETA | 30.62 | 35.77 | 36.19 | 36.21 | 36.14 | 35.99 | 35.84 | 35.76 | 35.73 | 35.63 | 35.42 | 35.43 | 35.31 | 35.33 | 35.30 | 35.22 | 35.25 | 35.20 | 35.17 | 35.08 | 35.33 |
| + RDumb | 30.62 | 35.73 | 36.20 | 30.88 | 35.24 | 36.24 | 30.83 | 34.62 | 36.11 | 32.01 | 35.14 | 36.39 | 33.46 | 34.56 | 36.67 | 34.33 | 32.97 | 36.23 | 36.16 | 31.11 | 34.28 |
| + ASR (Ours) | 28.48 | 30.25 | 33.48 | 33.03 | 32.42 | 35.28 | 35.33 | 35.27 | 35.72 | 36.19 | 36.85 | 37.43 | 38.49 | 38.34 | 38.65 | 38.41 | 38.94 | 38.92 | 38.96 | 38.98 | 35.97 |
| ROID | 34.11 | 36.82 | 37.04 | 36.99 | 36.86 | 36.78 | 37.20 | 37.05 | 36.59 | 36.89 | 36.68 | 36.74 | 36.86 | 36.95 | 36.66 | 36.83 | 36.62 | 36.68 | 36.67 | 36.84 | 36.69 |
| + RDumb | 34.28 | 36.93 | 36.91 | 34.00 | 36.74 | 36.85 | 34.27 | 37.04 | 37.14 | 34.02 | 36.41 | 36.76 | 34.37 | 36.37 | 36.80 | 34.67 | 35.94 | 36.63 | 36.43 | 34.23 | 35.84 |
| + ASR (Ours) | 35.08 | 38.58 | 38.72 | 39.20 | 39.30 | 39.71 | 39.94 | 40.45 | 40.32 | 40.65 | 40.41 | 40.40 | 41.15 | 40.75 | 41.07 | 41.33 | 41.05 | 40.80 | 41.10 | 41.34 | 40.07 |

Table D.14: Results on IN-C with CDC for $\delta = 1.0$ across revisit steps.

| | *Recurring visit* → | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta = 10.0$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | Mean |
| ETA | 29.79 | 34.80 | 35.72 | 35.68 | 35.72 | 35.56 | 35.33 | 35.47 | 35.31 | 35.23 | 35.23 | 35.09 | 35.02 | 35.01 | 34.88 | 34.78 | 34.83 | 34.75 | 34.70 | 34.62 | 34.88 |
| + RDumb | 29.53 | 35.62 | 36.09 | 31.11 | 35.54 | 36.10 | 30.86 | 35.04 | 36.16 | 32.19 | 34.86 | 36.30 | 32.64 | 34.03 | 36.51 | 34.39 | 33.35 | 36.66 | 36.86 | 29.56 | 34.17 |
| + ASR (Ours) | 29.40 | 35.06 | 35.55 | 36.01 | 36.63 | 36.71 | 36.92 | 37.25 | 37.31 | 37.30 | 37.41 | 37.53 | 37.79 | 37.79 | 37.92 | 37.96 | 38.08 | 38.14 | 38.19 | 38.11 | 36.85 |
| ROID | 33.60 | 36.70 | 36.51 | 37.06 | 36.74 | 36.99 | 37.11 | 36.82 | 36.79 | 37.04 | 36.71 | 37.23 | 36.86 | 37.06 | 36.99 | 36.77 | 36.59 | 36.82 | 36.87 | 36.99 | 36.71 |
| + RDumb | 33.76 | 36.66 | 36.67 | 34.30 | 37.01 | 37.26 | 33.51 | 36.69 | 36.76 | 34.44 | 36.16 | 37.19 | 34.38 | 36.55 | 36.52 | 35.05 | 35.77 | 36.98 | 36.34 | 33.80 | 35.79 |
| + ASR (Ours) | 33.62 | 36.95 | 37.89 | 38.37 | 39.54 | 39.49 | 40.42 | 40.22 | 40.63 | 40.83 | 40.77 | 41.23 | 41.24 | 41.48 | 41.60 | 41.66 | 41.99 | 41.82 | 41.51 | 41.68 | 40.15 |

Table D.15: Results on IN-C with CDC for $\delta = 10.0$ across revisit steps.

# E Additional Ablation Studies

## E.1 Effect of Adaptive Reset

We validate the effectiveness of our adaptive reset by comparing to variants using fixed reset intervals. Table E.1 demonstrates that our adaptive reset can effectively identify when the model is likely to collapse and thereby find optimal reset timing, resulting in strong performance.

| Reset interval | Easy | Medium | Hard | Mean |
|---|---|---|---|---|
| **Fixed** | | | | |
| $T = 1000$ | 15.96 | 5.91 | 1.10 | 7.66 |
| $T = 10000$ | 49.88 | 39.69 | 15.75 | 35.11 |
| $T = 20000$ | 49.83 | 40.58 | 17.16 | 35.86 |
| $T = 50000$ | 49.90 | 40.16 | 14.26 | 34.77 |
| **Dynamic** | | | | |
| ASR (Ours) | **51.19** | **42.42** | **21.36** | **38.32** |

Table E.1: Comparison with our variants using fixed reset intervals $T$ on CCC using Accuracy (%).

## E.2 Effect of Selective Reset

Table E.2 demonstrates the effectiveness of our selective reset in comparison with fixed-proportion variants. We find that resetting the latter half of the layers (i.e., 50%) achieves the best results among the variants. Similarly, our selective reset also starts with 50% when adjusting the reset proportion (i.e., $r_0 = 0.5$). As a result, this suggests that our selective reset is effective and that at least a 50% reset should be ensured to effectively remove accumulated errors.

| Reset target | Easy | Medium | Hard | Mean |
|---|---|---|---|---|
| **Fixed** | | | | |
| 20% | 49.27 | 40.01 | 16.83 | 35.37 |
| 50% | 50.91 | 42.07 | 20.80 | 37.93 |
| 80% | 50.72 | 41.40 | 19.68 | 37.27 |
| 100% | 49.83 | 40.35 | 15.99 | 35.39 |
| **Dynamic** | | | | |
| ASR (Ours) | **51.19** | **42.42** | **21.36** | **38.32** |

Table E.2: Comparison with our variants that reset a fixed % of layers closer to the output.

### E.3 Effect of Hybrid Knowledge Accumulation

In our hybrid knowledge accumulation strategy of EMA on top of CMA, CMA highlights (locally) past information to reduce the effect of recent parameters near collapse, and EMA weights (globally) recent information to reflect distribution shifts. Table E.3 compares our hybrid scheme to the CMA-only baseline, evaluated across all CCC levels with accuracy (%) reported.

| Method | Easy | Medium | Hard | Mean |
|---|---|---|---|---|
| CMA-only | 50.03 | 41.13 | 18.42 | 36.53 |
| Hybrid (Ours) | **51.19** | **42.42** | **21.36** | **38.32** |

Table E.3: Effect of our hybrid accumulation scheme.

### E.4 Optimality of Reparameterization

We check whether our reparameterization (Eq. (6)–(7)) is optimal. For modeling reparameterization, we use only 5% of a holdout set (transition speed 2000; random seed 44) from CCC-Hard, and select an expression that best balances simplicity and performance efficacy. As reported in Tables E.4-E.5, we compare our expression to other expressions across all CCC levels. We often observe comparable results between two expressions. Either expression with high performance on CCC-Hard should be preferable to mitigate the risk of poor adaptation in real-world applications.

| $\lambda_{\mathcal{F}}$ | Range | Easy | Medium | Hard | Mean |
|---|---|---|---|---|---|
| $\lambda_0$ | $\{\lambda_0\}$ | 51.07 | 42.33 | 20.27 | 37.89 |
| $\phi_t$ | $[0, 1]$ | 51.09 | 42.26 | 20.56 | 37.97 |
| $\lambda_0 \cdot (1-\phi_t)^2$ | $[\lambda_0, 0]$ | 51.15 | 42.34 | 20.42 | 37.97 |
| $\lambda_0 \cdot \phi_t$ | $[0, \lambda_0]$ | 51.16 | 42.40 | 21.27 | 38.28 |
| $\lambda_0 \cdot \phi_t^2$ | $[0, \lambda_0]$ | **51.19** | **42.42** | **21.36** | **38.32** |

Table E.4: Comparison with different expressions for $\lambda_{\mathcal{F}}$ across all CCC levels using accuracy (%).

| $\mu_{\mathcal{C}}$ | Range | Easy | Medium | Hard | Mean |
|---|---|---|---|---|---|
| $1-\mu_0$ | $\{1-\mu_0\}$ | 50.82 | 41.86 | 20.70 | 37.79 |
| $\phi_t$ | $[0, 1]$ | 51.48 | 42.42 | 0.31 | 31.40 |
| $1-\mu_0 \cdot \phi_t$ | $[1, 1-\mu_0]$ | 51.17 | 42.47 | 4.07 | 32.57 |
| $1-\mu_0 \cdot (1-\phi_t^2)$ | $[1-\mu_0, 1]$ | 51.14 | 42.40 | 21.11 | 38.22 |
| $1-\mu_0 \cdot (1-\phi_t)$ | $[1-\mu_0, 1]$ | 51.19 | 42.42 | 21.36 | **38.32** |

Table E.5: Comparison with different expressions for $\mu_{\mathcal{C}}$ across all CCC levels using accuracy (%).

### E.5 HYPERPARAMETER SENSITIVITY

Since validation sets are not available in TTA, tuning hyperparameters optimally is challenging. Instead, we tune hyperparameters using only 5% of a holdout set (transition speed 2000; random seed 44) from CCC-Hard. In addition, we demonstrate that our method is less sensitive to hyperparameter changes. We evaluate performance across all levels of CCC, slightly modifying the tuned values; the standard values are provided in Table C.2. Fig. E.1 demonstrates the effectiveness of our method in terms of robustness to hyperparameter variations. It also should be noted that the slight performance differences, observed in the figure below, are negligible. Finally, the use of the same hyperparameter settings across all benchmarks further highlights the advantage of our method.

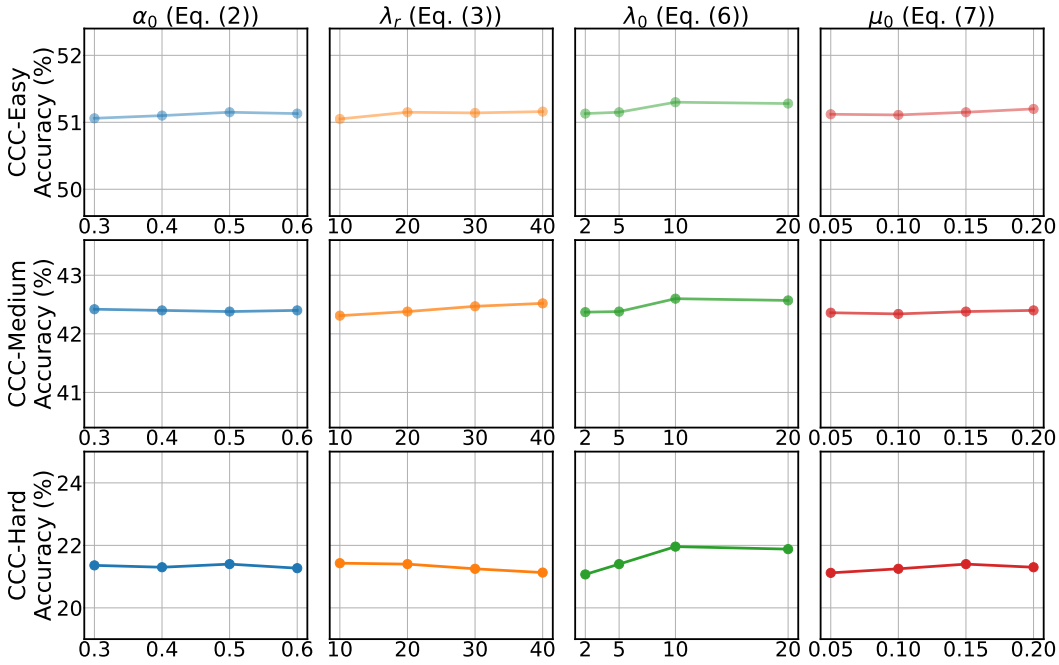


Figure E.1: Hyperparameter sensitivity analysis.

### E.6 Effect of Knowledge Recovery

Theoretically, our proposed regularizer can be seen to recover essential knowledge lost due to resets. This theoretical grounding stems from two key mechanisms. First, we accumulate updated parameters using a combination of CMA and EMA, preserving adaptation information in a manner similar to Polyak averaging (Polyak & Juditsky, 1992), which provides a reliable reference for previously acquired knowledge. Second, the Fisher-based regularization follows the principle of Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), assigning stronger penalties to parameters that are important for prior domains. Together, these mechanisms encourage important parameters to remain close to their pre-reset values, effectively restoring knowledge that would otherwise be lost.

We integrate several components to complement each other. In particular, the knowledge recovery module is introduced in Sec. 3.4 to effectively restore information erased by resets. We evaluate its effectiveness under the same setup as Table D.6 by measuring how much knowledge from previous domains is recovered. Knowledge recovery is measured as the gap between the current performance and the best performance achieved so far for each domain, which is then averaged across domains. Positive values indicate recovery, while negative values indicate forgetting. As shown in Table E.6, our method consistently recovers substantial knowledge without forgetting. For instance, at revisit #10, ETA+ASR achieves 0.58 compared to -1.94 without recovery, and ROID+ASR achieves 0.16 compared to -0.52 without recovery. This confirms that the recovery module effectively compensates for knowledge erasure from reset. Note that knowledge refers to information encoded in the model weights accumulated during adaptation, which correspond to $\bar{\theta}$ in Eq. (4). Essential knowledge is identified via Fisher information, which highlights weights that are more informative about previous domains. Direct quantification for knowledge is challenging; therefore, we use task performance as a proxy to assess it.

| Recovery (Revisit#) | 1 | ... | 10 | 15 | 20 | Mean |
|---|---|---|---|---|---|---|
| ETA + ASR (Ours) | 0.0 | ... | +0.58 | +0.08 | +0.02 | +0.24 |
| + w/o knowledge recovery | 0.0 | ... | -1.94 | -1.16 | -0.76 | -0.56 |
| ROID + ASR (Ours) | 0.0 | ... | +0.16 | +0.24 | +0.01 | +0.12 |
| + w/o knowledge recovery | 0.0 | ... | -0.52 | -0.42 | -0.10 | -0.14 |

Table E.6: Knowledge recovery measured across multiple revisits on IN-C.

Additionally, we evaluate the effectiveness of knowledge recovery through accuracy. We also use a domain-recurring setting on IN-C, where the same domain reappears multiple times, to test whether a model preserves previously learned information even though it has been reset. We compare our method with a variant without the knowledge recovery module (Sec. 3.4). As shown in Table E.7, the variant without the recovery module gradually declines in accuracy across later revisits, while our method consistently maintains its performance, demonstrating that the recovery module effectively mitigates the forgetting of prior domains' knowledge.

| Accuracy (Revisit#) | 1 | ... | 10 | 15 | 20 | Mean |
|---|---|---|---|---|---|---|
| ETA | 30.64 | ... | 36.16 | 35.96 | 35.80 | 35.88 |
| + ASR (Ours) | 28.68 | ... | 38.60 | 38.97 | 39.10 | 36.90 |
| + w/o knowledge recovery | 28.64 | ... | 37.45 | 36.49 | 36.34 | 36.56 |
| ROID | 35.32 | ... | 38.08 | 38.15 | 38.02 | 37.96 |
| + ASR (Ours) | 35.66 | ... | 42.20 | 42.06 | 42.96 | 41.56 |
| + w/o knowledge recovery | 35.35 | ... | 41.64 | 41.64 | 41.19 | 40.96 |

Table E.7: Performance comparison across multiple revisits on IN-C.

The benefit of knowledge recovery appears negligible because evaluation in Table E.7 is conducted under an easy-to-adapt setting. However, its benefit is not negligible in challenging adaptation scenarios. Indeed, Table E.7 confirms that the knowledge recovery module is functioning as intended, but IN-C is not an appropriate benchmark for measuring its performance contribution. As described in Sec. 3.5, under challenging adaptation scenarios, we increase the regularization coefficient to encourage the model to reuse prior-domain information, thereby enhancing the effect of the knowledge recovery module. However, IN-C is relatively easy to adapt to. Table 1 also shows that baseline accuracies are very similar in IN-C, so the benefit of knowledge recovery does not manifest strongly in this setting.

We consider CCC-Hard to illustrate the recovery module's contribution. In several splits (e.g., 4, 7, and 8), removing the knowledge recovery module leads to substantial accuracy drops, while the full model consistently maintains higher accuracy. These observations indicate that the module functions flexibly, providing effective support under challenging domain shifts.

| Acc. (Split#) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| ROID | 12.64 | 15.79 | 13.28 | 9.63 | 12.65 | 11.81 | 10.66 | 8.72 | 17.12 |
| + ASR (Ours) | 20.99 | 22.51 | 20.40 | 21.22 | 22.93 | 21.36 | 22.37 | 23.84 | 24.25 |
| + w/o recovery | 20.95 | 22.50 | 20.35 | 18.28 | 22.69 | 20.18 | 9.70 | 15.67 | 23.57 |

Table E.8: Performance comparison across nine splits in CCC-Hard.

# F Additional Analysis

## F.1 Limitations of Full-Parameter Reset

**a) Performance drops.** We measure post-reset performance drops for RDumb on CCC-Hard under the same setup as Fig. 1 to demonstrate the limitation of full-parameter reset. We compute the change in average accuracy by comparing 10 batches before and after each reset, and then average these values over all reset points. RDumb exhibits an average 1.26%p drop per reset, which corresponds to roughly 12% of its overall average accuracy (9.77%). This confirms that RDumb's degradation at each reset is non-trivial.

**b) Recovery delays.** To measure recovery delays after a reset, we count how many batches RDumb requires to reach the highest accuracy observed in the reset-preceding 20 batches. When full recovery does not occur before the next reset, we count all batches until that reset. On average, RDumb requires 330 batches to recover, while it resets every 1000 batches. Therefore, RDumb takes substantially long to regain its pre-reset performance, which highlights the inefficiency of its full-parameter reset mechanism.

## F.2 Risk of Proximity to Reset

As we noted, proximity to reset potentially compromises parameter integrity and ultimately harms adaptation. We empirically demonstrate this risk by slightly delaying resets, which allows corrupted parameters to accumulate in $\bar{\theta}$ from Eq. (4). Under recurring scenarios (IN-C), we observe harmful effects when corrupted domain information is re-utilized. Normally, resets have been triggered when $\mathcal{C}_t > \bar{\mathcal{C}}_{t-1}$. For the delayed variant, we postpone the resets until $\mathcal{C}_t - \bar{\mathcal{C}}_{t-1} > \epsilon$, retaining parameters beyond the standard reset points. As shown in Table F.1, delaying resets leads to substantial performance drops, even below ETA, confirming that parameters are particularly vulnerable to corruption after the standard reset points, and that such corruption significantly impairs adaptation.

| IN-C (Revisit#) | $\epsilon$ | 1 | 5 | 10 | 15 | 20 | Mean |
|---|---|---|---|---|---|---|---|
| ETA | - | 30.64 | 36.76 | 36.16 | 35.96 | 35.80 | 35.88 |
| + ASR (Ours) | 0.0 | 28.68 | 33.00 | 38.60 | 38.97 | 39.10 | 36.90 |
| + w/ delay | 0.001 | 28.42 | 33.39 | 37.80 | 38.82 | 39.02 | 36.25 |
| + w/ delay | 0.01 | 27.94 | 28.06 | 27.94 | 28.30 | 28.12 | 28.29 |
| ROID | - | 35.32 | 38.00 | 38.08 | 38.15 | 38.02 | 37.96 |
| + ASR (Ours) | 0.0 | 35.66 | 41.03 | 42.20 | 42.06 | 42.96 | 41.56 |
| + w/ delay | 0.001 | 35.08 | 40.01 | 41.42 | 41.96 | 41.54 | 40.85 |
| + w/ delay | 0.01 | 35.60 | 37.78 | 38.28 | 38.61 | 38.62 | 38.07 |

Table F.1: Performance on IN-C with and without delayed resets. $\epsilon$ indicates the delay threshold.

### F.3 Fair Comparison for Reset

We compare our reset mechanism with existing reset mechanisms, proposed by SAR (Niu et al., 2023), RDumb (Press et al., 2023), and DA-TTA (Wang et al., 2024), with ROID across all CCC levels, as demonstrated in Table F.2. They reset all model parameters periodically (RDumb), and only when extremely high confidence (SAR) or a significant distribution discrepancy from the source (DA-TTA) is detected. For our approach (ASR), we isolate other components except for our reset mechanism for a fair comparison; otherwise results are reported as 51.19%, 42.42%, and 21.36% for CCC-Easy, -Medium, and -Hard. Existing approaches, except for SAR, improve performance on CCC-Hard but degrade it on the other levels. However, our approach consistently outperforms the others, surpassing the second-best by +2.8%p on average.

| Method | Easy | Medium | Hard | Mean |
|---|---|---|---|---|
| ROID | 49.74 | 40.19 | 11.81 | 33.91 |
| + SAR | 49.73 | 40.06 | 5.29 | 31.69 |
| + RDumb | 49.56 | 39.77 | 14.06 | 34.46 |
| + DA-TTA | 45.98 | 35.76 | 15.53 | 32.42 |
| + ASR (Ours) | **50.70** | **41.72** | **19.36** | **37.26** |

Table F.2: Performance comparison across reset mechanisms on CCC levels.

### F.4 Robustness to Truly Small Batch Sizes

We evaluate our method on truly small batch sizes, specifically 2 and 4, on a single split (transition speed 1000; random seed 43) of CCC-Easy with ROID as our base model, following the setting of Fig. 10. As shown in Table F.3, our method consistently outperforms baselines. At batch size 4, ASR achieves 25.58, compared to 17.85 for RDumb, demonstrating that its robustness extends to smaller batch sizes than 16. However, at batch size 2, the gap between ASR and RDumb narrows, as our reset mechanism requires a minimum number of samples to function effectively. Please note that, online TTA and continual TTA are different settings, and our focus is on a variation of the latter one: long-term continual TTA. Online TTA is an extreme scenario with the batch size of 1, and most TTA methods fail to work under such an extreme condition. All methods including ASR yield near-random performance ($\sim$0.1). One practical approach for ASR in this setting is to temporarily store online samples and evaluate the reset criterion once enough samples are collected.

| Batch size | 2 | 4 |
|---|---|---|
| ROID | 0.13 | 16.91 |
| + RDumb | 5.87 | 17.85 |
| + ASR (Ours) | 6.46 | 25.58 |

Table F.3: Performance comparison for truly small batch sizes