# Evaluation of Test-Time Adaptation Under Computational Time Constraints

**Motasem Alfarra** [1 2] **Hani Itani** [1] **Alejandro Pardo** [1] **Shyma Alhuwaider** [1] **Merey Ramazanova** [1] **Juan C. Pérez** [1]
**Zhipeng Cai** [2] **Matthias Müller** [2] **Bernard Ghanem** [1]

## Abstract

This paper proposes a novel online evaluation protocol for Test Time Adaptation (TTA) methods, which penalizes slower methods by providing them with fewer samples for adaptation. TTA methods leverage unlabeled data at test time to adapt to distribution shifts. Although many effective methods have been proposed, their impressive performance usually comes at the cost of significantly increased computation budgets. Current evaluation protocols overlook the effect of this extra computation cost, affecting their real-world applicability. To address this issue, we propose a more realistic evaluation protocol for TTA methods, where data is received in an online fashion from a constant-speed data stream, thereby accounting for the method's adaptation speed. We apply our proposed protocol to benchmark several TTA methods on multiple datasets and scenarios. Extensive experiments show that, when accounting for inference speed, simple and fast approaches can outperform more sophisticated but slower methods. For example, SHOT from 2020, outperforms the state-of-the-art method SAR from 2023 in this setting. Our results reveal the importance of developing practical TTA methods that are both accurate and efficient[1].

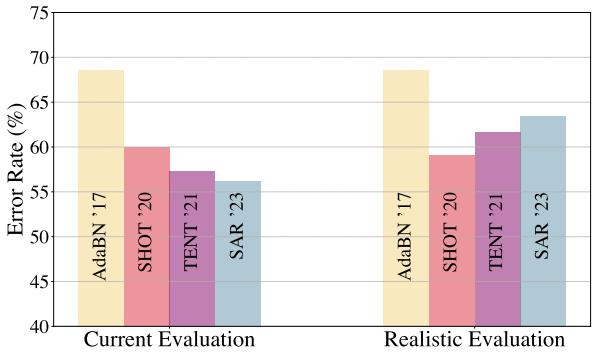



Figure 1: **The trend of average error rate using *offline* evaluation *vs* our proposed *online* evaluation.** In the offline setup, TTA methods demonstrate progress across time with a decreasing average error rate, *e.g.* from $68.5\%$ using AdaBN to $56.2\%$ using SAR. We propose a realistic evaluation protocol that accounts for the adaptation speed of TTA methods. Under this protocol, fast methods (*e.g.* AdaBN) are unaffected, while slower (but more recent and sophisticated) methods (*e.g.* SAR) are penalized.

## 1. Introduction

In recent years, Deep Neural Networks (DNNs) have demonstrated remarkable success in various tasks (He et al., 2016) thanks to their ability to learn from large datasets (Deng et al., 2009). However, a significant limitation of DNNs is their poor performance when tested on out-of-distribution data, which violates the i.i.d. assumption that the training and testing data are from the same distribution (Hendrycks et al., 2021; Hendrycks & Dietterich, 2019; Kar et al., 2022). Such failure cases are concerning, since distribution shifts are common in real-world applications, *e.g.*, image corruptions (Hendrycks & Dietterich, 2019), changing weather conditions (Sakaridis et al., 2021), or security breaches (Goodfellow et al., 2014).

Test Time Adaptation (TTA) (Saenko et al., 2010; Sun et al., 2020; Liu et al., 2021) has demonstrated promising results for solving the above problem. TTA leverages the unlabeled data that arrives at test time by adapting the forward pass of pre-trained DNNs according to some proxy task (Liang et al., 2020; Lee et al., 2013). Though recent methods have made significant progress at improving accuracy under distribution shifts (Wang et al., 2020; Niu et al., 2022; Gao et al., 2022), many of them incur high computational overhead. For instance, some methods require self-supervised fine-tuning on the data (Chen et al., 2022), while others perform a diffusion process per input (Gao et al., 2022).

The computational overhead of TTA methods decreases

[1]King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia [2]Intel Labs, Munich, Germany. Correspondence to: Motasem Alfarra <motasem.alfarra@kaust.edu.sa>.

*Proceedings of the $41^{st}$ International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. 

[1]Code: github/MotasemAlfarra/Online-Test-Time-Adaptation

their inference speed, which is a critical property in many real-world applications that require the TTA method to produce predictions at the speed of the stream itself. This property, however, is overlooked in the current evaluation protocols for TTA methods. In particular, these protocols assume a setting, which neglects how events constantly unfold regardless of the model's speed, causing the model to miss incoming samples when it is busy processing previous ones. For TTA methods that adapt using test data, missing samples has a direct effect on the method's accuracy, as it will have fewer samples for adaptation. That is, the slower the TTA method, the fewer samples it can leverage for adapting to the distribution shift. Thus, the current protocol for evaluating TTA methods is not suitable for assessing their efficacy in real-world deployment.

In this work, we propose a novel *realistic* evaluation protocol that factors in inference speed to assess the real-world applicability of TTA methods. Our evaluation protocol is inspired by Online Learning (Cai et al., 2021; Shalev-Shwartz et al., 2012) and mimics real-world scenarios by exposing all TTA methods to a constant-speed stream of data. In this setting, the performance of slow TTA methods is intrinsically penalized, as the time spent adapting to a sample may lead to dropped samples that could have been useful for adaptation. Specifically, our protocol dictates that if a method $g_{\text{slow}}$ is $k$ times slower than the stream, then it may only use every $k^{\text{th}}$ sample for adaptation. In contrast, a method $g_{\text{fast}}$ that is as fast as the stream is allowed to adapt to every sample. Figure 1 shows the effect of evaluating several methods under our proposed protocol, where slower methods (*e.g.*, SAR (Niu14 et al., 2023)) are penalized and faster but simpler methods become better alternatives (*e.g.*, SHOT (Liang et al., 2020) and AdaBN (Li et al., 2016)).

We apply our proposed evaluation protocol to benchmark several TTA methods on multiple datasets, and provide a fair assessment of their performance subject to the realistic consequences of slower inference speeds. Our experimental results highlight the importance of developing TTA methods that adapt to distribution shifts with minimal impact on inference speed. Our contributions are two-fold:

1. We propose a realistic evaluation protocol for TTA methods that penalizes slower methods by providing them with fewer samples for adaptation. Our approach is effective at assessing TTA methods' efficacy in scenarios where data arrives as a constant-speed stream.
2. Following our proposed protocol, we provide a comprehensive experimental analysis of 15 TTA methods evaluated on 3 large-scale datasets under 3 different evaluation scenarios. These scenarios consider adaptation to a single domain and continual adaptation to several domains. Our analysis shows that, when inference speed is accounted for, simple (but faster) approaches can benefit from adapting to more data, and thus outperform more sophisticated (but slower) methods. Figure 1 demonstrates this for four TTA methods. We hope our evaluation scheme inspires future TTA methods to consider inference speed as a critical dimension that affects their real-world performance.

## 2. Related Work

**Test Time Adaptation.** The Test Time Adaptation (TTA) setup relaxes the "i.i.d" assumption between the training and testing distributions (Sun et al., 2020; Boudiaf et al., 2022). This relaxation is usually attained through a lifelong learning scheme on all received unlabeled data (Chen et al., 2022; Gong et al.). Earlier approaches such as TTT (Sun et al., 2020) and TTT++ (Liu et al., 2021), among others (Torralba & Efros, 2011; Tzeng et al., 2017), include a self-supervised loss (Gidaris et al., 2018) during training, which can then provide an error signal during adaptation. Despite their effectiveness, such approaches assume having control over how the model is trained.

**Fully Test Time Adaptation.** Fully TTA methods are a subtype of TTA method that adapts at test time by modifying the model's parameters (Liang et al., 2020; Lee et al., 2013; Mirza et al., 2022b; Mancini et al., 2018; Kojima et al., 2022) or its input (Gao et al., 2022) by using the incoming unlabeled data. Fully TTA methods are practical, as they avoid assumptions on the training phase of a given model (Wang et al., 2020; Gao et al., 2022; Iwasawa & Matsuo, 2021). The first of these approaches adjusts the statistics of the Batch Normalization (BN) layers (Mirza et al., 2022a; Schneider et al., 2020; Li et al., 2016). For example, BN-adaptation (Schneider et al., 2020) leverages the statistics of the source data as a prior and infers the statistics for every received sample. On the other hand, AdaBN (Li et al., 2016) discards the statistics of the source domain and uses the statistics computed on the target domain. In line with light TTA methods, LAME (Boudiaf et al., 2022) proposes to only adapt the model's output by finding the latent assignments that optimize a manifold-regularized likelihood of the data. In this work, we found that such efficient methods preserve their accuracy under our proposed evaluation. While fully TTA methods have been studied in the context of adversarial domain shifts (Alfarra et al., 2022; Croce et al., 2022; Pérez et al., 2021), in this work we focus on the context of natural shifts such as realistic image corruptions (Hendrycks & Dietterich, 2019; Kar et al., 2022).

Another line of work aims at adapting to distribution shifts by minimizing entropy. For instance, SHOT (Liang et al., 2020) adapts the feature extractor to minimize the entropy of individual predictions; while maximizing the entropy of the predicted classes. TENT (Wang et al., 2020) updates the learnable parameters of the BN layers to minimize the

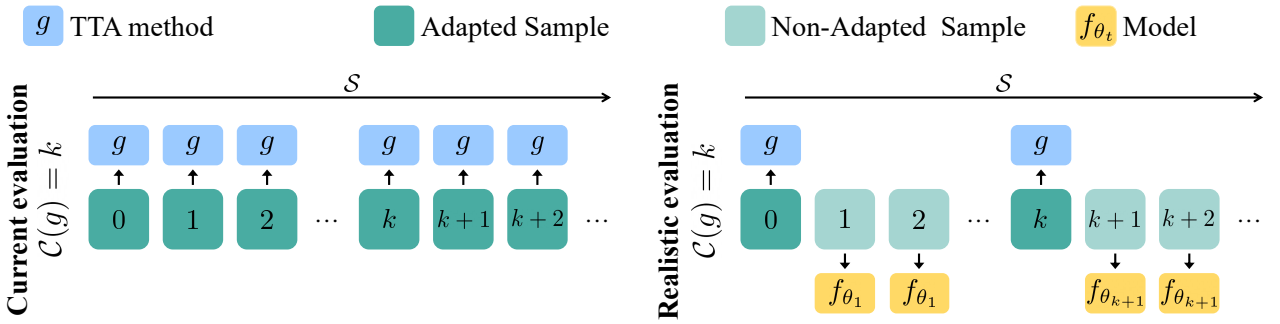



Figure 2: **Inference under the current and realistic evaluation protocols.** The current evaluation setting (left) assumes that the incoming batches of stream $\mathcal{S}$ can wait until the adaptation process of a TTA method $g$ finishes. This assumption is untenable in a real-time deployment scenario. Our proposed realistic evaluation (right) simulates a more realistic scenario where $\mathcal{S}$ reveals data at a constant speed. In this setup, slower TTA methods will adapt to a smaller portion of the stream. The remaining part of the stream will be predicted without adaptation by employing the most recent adapted model. We refer to the most recent adapted model as $f_{\theta_{t+1}}$, with $t$ denoting the time when the last sample was adapted to by $g$. When $g$ is still adapting to a sample, the incoming sample is fed to $f_{\theta_{t+1}}$ to produce predictions.

entropy of predictions. EATA (Niu et al., 2022) combines TENT with an active selection of reliable and non-redundant samples from the target domain and an anti-forgetting loss (Kirkpatrick et al., 2017). Further, SAR (Niu14 et al., 2023) equips TENT with an active sampling scheme that filters samples with noisy gradients.

Other works use data-augmentation at test time (Ashukha et al., 2020). For example, MEMO (Zhang et al., 2021) adapts model parameters to minimize the entropy over a sample and multiple augmentations of it. CoTTA (Wang et al., 2022) uses augmentations to generate reliable pseudo-labels and then peform distillation. Finally, DDA (Gao et al., 2022) proposes to leverage a diffusion model (Ho et al., 2020) to restore corrupted inputs back to the source data distribution. These methods require multiple forward passes through the network or a diffusion model, leading to slower inference speeds.

## 3. Methodology

In this section, we present our proposed Realistic TTA evaluation protocol. We first describe the current TTA evaluation protocol and its limitations Then, we introduce our Realistic TTA evaluation protocol, which addresses the shortcomings of the offline protocol.

### 3.1. Current Protocol

TTA considers the practical setup, in which trained models are deployed in a target domain that exhibits distribution shifts to which they must adapt. Let $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ be a classifier, parameterized by $\theta$, that predicts the label $y \in \mathcal{Y}$ for a given input $x \in \mathcal{X}$. Before test time, $f_\theta$ is assumed to have been trained on the dataset $\mathcal{D}_{\text{train}} \subset \mathcal{X} \times \mathcal{Y}$. At test time, *i.e.* when executing TTA, $f_\theta$ is presented with a stream of data $\mathcal{S}$, sampled from $\mathcal{X}$, with potentially multiple distribution shifts w.r.t. $\mathcal{D}_{\text{train}}$. Under this setup, a TTA method is a function $g(\theta, x)$ that sequentially adapts the model's parameters $\theta$ and/or the input $x$ to enhance the performance under distributions shifts. Currently, TTA methods are evaluated in an offline setting.

Formally, the *Current TTA evaluation protocol* simulates the interaction between the stream $\mathcal{S}$ and the TTA method $g$, at each time step $t \in \{0, 1, \ldots, \infty\}$, as follows:

**Curr.1** $\mathcal{S}$ reveals a sample $x_t$.

**Curr.2** $g$ adapts $x_t$ to $\hat{x}_t$, $\theta_t$ to $\hat{\theta}_t$, generates prediction $\hat{y}_t$, and updates parameters $\theta_{t+1} = \alpha\theta_t + (1-\alpha)\hat{\theta}_t$.[2]

Note that all existing TTA methods can be modeled using this framework. For example, TENT (Wang et al., 2020) adapts network parameters to minimize entropy with $\alpha = 0$, while leaving inputs unchanged, *i.e.* $\hat{x}_t = x_t$ and $\theta_{t+1} = \hat{\theta}_t$. DDA (Gao et al., 2022) adapts inputs via a diffusion process while preserving network parameters with $\alpha = 1$, *i.e.* $\hat{x}_t = \hat{x}_t$ and $\theta_{t+1} = \theta_t$. CoTTA (Wang et al., 2022) applies knowledge distillation, and updates network parameters with an exponential moving average, *i.e.* setting $0 < \alpha < 1$.

**Shortcomings of the Current TTA protocol.** In the current protocol, the performance of a TTA method $g$ is measured by comparing the ground truth labels $y_t$ with the predictions after adaptation $\hat{y}_t$. An evaluation based only on this measure implicitly assumes that the stream is not constant

[2]Note that some methods abstain from adapting either $x_t$ or $\theta_t$.

speed, but rather waits for $g$ to adapt to $x_t$ (**Curr.2**) before revealing the next batch $x_{t+1}$ (**Curr.1**). Figure 2 provides an illustration of this situation. This assumption results in the offline protocol favoring slower TTA methods, as the method's performance is agnostic to its inference speed. However, in practical applications where the test data arrives at a constant speed, the offline protocol is not suitable for assessing a method's performance. Next, we propose a remedy for this shortcoming.

### 3.2. Realistic Online Evaluation Protocol

We propose a realistic evaluation of TTA methods that *explicitly* considers the relation between the speed of the method and the speed at which the stream reveals new data. This setup is more realistic, as it intrinsically penalizes the performance of slower TTA methods: *long times spent in adaptation result in fewer samples to adapt to*.

A crucial aspect of our realistic TTA protocol is accounting for the implications of simulating a constant speed data stream $\mathcal{S}$. For instance, consider a stream $\mathcal{S}$ that reveals data at a constant rate $r$ samples per second. If a method $g_{\text{fast}}$ adapts to samples at speed $r$, then $g_{\text{fast}}$ will be able to adapt to every sample. On the other hand, if $g_{\text{slow}}$ adapts to samples at a speed $r/2$, then $g_{\text{slow}}$ will skip every other sample. We formalize the notion of the relation between the speed of the stream and the speed of a method $g$ as the "relative adaptation speed of $g$". This quantity, denoted by $\mathcal{C}(g) \in \mathbb{N}$, is simply the integer ratio of the speed of $\mathcal{S}$ to the speed of $g$. For instance, in the previous example, $\mathcal{C}(g_{\text{fast}}) = 1$, meaning $g_{\text{fast}}$ adjusts as fast as $\mathcal{S}$ reveals data, while $\mathcal{C}(g_{\text{slow}}) = 2$, indicating $\mathcal{S}$ reveals its second batch while $g_{\text{slow}}$ is still adapting to the first one.

Without loss of generality, we assume that $f_\theta$ runs in real-time, *i.e.* that its speed is equal to $r$, and thus $\mathcal{C}(f_\theta) = 1$. This assumption allows us to suppose that the samples that are not processed by $g$ can be processed by $f_\theta$. Under this setup, we define our realistic protocol by introducing the relative adaptation speed $\mathcal{C}(g)$ into the offline protocol. In particular, we simulate $g$'s availability by *conditionally* performing the adaptation step (**Curr.2**), depending on $\mathcal{C}(g)$. In this manner, $g$ is only permitted to adapt when its previous adaptation step has finished. Formally, the *realistic TTA evaluation protocol* simulates the interaction between the constant speed stream $\mathcal{S}$ and the TTA method $g$, at each time step $t \in \{0, 1, \dots, \infty\}$, as follows:

**RTTA 1** $\mathcal{S}$ reveals a sample $x_t$.

**RTTA 2** If $(t \mod \mathcal{C}(g)) = 0$, then $g$ adapts $x_t$ to $\hat{x}_t$, $\theta_t$ to $\hat{\theta}_t$, generates a prediction $\hat{y}_t$, and updates parameters via $\theta_{t+1} \leftarrow \alpha\theta_t + (1-\alpha)\hat{\theta}_t$. Otherwise, $f_{\theta_t}$ generates a prediction $\hat{y}_t$.

Table 1: **Average $\mathcal{C}(g(x_t))$.** We report the average relative adaptation speed $\mathcal{C}(g)$ for 5 TTA methods. The higher $\mathcal{C}(g)$ is, the smaller the portion of data to which $g$ adapts is.

| Method | AdaBN | TENT | TTAC-NQ | MEMO | DDA |
|---|---|---|---|---|---|
| $\mathcal{C}(g)$ | 1 | 3 | 12 | 54 | 810 |

Here, "mod" represents the modulo operation. The above protocol assesses the performance of TTA methods by factoring in their speed. As such, faster methods are granted more adaptation steps and, conversely, slower methods are granted fewer (see Figure 2). Note that explicitly modeling the relative adaptation speeds allows us to evaluate TTA methods under different adaptation speeds by setting $\mathcal{C}(g)$ to arbitrary values. For instance, note that our realistic protocol recovers the original offline protocol by setting $\mathcal{C}(g) = 1$ for all methods. Next, we explain the calculation of $\mathcal{C}(g)$ for our realistic protocol.

**Online computation of $\mathcal{C}(g)$.** In practice, estimating the relative adaptation speed $\mathcal{C}(g)$ can be a noisy process. The noise in this estimation essentially comes from two factors: hardware and input dependence. Hardware-induced noise applies to all methods, while input dependence applies to methods like ETA (Niu et al., 2022) which, upon receiving an input, may optionally abstain from adapting to it. This noise means that $\mathcal{C}(g)$ potentially varies across iterations.

Our protocol accounts for this variability by conducting an online computation of $\mathcal{C}(g)$ on each revealed input. That is, instead of using a fixed value of $\mathcal{C}(g)$ at each iteration $t$, our protocol rather uses $\mathcal{C}\left(g(x_t)\right)$. Formally, if we let $R\left(g(x)\right)$ denote the speed at which $g$ processes $x$, then the relative adaptation speed of $g$ at $x$ is defined as $\mathcal{C}\left(g(x_t)\right) = \lceil r / R(g(x)) \rceil$, where the ceiling function accounts for the stream's discrete-time nature. Note that since we assumed $\mathcal{C}(f_\theta) = 1$, then $R\left(f_\theta(x)\right) = r$. We report the empirical behavior of this online computation of $\mathcal{C}\left(g(x_t)\right)$ for various TTA methods in Table 1, and leave the rest of the methods and the computation details to the Appendix. Next, we leverage our Realistic TTA protocol to conduct a comprehensive empirical study of several TTA methods.

## 4. Experiments

We follow prior art (Wang et al., 2020; Niu14 et al., 2023; Gao et al., 2022) and focus on the task of image classification. In all our experiments, we assume that $f_\theta$ is a ResNet-50-BN[3] (He et al., 2016) trained on ImageNet (Deng et al., 2009) (pretrained weights obtained from `torchvision`). We further assume that the stream $\mathcal{S}$ reveals batches of size

[3]SAR demonstrated the superiority of using batch independent normalization layers under batch size of 1. We leave this ablation to the Appendix along with experiments on other architectures.

Table 2: **Episodic Error Rate on ImageNet-C.** We report the error rate of different TTA methods on ImageNet-C benchmark under both the realistic and the current setup. A lower error rate indicates a better TTA method. The highlighted numbers indicate a better performance per method across setups. Episodic means the model will adapt to one corruption at a time. The model is reset back to the base model when moving to the next corruption. The current setup is merely the reproduction of every method. The first sub-table corresponds to methods that do not incur any or few extra computations, *i.e.* $\mathcal{C}(g) = 1$. We show that methods generally perform worse in the realistic setup. The more computationally complex the TTA method is, the less data it will adapt to, and the worse is its performance.

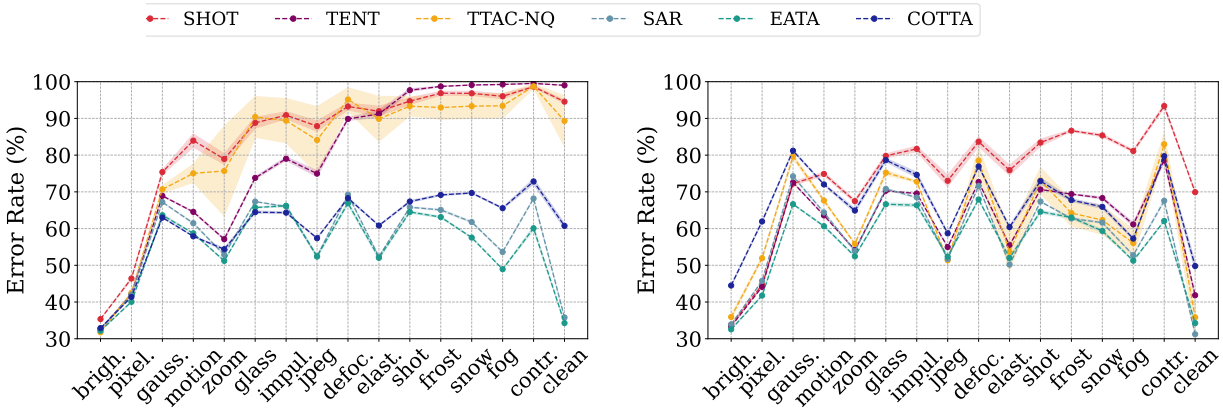
| Method | Realistic | Noise | | | Blur | | | | Weather | | | | Digital | | | | Avg. | Δ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | gauss. | shot | impul. | defoc. | glass | motion | zoom | snow | frost | fog | brigh. | contr. | elast. | pixel. | jpeg | | |
| Source | ✓ | 97.8 | 97.1 | 98.1 | 82.1 | 90.2 | 85.2 | 77.5 | 83.1 | 76.7 | 75.6 | 41.1 | 94.6 | 83.0 | 79.4 | 68.4 | 82.0 | - |
| AdaBN | ✓ | 84.9 | 84.3 | 84.3 | 85.0 | 84.7 | 73.6 | 61.1 | 65.8 | 66.9 | 52.1 | 34.8 | 83.3 | 56.1 | 51.1 | 60.3 | 68.5 | - |
| LAME | ✓ | 98.3 | 97.6 | 98.6 | 82.4 | 90.9 | 86.1 | 78.1 | 84.5 | 77.5 | 77.3 | 41.4 | 94.8 | 84.8 | 80.0 | 68.9 | 82.7 | - |
| BN | ✓ | 84.6 | 83.9 | 83.8 | 80.1 | 80.2 | 71.7 | 60.4 | 65.4 | 65.2 | 51.6 | 34.6 | 76.3 | 54.4 | 49.7 | 59.2 | 66.7 | - |
| SHOT | ✗ | 73.4 | 70.2 | 73.0 | 76.6 | 75.5 | 59.8 | 53.8 | 54.2 | 63.4 | 44.7 | 35.5 | 79.3 | 46.9 | 43.2 | 49.7 | 59.9 | (-0.8) |
| | ✓ | 73.6 | 69.0 | 71.1 | 74.6 | 74.8 | 60.0 | 52.9 | 54.1 | 61.3 | 44.1 | 34.1 | 77.8 | 46.8 | 43.1 | 49.2 | 59.1 | |
| TENT | ✗ | 71.3 | 69.4 | 70.2 | 72.0 | 72.9 | 58.7 | 50.7 | 52.8 | 58.8 | 42.7 | 32.7 | 73.3 | 45.5 | 41.5 | 47.7 | 57.3 | (+4.3) |
| | ✓ | 75.7 | 78.3 | 75.2 | 76.3 | 77.3 | 64.6 | 55.6 | 57.3 | 61.4 | 45.9 | 33.5 | 77.1 | 50.1 | 44.2 | 51.4 | 61.6 | |
| SAR | ✗ | 69.5 | 69.7 | 69.0 | 71.2 | 71.7 | 58.1 | 50.5 | 52.9 | 57.9 | 42.7 | 32.7 | 62.9 | 45.5 | 41.6 | 47.8 | 56.2 | (+7.2) |
| | ✓ | 79.4 | 78.5 | 78.1 | 79.9 | 79.3 | 67.5 | 56.1 | 60.5 | 63.1 | 47.4 | 34.0 | 75.3 | 51.7 | 46.6 | 53.8 | 63.4 | |
| CoTTA | ✗ | 78.4 | 77.8 | 77.2 | 80.5 | 79.1 | 64.0 | 53.3 | 57.8 | 60.7 | 44.1 | 32.9 | 73.1 | 48.6 | 42.3 | 52.6 | 61.5 | (+6.5) |
| | ✓ | 82.9 | 81.6 | 81.9 | 87.4 | 85.6 | 75.6 | 61.1 | 63.1 | 64.9 | 49.9 | 34.8 | 91.2 | 54.0 | 48.8 | 56.6 | 68.0 | |
| TTAC-NQ | ✗ | 71.3 | 70.3 | 70.8 | 82.1 | 77.4 | 63.9 | 53.9 | 49.9 | 55.5 | 43.9 | 32.8 | 81.4 | 43.7 | 41.1 | 46.7 | 59.0 | (+7.5) |
| | ✓ | 79.4 | 75.7 | 78.9 | 86.6 | 86.2 | 77.1 | 61.8 | 58.8 | 62.4 | 51.5 | 34.4 | 88.5 | 52.1 | 49.1 | 55.5 | 66.5 | |
| EATA | ✗ | 65.5 | 62.4 | 63.5 | 66.6 | 67.2 | 52.0 | 47.3 | 48.2 | 54.1 | 39.9 | 32.1 | 55.0 | 42.3 | 39.2 | 44.8 | 52.0 | (+3.6) |
| | ✓ | 69.3 | 67.1 | 69.2 | 71.1 | 71.7 | 57.5 | 49.9 | 51.9 | 57.4 | 42.4 | 32.6 | 60.7 | 45.1 | 41.4 | 47.4 | 55.6 | |
| MEMO | ✗ | 92.5 | 91.3 | 91.0 | 84.0 | 87.0 | 79.3 | 72.4 | 74.6 | 71.3 | 67.9 | 39.0 | 89.0 | 76.2 | 67.0 | 62.4 | 76.3 | (+5.6) |
| | ✓ | 97.7 | 97.0 | 98.0 | 82.1 | 90.1 | 85.1 | 77.4 | 83.0 | 76.6 | 75.4 | 41.0 | 94.5 | 82.9 | 79.2 | 68.2 | 81.9 | |
| DDA | ✗ | 58.6 | 57.8 | 59.0 | 87.0 | 81.6 | 76.6 | 65.9 | 67.9 | 66.7 | 64.0 | 40.0 | 92.2 | 52.2 | 46.6 | 49.9 | 64.4 | (+17.6) |
| | ✓ | 97.8 | 97.0 | 98.1 | 82.1 | 90.2 | 85.2 | 77.5 | 83.1 | 76.7 | 75.6 | 41.1 | 94.6 | 83.0 | 79.4 | 68.3 | 82.0 | |

64[4], except for MEMO (Zhang et al., 2021), which predicts on single images to incentivize prediction consistency over an input and its augmentations. Regarding datasets, we follow earlier works (Wang et al., 2020; Niu14 et al., 2023; Niu et al., 2022; Gao et al., 2022; Zhang et al., 2021), and thus evaluate on the ImageNet-C dataset (Hendrycks & Dietterich, 2019) with a corruption level of 5 for all 15 corruptions. We further extend our evaluation and consider CIFAR10-C, ImageNet-R (Hendrycks et al., 2021), and the more recent ImageNet-3DCC (Kar et al., 2022), which leverages depth estimates to construct more spatially-consistent corruptions.

Our experiments compare the performance of the baseline model $f_\theta$ (without test time adaptation) against 15 state-of-the-art TTA methods published in top-tier venues (*e.g.*, CVPR, NeurIPS, and ICLR) between 2017 and 2023. In particular, we consider: BN (Schneider et al., 2020) and AdaBN (Li et al., 2016), which adjust the statistics of the batch normalization layers; SHOT (Liang et al., 2020) and SHOT-IM (Liang et al., 2020), which fine-tune the feature extractor to maximize mutual information; entropy minimization approaches such as TENT (Wang et al., 2020), ETA (Niu et al., 2022) (a more efficient version of TENT), and SAR (Niu14 et al., 2023), which trains the learnable parameters of the batch normalization layers; distillation approaches, such as CoTTA (Wang et al., 2022), Pseudo Labeling (PL) (Lee et al., 2013), and the very recent and efficient LAME (Boudiaf et al., 2022); EATA (Niu et al., 2022) and TTAC (Su et al., 2022) that assume access to the source training data; data-dependent approaches such as MEMO (Zhang et al., 2021) and the diffusion-based method DDA (Gao et al., 2022). For all methods, we use their official implementation with their recommended hyperparameters. We report our experimental results on a subset of 12 baselines, while leaving ETA, SHOT-IM, and PL to the appendix due to space constraints and their similarity to SHOT and EATA.

As mentioned in Section 3.2 , our protocol performs an online computation of the relative adaptation speed of $g$. In particular, for each batch revealed by the stream, we compute $\mathcal{C}\left(g(x)\right)$. Then, if $\mathcal{C}(g(x_i)) = k$, all the samples $\{x_{i+1}, x_{i+2}, \ldots, x_{i+k}\}$ are processed by $f_{\theta_i}$ without adaptation. Otherwise, if $\mathcal{C}(g(x_i)) = 1$, then these samples are processed by $g$. For methods that accumulate parameter updates such as TENT (Wang et al., 2020), $f_{\theta_i}$ is the most recent updated model $g(f_{\theta_{i-1}})$. We report all our main results as the average across three seeds, and leave the detailed

[4]This batch size is recommended by most baselines (Wang et al., 2020; Niu et al., 2022)




(a) **Current** Continual TTA.

(b) **Realistic** Continual TTA.

Figure 3: **Continual Error Rate on ImageNet-C.** We report the continual error rate of several TTA methods on ImageNet-C benchmark under both realistic and current setups. A lower error rate indicates a better TTA method. Continual evaluation means the corruptions are presented in a sequence without resetting the model in between. We choose the same order as presented along the x-axis; starting with brightness and ending with clean validation set. In the current setup, we observe an increasing trend for SHOT, TENT, and TTAC-NQ. This is hypothesized to be due to overfitting on the early distribution shifts. This behavior is mitigated in the realistic setup due to adapting to fewer batches. EATA and SAR perform equally well in both realistic and current continual setups due to sample rejection. We report the standard deviation across 3 seeds.

analysis to the Appendix. Throughout the experiments, we refer to our realistic evaluation protocol as "realistic/online", and refer to the current protocol as "current/offline". Next, we evaluate all methods on four different scenarios: *(i)* when domain shifts happen in an episodic manner, *(ii)* when domain shifts happen continually, *i.e.* one after the other, *(iii)* when the stream speed varies, *(iii)* when domain shifts happen continually with label correlation; practical evaluation (Yuan et al., 2023) ,and *(v)* when the baseline $f_\theta$ is unavailable for evaluating the samples skipped by the TTA method $g$ (left for the appendix).

### 4.1. Episodic Evaluation of TTA

First, we consider an episodic evaluation of domain shifts, whereby $\mathcal{S}$ contains a single domain (*e.g.* one corruption) from ImageNet-C. We analyze this simple and most common setup to assess the performance of TTA methods under real-time evaluation. We report the error rates on all corruptions in Table 2 and the average error rate across corruptions. We summarize the insights as follows:

**(i) The performance of TTA methods often degrades significantly under the realistic setup.** Most methods induce a significant computational overhead, which prevents them from adapting to every sample from the test stream. For example, the error rate increases by 7.5% for TTAC-NQ and 4.3% for TENT, where $\mathcal{C}(g_{\text{TTAC-NQ}}) = 12$ and $\mathcal{C}(g_{\text{TENT}}) = 3$ (see Table 1). That is, TENT adapts to one-third of the batches revealed by the stream, while TTAC-NQ adapts to one every twelve batches.

**(ii) Very efficient methods, with $\mathcal{C}(g) = 1$, such as LAME and BN, do not lose in performance.** Evaluating such methods in offline or realistic setups is inconsequential, as their adaptation incurs negligible additional computation (since they adapt during the forward pass (Li et al., 2016; Schneider et al., 2020) or by adjusting the logits (Boudiaf et al., 2022) at a speed that pales in comparison to that of the stream). Interestingly, in our realistic evaluation, the simple BN (published in 2020) with an average error rate of 66.7% outperforms more recent and advanced methods such as SAR (published in 2023) by 1.7%. Furthermore, AdaBN (published in 2017) significantly outperforms the very recent diffusion-based DDA by a notable 13%.

**(iii) Data-dependent approaches, such as MEMO and DDA, are extremely inefficient.** Despite the independence of MEMO and DDA on batch size, they incur a massive computational burden. For instance, $\mathcal{C}(g_{\text{MEMO}}) = 54$ and $\mathcal{C}(g_{\text{DDA}}) = 810$. Thus, both methods will be busy adapting for considerable portions of the stream, leaving most predictions to the non-adapted classifier. This phenomenon is the reason behind the reported performance of these methods being so close to that of $f_\theta$ (*i.e.* around 82%). This result calls for future research to focus on increasing the efficiency of data-dependent adaptation methods.

**(iv) Sample rejection-oriented methods can perform well under the realistic protocol.** EATA adapts efficiently due to its fast sample rejection algorithm, which relies solely on

the forward pass to admit samples for adaptation. EATA's low error rate of 55.6%, combined with a small performance drop of less than 4%, positions it as the *top performer* under the realistic evaluation protocol on ImageNet-C. On the other hand, SAR does not benefit from sample rejection. SAR's performance drop of 7.5% is due to its dependence on gradients for sample rejection, which reduces its speed.

**(v) SHOT benefits from the realistic protocol.** Interestingly, we found that SHOT (and SHOT-IM in the Appendix), a fine-tuning-based approach, benefits from our realistic evaluation. In particular, we found that SHOT's error rate decreases by 2% on fog corruption and by 0.8% on average. This observation could suggest that SHOT could potentially improve performance by disposing of fine-tuning on every batch. It is also worth mentioning that, under our realistic evaluation, SHOT (introduced in 2020) outperforms *all* methods except EATA.

**(vi) Performance changes are consistent across corruptions.** Note that all methods that are somewhat efficient can improve the source model across all corruptions, in both the offline and realistic setups. Furthermore, the performance changes when comparing the offline and realistic setups are consistent across all corruptions. This finding suggests that the performance of these methods is independent of the domain shift being considered. We further test this hypothesis by benchmarking these methods on two other datasets with other types of domain shifts in Section 4.4.

### 4.2. Continual Evaluation of TTA

Next, we analyze the more challenging continual setup, following (Wang et al., 2022; Niu et al., 2022). In particular, we construct the stream $\mathcal{S}$ by concatenating *all* corruptions from ImageNet-C. That is, we adapt TTA methods continually on all corruptions followed by the clean validation set, without ever resetting the network weights. We introduce the notion of realistic adaptation to the continual setup to study the effects of a constant stream speed on the benchmark. We report results in Figure 3 for both the offline and realistic protocols, where the horizontal-axis shows how corruptions are ordered in the stream. We limit the experiments in this section to six TTA methods (SHOT, TENT, TTAC-NQ, COTTA, EATA, and SAR), and leave the remaining details for the Appendix. We observe:

**(i) Methods that do not perform sample rejection (SHOT, TENT, TTAC) scale poorly in the offline-continual setup.** This phenomenon can be attributed to these methods overfitting to early distributions. However, methods that do perform sample rejection (SAR and EATA) do not overfit as easily to corruptions, and can thus adapt to the rest of the stream. Even worse, such methods tend to even significantly degrade the performance on clean data.

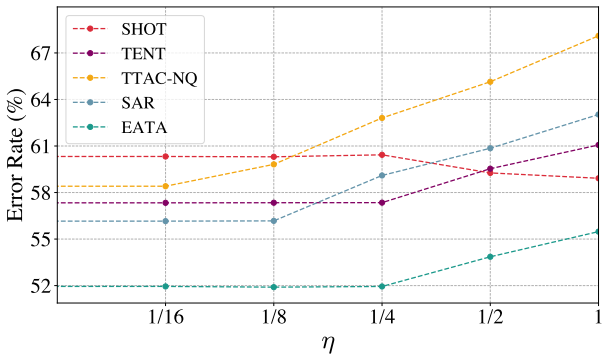



Figure 4: **Average Error Rate on ImageNet-C Under Slower Stream Speeds.** We report the average error rate for several TTA methods on ImageNet-C under slower stream speeds. In our proposed realistic model evaluation, the stream speed $r$ is normalized by the time needed for a forward pass using the base model. We evaluate different TTA methods under a stream with speed $\eta r$ with $\eta \in (0, 1]$. An $\eta = 1/16$ means the stream is 16 times slower than the forward pass of the base model. We report the standard deviation across 3 different random seeds. Different TTA methods degrade differently when varying $\eta$.

**(ii) In the realistic-continual setup, methods that do not perform sample rejection benefit from skipping adaptation on some batches, and become competitive with the methods that perform sample rejection.** That is, while skipping parts of the stream deteriorated the performance of such methods in the episodic evaluation , this skipping actually helped in preventing these methods from over-fitting in the continual setup.

### 4.3. Stream Speed Analysis

In the previous experiments, we normalized the stream speed to be the same as that of $f_\theta$'s forward pass. That is, we assumed that the rate $r$ at which $\mathcal{S}$ reveals new batches is equal to $R\left(f_\theta(x)\right)$. However, some applications may enjoy a slower stream, giving TTA methods more time to adapt to samples. To explore this scenario, we vary the speed at which the stream reveals new data. In particular, let the new stream rate be $\eta\, r$ with $\eta \in (0, 1]$. Hence, as $\eta \to 0$, the stream slows down and allows methods to adapt to all samples. Conversely, as $\eta \to 1$, the stream speeds up, and at $\eta = 1$ we recover our realistic evaluation protocol.

We experiment with the stream speed by setting $\eta \in \{1/16, 1/8, 1/4, 1/2, 1\}$, and evaluate five representative TTA methods (SHOT, TENT, TTAC-NQ, SAR, and EATA) in the episodic setup . Figure 4 summarizes our results by reporting the average error rate across all corruptions. We next list our observations:

**(i) The performance of TTA methods varies widely.** For

Table 3: **Episodic Error Rate on ImageNet-C with ViT.** We report the error rate of three baselines (Source, Tent, SAR) on the 15 different corruptions on ImageNet-C when the backbone is ViT architecture pretrained on ImageNet. We observe that while generally better backbones yield smaller error rate, expensive methods perform worse under our realistic evaluation. The more expensive the method is (e.g. SAR compared to Tent), the more performance reduction it suffers.

| Method | Realistic | Noise | | | Blur | | | | Weather | | | | Digital | | | | Avg. | Δ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | gauss. | shot | impul. | defoc. | glass | motion | zoom | snow | frost | fog | brigh. | contr. | elast. | pixel. | jpeg | | |
| Source | ✓ | 90.5 | 93.3 | 91.8 | 71.0 | 76.6 | 66.1 | 72.9 | 84.1 | 73.5 | 52.8 | 45.3 | 55.9 | 69.5 | 55.5 | 52.2 | 70.1 | - |
| Tent | ✗ | 69.9 | 95.9 | 68.9 | 55.8 | 62.0 | 52.3 | 57.9 | 57.2 | 53.6 | 41.8 | 28.9 | 40.7 | 59.1 | 39.7 | 42.0 | 55.0 | (-6.1) |
| | ✓ | 80.7 | 88.9 | 81.0 | 63.0 | 69.5 | 58.3 | 64.9 | 65.8 | 59.7 | 47.7 | 33.2 | 47.3 | 64.6 | 45.1 | 46.4 | 61.1 | |
| SAR | ✗ | 55.5 | 56.9 | 55.1 | 47.5 | 50.4 | 44.3 | 48.7 | 42.4 | 47.3 | 33.6 | 25.4 | 35.6 | 44.8 | 33.5 | 36.4 | 43.8 | (-10.7) |
| | ✓ | 70.0 | 72.5 | 69.4 | 56.6 | 63.4 | 54.0 | 60.0 | 56.4 | 53.5 | 43.0 | 30.5 | 43.3 | 58.7 | 41.5 | 43.8 | 54.5 | |

example, TTAC-NQ starts degrading faster (at $\eta = 1/16$) due to its slow adaptation speed. For other methods, the $\eta$ at which they degrade varies. For instance, while TENT has a higher error rate than SAR in slow streams ($\eta \leq 1/8$), TENT outperforms SAR in the regime of faster streams $\eta \leq 1/4$. Interestingly, SHOT (Liang et al., 2020) ranks the worst at $\eta \leq 1/8$, then ranks second when $\eta \geq 1/2$, becoming a viable alternative. At last, the order of different methods significantly changes depending on the speed of the stream. For example, SAR changes from being second best at $\eta \leq 1/8$ to third at $\eta = 1/4$ and then to fifth (*i.e.* second worst) at $\eta \geq 1/2$.

**(ii) EATA provides a good trade-off between speed and performance.** In fact, EATA gives the best overall performance (lowest error rate) independent of the stream's speed. This virtue is attributable to EATA's combination of good performance and adaptation speed based on efficient sample rejection. Results on other datasets are in the Appendix.

### 4.4. Results on Other Benchmarks and Architectures

We extend our evaluation protocol to cover ImageNet-3DCC (Kar et al., 2022) and ImageNet-R (Hendrycks et al., 2021) datasets and ResNet-18 (results in the appendix) and ViT (Kolesnikov et al., 2021) architectures. ImageNet-R contains rendition versions of ImageNet spanning 200 classes. ImageNet-3DCC constructs more spatially-consistent corruptions than ImageNet-C by leveraging depth estimates. For ViT, we conduct episodic evaluation on ImageNet-C in a similar setup to Section 4.1 and report the results in Table 3 for the non-adapted model, Tent, and SAR. For ImageNet-R and ImageNet-3DCC, we fix the architecture to ResNet-50 and experiment on the entire datasets and set the severity level to 5 in ImageNet-3DCC. Due to the space constraint, we limit our experiments to the episodic evaluation, and leave other results and analyses to the Appendix. We evaluate the effectiveness of 10 TTA methods in Table 4, where we report the average error rate across all corruptions.

We observe that **our results are consistent across all considered datasets and architectures.** Similar to our results in Table 2, the more computationally involved SAR degrades more than Tent when leveraging ViT architecture. Regarding other datasets, we find that simple methods that adapt during the forward pass are unaffected by the realistic setup. All the other methods, except SHOT, experience degradation in their results on both datasets. We observe again that, on these two datasets, while SHOT actually benefits from the realistic evaluation, EATA remains the best alternative on both ImageNet-R and ImageNet-3DCC.

Table 4: **Average Error Rate on ImageNet-R and ImageNet-3DCC.** We report the average error rate of different TTA methods on ImageNet-R and ImageNet-3DCC under both the realistic and current setups. A lower error rate indicates a better TTA method. The highlighted numbers indicate a better performance per method across setups. We observe that methods generally perform worse in the more realistic realistic setup. The conclusions are consistent with what we observed on ImageNet-C (Table 2).

| Method | ImageNet-R | | | ImageNet-3DCC | | |
|---|---|---|---|---|---|---|
| | Current | Realistic | Δ | Current | Realistic | Δ |
| Source | 63.8 | 63.8 | - | 73.9 | 73.9 | - |
| AdaBN | 60.6 | 60.6 | 0 | 72.1 | 72.1 | 0 |
| BN | 60.0 | 60.0 | 0 | 70.5 | 70.5 | 0 |
| LAME | 60.5 | 60.5 | 0 | 72.1 | 72.1 | 0 |
| SHOT | 70.3 | 62.6 | (+7.7) | 69.2 | 67.0 | (+2.2) |
| TENT | 58.1 | 59.1 | (-1.0) | 64.5 | 66.8 | (-2.3) |
| SAR | 57.5 | 59.6 | (-2.1) | 63.5 | 71.4 | (-7.9) |
| CoTTA | 57.3 | 61.5 | (-4.5) | 66.4 | 75.6 | (-9.2) |
| EATA | 55.7 | 57.1 | (-1.4) | 60.9 | 63.1 | (-2.2) |
| TTAC-NQ | 59.2 | 60.8 | (-1.6) | 65.7 | 73.6 | (-7.9) |

### 4.5. Evaluation under Practical TTA

Recently, (Yuan et al., 2023) extended the continual test-time adaptation evaluation to include label-imbalances; known as Practical Test-Time Adaptation (PTTA) setup. In this setting, the stream not only reveals a continual sequence of distribution shifts, but also the revealed batches

Table 5: **Episodic Error Rate on CIFAR10-C under Practical Evaluation (Yuan et al., 2023).** We report the error rate of two baselines (Source, RoTTA (Yuan et al., 2023)) on the 15 different corruptions on CIFAR10-C when the backbone is ResNet-18. We observe that under our computational constrained evaluation, the only method tailored to this setting; RoTTA, performs worse than the non-adapted baseline.

| Method | Realistic | Noise | | | Blur | | | | Weather | | | | Digital | | | | Avg. | Δ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | gauss. | shot | impul. | defoc. | glass | motion | zoom | snow | frost | fog | brigh. | contr. | elast. | pixel. | jpeg | | |
| Source | ✓ | 72.3 | 65.7 | 72.9 | 46.9 | 54.3 | 34.8 | 42.0 | 25.1 | 41.3 | 26.0 | 9.3 | 46.7 | 26.6 | 58.5 | 30.3 | 43.5 | - |
| RoTTA | ✗ | 36.9 | 34.9 | 45.8 | 16.6 | 44.2 | 19.9 | 16.53 | 21.6 | 22.4 | 18.8 | 9.8 | 20.6 | 28.4 | 27.1 | 34.5 | 26.5 | (-22.5) |
| | ✓ | 55.0 | 54.4 | 63.2 | 43.3 | 62.3 | 43.7 | 43.5 | 44.8 | 47.7 | 43.4 | 35.3 | 41.8 | 54.0 | 47.7 | 54.6 | 49.0 | |

have significant label imbalances. To combat this combined challenge, the work of (Yuan et al., 2023) proposed to leverage a balanced memory bank for adaptation. In this section, we extend our computational constrained evaluation to the PTTA setup and compare RoTTA (Yuan et al., 2023) with a non-adapted model on CIFAR10-C benchmark.

Table 5 summarizes the results. We observe that while RoTTA indeed reduces the error rate under the PTTA setup on CIFAR10-C (17% below the non-adapted model), our realistic evaluation uncovers its computational limitation. We found that RoTTA's error rate increases by over 22% surpassing the error rate of the non-adapted model. Note that RoTTA stores samples from the stream in a memory bank then adapts the model on sampled samples from the memory bank. Thus, the slower the adaptation of RoTTA, the less diverse the samples in the memory bank, hindering its adaptation.

### 4.6. Effect of Hyper-parameter Tuning

The performance of different TTA methods heavily depends on their hyper-parameter settings (Zhao et al., 2023). Here, we assess the impact of our proposed evaluation on TTA methods when tuning their hyperparameters. For that regard, we conduct hyper parameter search for Tent (as a fundamental baseline) and experiment with different learning rates (the only hyper-parameter for Tent).

Table 6 summarizes the results under episodic evaluation for 4 different corruptions on ImageNet-C. We observe that while conducting hyper-parameter search indeed improves the performance of TENT, its error rate increases under our realistic evaluation across all hyperparameters. That is, while conducting hyper-parameter search might indeed result in a better performance for TTA methods, the insights obtained through our proposed evaluation scheme remains consistent: more efficient TTA methods will have a smaller performance drop under the realistic evaluation.

## 5. Conclusions

In this work, we find that the performance of Test Time Adaptation (TTA) methods can vary depending on the context in which they are used. In the episodic evaluation, the efficiency of the method is the most important factor, with more efficient methods like AdaBN and BN showing consistent performance, while data-dependent approaches suffer. Sample rejection methods generally perform well, and fine-tuning approaches such as SHOT can even improve when adapting to fewer samples. In the continual evaluation, methods that do not perform sample rejection scale poorly in the offline-continual setup but benefit from skipping adaptation on some batches in the realistic-continual setup. Furthermore, our stream speed analysis shows that the performance of TTA methods can vary widely at different speeds. Our findings are consistent across corruptions and multiple datasets. They can help researchers and practitioners to better understand the strengths and weaknesses of different TTA methods, and to choose the most appropriate method for their specific use case.

Table 6: **Effect of our evaluation under hyperparameter tuning.** We report the error rate for Tent under different learning rates under both the current and our proposed realistic evaluation. While carefully tuning the learning rate for Tent results in a better performance, our realistic evaluation causes a performance drop under all learning rates.

| $lr$ | Realistic | gauss. | motion | fog | pixel. | Avg. | Δ |
|---|---|---|---|---|---|---|---|
| $1 \times 10^{-4}$ | ✗ | 74.1 | 63.3 | 44.7 | 43.5 | 56.4 | (-4.4) |
| | ✓ | 79.7 | 69.0 | 47.8 | 46.8 | 60.8 | |
| $2 \times 10^{-4}$ | ✗ | 71.1 | 59.7 | 43.1 | 41.9 | 53.9 | (-4.7) |
| | ✓ | 77.6 | 66.1 | 46.0 | 45.0 | 58.7 | |
| $3 \times 10^{-4}$ | ✗ | 69.6 | 58.1 | 42.4 | 41.1 | 52.8 | (-4.2) |
| | ✓ | 74.9 | 64.0 | 45.0 | 44.0 | 57.0 | |
| $4 \times 10^{-4}$ | ✗ | 68.8 | 57.1 | 42.0 | 40.8 | 52.2 | (-3.7) |
| | ✓ | 73.7 | 62.3 | 44.5 | 43.2 | 55.9 | |

## Acknowledgements

This work was partially done during a research internship of the first author at Intel Labs. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2021-4648. We would like to thank Yasir Ghunaim and Mattia Soldan for the helpful discussion.

## Impact Statement

Our work advances Machine Learning by proposing a realistic evaluation protocol for Test Time Adaptation methods, prioritizing computational efficiency. This approach promotes the development of AI systems that are both accessible in resource-limited settings and environmentally sustainable, by favoring simpler, faster methods. Such advancements contribute to more inclusive and responsible AI deployment, aligning with ethical goals of broadening access and reducing environmental impacts

## References


Alfarra, M., Pérez, J. C., Thabet, A., Bibi, A., Torr, P. H., and Ghanem, B. Combating adversaries with anti-adversaries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 5992–6000, 2022.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020.

Boudiaf, M., Mueller, R., Ben Ayed, I., and Bertinetto, L. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353, 2022.

Cai, Z., Sener, O., and Koltun, V. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8281–8290, 2021.

Chen, D., Wang, D., Darrell, T., and Ebrahimi, S. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022.

Croce, F., Gowal, S., Brunner, T., Shelhamer, E., Hein, M., and Cemgil, T. Evaluating the adversarial robustness of adaptive test-time defenses. In *International Conference on Machine Learning*, pp. 4421–4435. PMLR, 2022.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Gao, J., Zhang, J., Liu, X., Darrell, T., Shelhamer, E., and Wang, D. Back to the source: Diffusion-driven test-time adaptation. *arXiv preprint arXiv:2207.03442*, 2022.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., and Lee, S.-J. Note: Robust continual test-time adaptation against temporal correlation. In *Advances in Neural Information Processing Systems*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Iwasawa, Y. and Matsuo, Y. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34: 2427–2440, 2021.

Kar, O. F., Yeo, T., Atanov, A., and Zamir, A. 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18963–18974, 2022.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Kojima, T., Matsuo, Y., and Iwasawa, Y. Robustifying vision transformer without retraining from scratch by test-time class-conditional feature alignment. *arXiv preprint arXiv:2206.13951*, 2022.

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Uszkoreit, J., Beyer, L., Minderer, M., Dehghani, M., Houlsby, N., Gelly, S., Unterthiner, T., and Zhai, X. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

Lee, D.-H. et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896, 2013.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

Liang, J., Hu, D., and Feng, J. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pp. 6028–6039. PMLR, 2020.

Liu, Y., Kothari, P., Van Delft, B., Bellot-Gurlet, B., Mordan, T., and Alahi, A. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021.

Mancini, M., Karaoguz, H., Ricci, E., Jensfelt, P., and Caputo, B. Kitting in the wild through online domain adaptation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1103–1109. IEEE, 2018.

Mirza, M. J., Micorek, J., Possegger, H., and Bischof, H. The norm must go on: dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14765–14775, 2022a.

Mirza, M. J., Soneira, P. J., Lin, W., Kozinski, M., Possegger, H., and Bischof, H. Actmad: Activation matching to align distributions for test-time-training, 2022b. URL https://arxiv.org/abs/2211.12870.

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Niu14, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., and Tan15, M. Towards stable test-time adaptation in dynamic wild world. *International Conference on Learning Representations*, 2023.

Pérez, J. C., Alfarra, M., Jeanneret, G., Rueda, L., Thabet, A., Ghanem, B., and Arbeláez, P. Enhancing adversarial robustness via test-time transformation ensembling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 81–91, 2021.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. Adapting visual category models to new domains. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pp. 213–226. Springer, 2010.

Sakaridis, C., Dai, D., and Van Gool, L. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10765–10775, 2021.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 2020.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Su, Y., Xu, X., and Jia, K. Revisiting realistic test-time training: Sequential inference and adaptation by anchored clustering. *arXiv preprint arXiv:2206.02721*, 2022.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *CVPR 2011*, pp. 1521–1528. IEEE, 2011.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176, 2017.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Wang, Q., Fink, O., Van Gool, L., and Dai, D. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022.

Yuan, L., Xie, B., and Li, S. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15922–15932, 2023.

Zhang, M., Levine, S., and Finn, C. Memo: Test time robustness via adaptation and augmentation. *arXiv preprint arXiv:2110.09506*, 2021.

Zhao, H., Liu, Y., Alahi, A., and Lin, T. On pitfalls of test-time adaptation. *International Conference on MAchine Learning*, 2023.

# A. Methodology

## A.1. Online Computation of $\mathcal{C}(g)$

Section 3.2 discussed the online evaluation protocol of TTA methods. Here, we give more details on the calculation of $\mathcal{C}(g)$, the relative adaptation speed of $g$, during our online evaluation. First, we set $R(g(x))$ as the time recording function for $g$ to perform a forward pass for a single batch. To ensure a reliable time calculation, we execute `torch.cuda.synchronize()` before starting the timer and before ending it. This ensures all GPU operations are finished for the moment time is computed. To alleviate hardware dependence, we also calculate $R(f_\theta(x))$ for *each* evaluation step computing the relative adaptation complexity. It is worth mentioning that $\mathcal{C}(g)$ for SHOT, EATA, SAR, and COTTA are $[3, 3, 8, 103]$ on average, respectively.

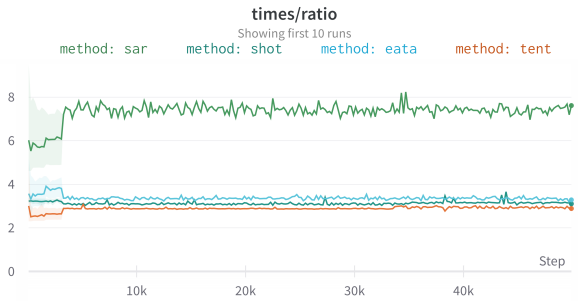



Figure 5: $\mathcal{C}(g)$ **computation across iterations.** We report our online calculations for the relative adaptation speed of $g$, $\mathcal{C}(g)$, for SAR, SHOT, EATA, and TENT throughout a full evaluation episode. We observe that, overall, $\mathcal{C}(g)$ has a stable behavior throughout evaluation iterations.

# B. Experiments

## B.1. Episodic Evaluation of TTA

**SHOT, PL, and ETA** For completeness, we report the results on 3 baselines: Pseudo Label (Lee et al., 2013), SHOT-IM (Liang et al., 2020), and ETA (Niu et al., 2022) in Table 7. We follow the same setup as in the main paper. Our results are consistent with the findings of Section 4.1 and Table 2. In particular, SHOT-IM improves its performance under the online evaluation, similar to SHOT. Further, the performance of ETA and PL degrades under the online evaluation due to the additional computational burden. Nevertheless, ETA is similar to EATA in providing the best tradeoff between additional computational requirements and performance improvements.

**SAR with GN** We equip our results to include ResNet50 with Group Normalization (GN) layers, following (Niu14 et al., 2023). We report the results in Table 7, where we observe that: **(i)** Under a relatively large batch size (64), ResNet50 with GN underperforms ResNet50 with Batch Normalization. In fact, the average error rate for SAR increases from 56.2% to 65.8%. **(ii)** The online evaluation penalizes SAR in both architecture choices with a performance degradation of 3.6% under the GN-based ResNet. Finally, it is worth mentioning that SAR with GN layers attains a similar performance under a batch size of 1.

**Ablating Batch Sizes** In the experiments section, we fixed the batch size to 64 following the recommendations of earlier works (Wang et al., 2020; Niu et al., 2022). Here, we investigate the effect of our proposed online evaluation under different choices of batch sizes. To that end, we vary the batch size in $\{1, 16, 32, 128\}$, and report the results in Figure 6. We draw the following observations:

Table 7: **Episodic Error Rate on ImageNet-C.** We report the error rate of different TTA methods on the ImageNet-C benchmark under both the online and offline setups. A lower error rate indicates a better TTA method. The highlighted numbers indicate a better performance per method across setups. Episodic means the model will adapt to one corruption at a time. The model is reset back to the base model when moving to the next corruption. The offline setup is merely the reproduction of every method. We show that methods generally perform worse in the more realistic online setup. The more computationally complex the TTA method is, the less data it will adapt to, and the worse its performance. SAR-GN represents SAR when deployed on ResNet50 with Group Normalization (GN) layers, following (Niu14 et al., 2023).

| Method | Online | Noise | | | Blur | | | | Weather | | | | Digital | | | | Avg. | Δ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | gauss. | shot | impul. | defoc. | glass | motion | zoom | snow | frost | fog | brigh. | contr. | elast. | pixel. | jpeg | | |
| SHOT-IM | ✗ | 73.1 | 69.8 | 72.0 | 76.9 | 75.9 | 58.5 | 52.7 | 53.3 | 62.2 | 43.8 | 34.6 | 82.6 | 46.0 | 42.3 | 48.9 | 59.5 | (-0.3) |
| | ✓ | 71.1 | 68.6 | 70.7 | 73.2 | 73.6 | 59.1 | 51.9 | 52.8 | 60.5 | 43.7 | 33.6 | 77.3 | 45.7 | 42.1 | 48.6 | 58.2 | |
| PL | ✗ | 92.2 | 92.2 | 92.8 | 97.0 | 89.8 | 57.7 | 49.6 | 50.7 | 57.1 | 41.5 | 32.6 | 91.1 | 44.3 | 40.3 | 46.6 | 65.0 | (+0.3) |
| | ✓ | 90.6 | 86.3 | 83.6 | 93.2 | 89.7 | 63.0 | 51.7 | 55.0 | 59.3 | 43.8 | 32.9 | 92.3 | 47.3 | 42.4 | 49.3 | 65.3 | |
| ETA | ✗ | 64.9 | 62.7 | 63.6 | 66.4 | 66.3 | 52.4 | 47.3 | 48.2 | 54.1 | 40.2 | 32.2 | 54.8 | 42.3 | 39.2 | 44.7 | 52.0 | (+3.6) |
| | ✓ | 70.2 | 67.0 | 69.6 | 71.5 | 71.5 | 56.9 | 50.2 | 51.9 | 57.0 | 42.0 | 32.5 | 60.5 | 44.6 | 40.8 | 47.1 | 55.6 | |
| SAR-GN | ✗ | 71.8 | 69.0 | 70.3 | 81.5 | 81.0 | 69.6 | 69.5 | 57.1 | 56.6 | 94.3 | 29.2 | 56.0 | 84.8 | 51.4 | 44.7 | 65.8 | (+3.6) |
| | ✓ | 82.0 | 80.2 | 82.1 | 80.2 | 88.6 | 78.5 | 75.1 | 59.6 | 53.9 | 66.9 | 30.7 | 63.3 | 81.3 | 71.3 | 47.5 | 69.4 | |

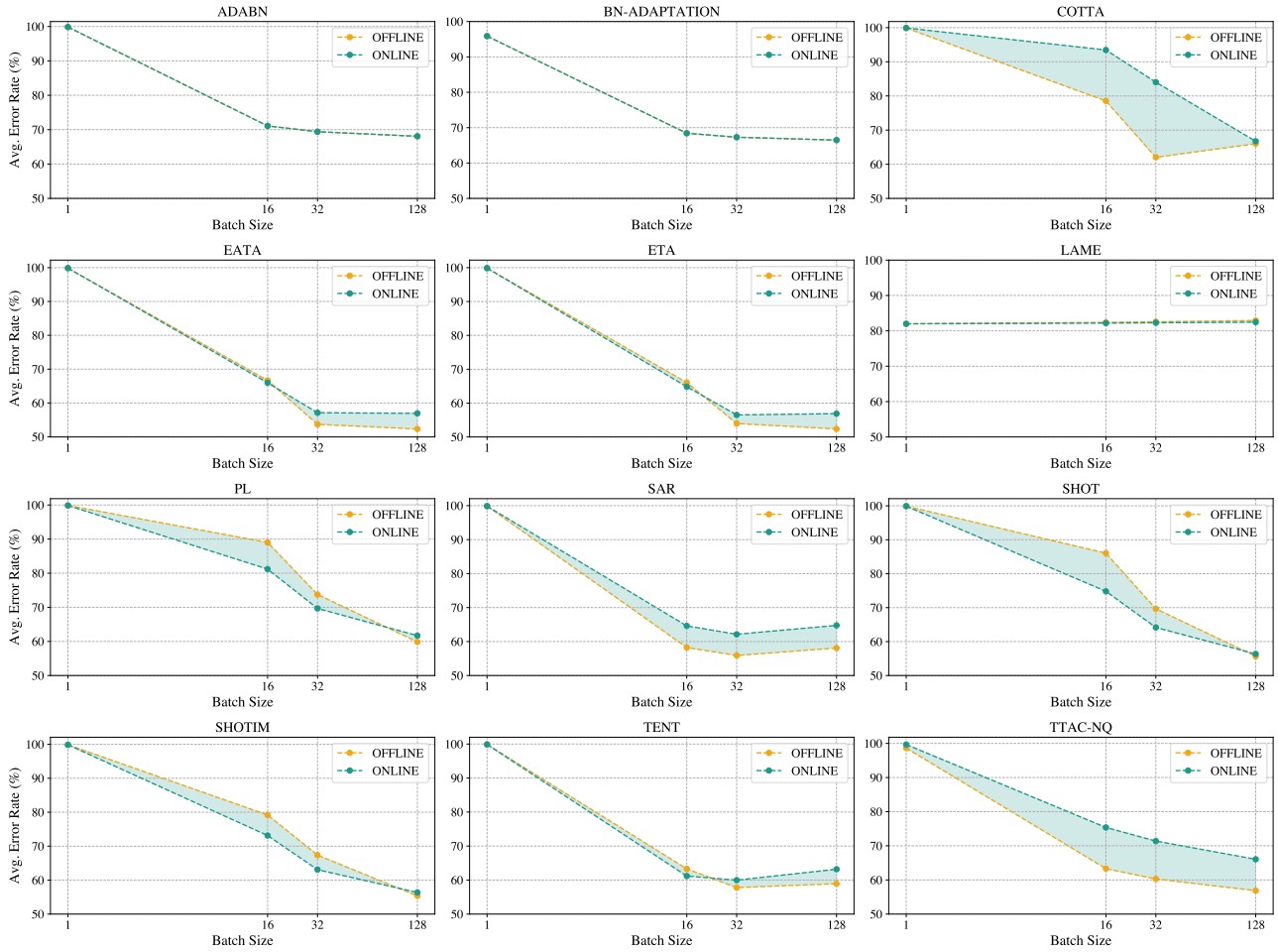



Figure 6: **Batch Size Analysis current *vs.* realistic setups for every method.** We assess the performance variation of 12 different TTA methods under varying batch sizes. We experiment with batch sizes in $\{1, 16, 32, 128\}$. We do not include the baseline, MEMO, and DDA, since they are data-dependent approaches and are unaffected by batch size. All TTA methods, except LAME, are severely affected by smaller batch sizes. Nonetheless, the realistic evaluation degrades the performance of all methods, except SHOT and SHOT-IM.

**(i) Online evaluation improves the performance of SHOT and SHOT-IM**. This result is consistent with the earlier observations in Table 2. Note that PL shares a similar trend as well.

**(ii) The performance of TTA methods degrades when switching from offline to online evaluation, regardless of the batch size.** This result is highlighted in COTTA, ETA, EATA, SAR, TENT, and TTAC-NQ.

**(iii) Performance of TTA methods vastly varies when varying the batch size.** This result is consistent with earlier findings in the literature (Gao et al., 2022; Niu14 et al., 2023), where most TTA methods fail with small batch sizes.

At last, and to ease comparison across methods, we summarize all the plots for all methods in Figure 7.

**Consistency with 3 random seeds.** For all of our experiments, we run each experiment with 3 random seeds. In most of our results, we found out that the standard deviation of performance across runs is very small. Our results in Figures 3 and 4 demonstrate this variation in the shaded area for 5 different TTA methods.

### B.2. Continual Evaluation of TTA

We further explore another setup for the continual evaluation of TTA. In particular, we follow (Wang et al., 2022) in concatenating all corruptions in ImageNet-C with 11 different orders. We then report the average performance of each method across all runs and corruptions in Table 8. We run each experiment with 3 random seeds, and report our results with standard deviations. For the remaining implementation

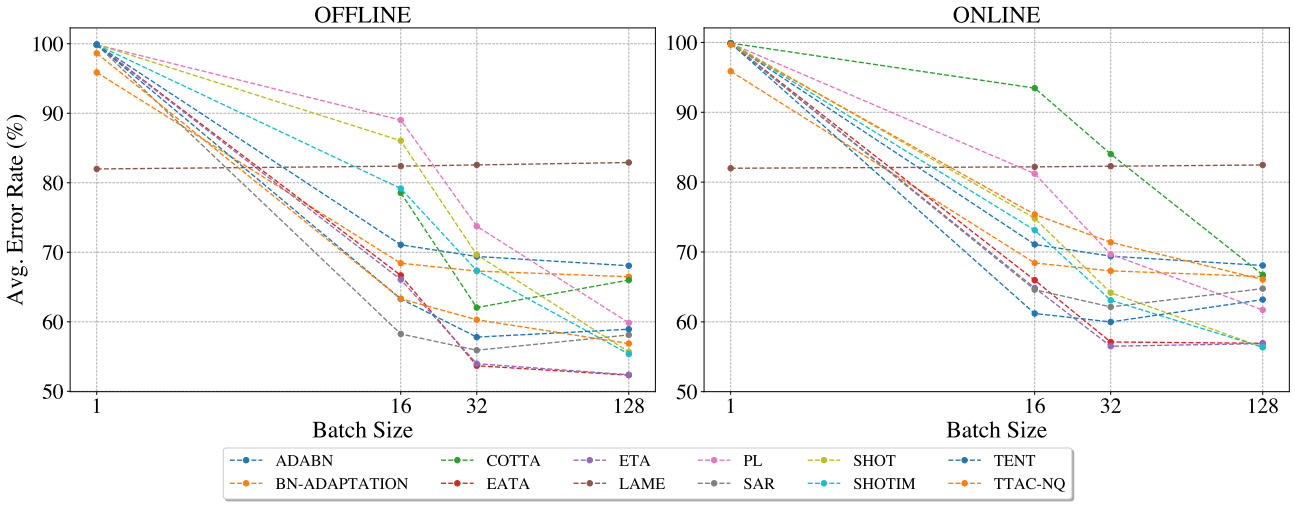



Figure 7: **Summary of batch size analysis: current *vs.* realistic setups.** Left: Current evaluation, i.e.,Section 3.1. Right: Realistic evaluation,i.e.,Section 3.2. While EATA achieves the lowest error rate under batch sizes $\geq 32$, SHOT becomes a very competitive baseline, outperforming EATA, at a batch size of 128.

Table 8: **Continual Error Rate on ImageNet-C.** We report the average continual error rate for 11 different corruption orders, with 3 different seeds, under both the offline and online setups with a corruption severity level of 5. *Continual* refers to continually adapting after each corruption without resetting. This metric indicates the model's capability to learn from previous corruptions. The offline setup refers to the performance of the model in a continual learning scheme, whereas the online setup refers to the performance of the model in a continual learning scheme, under our more realistic online setup. We show that the more complex a method is, the fewer samples it adapts to, achieving better performance in a continual learning scheme.

| Avg. Error (%) | COTTA | ETA | TENT | SAR | EATA | SHOT | TTAC-NQ |
|---|---|---|---|---|---|---|---|
| Offline | $65.3 \pm 5.9$ | $56.4 \pm 2.3$ | $84.6 \pm 16.0$ | $59.8 \pm 3.0$ | $56.4 \pm 2.3$ | $88.4 \pm 11.4$ | $81.8 \pm 11.4$ |
| Online | $69.3 \pm 2.8$ | $57.7 \pm 2.0$ | $65.6 \pm 5.0$ | $60.4 \pm 1.8$ | $57.7 \pm 1.9$ | $78.2 \pm 7.7$ | $65.1 \pm 3.8$ |

details, we follow our setup in main paper. We observe that, similar to our conclusions in Section 4.2, online evaluation helps methods that do not perform sample rejection (e.g.,TENT). Nonetheless, both ETA and EATA provide the best trade-off between performance and additional computational burden.

### B.3. Stream Speed Analysis

For completeness, we extend our stream speed analysis in Section 4.3 to cover the ImageNet-3DCC dataset. We preserve our experimental setup by varying the stream speed according to $\eta r$, with $\eta \in \{1/16, 1/8, 1/4, 1/2, 1\}$. Figure 8 summarizes our results for SHOT, TENT, TTAC-NQ, EATA, and SAR. We observe similar trends to the ones in Figure 4, where the performance of different TTA methods varies widely under different stream speeds. The large relative adaptation speed of TTAC-NQ degrades its performance under even slow streams (e.g.,$\eta = 1/8$), while SHOT reduces its error rate under faster streams. Furthermore, EATA is consistently outperforming all other considered approaches under different stream speeds.

### B.4. Evaluation on Other Benchmarks

We report the error rates on all corruptions of ImageNet-3DCC (Kar et al., 2022), along with the overall average error rate, in Table 9. The conclusions we draw for ImageNet-3DCC (Kar et al., 2022) are very similar to the ones observed on ImageNet-C (Hendrycks & Dietterich, 2019) (in Section 4.1). We observe that efficient methods, with $\mathcal{C}(g) = 1$, such as LAME and BN, maintain performance. Furthermore, the performance of some TTA methods (Wang et al., 2020; Niu14 et al., 2023; Niu et al., 2022; Wang et al., 2022) degrades in the online setup, while others that use pseudo labeling (Lee et al., 2013; Liang et al., 2020) actually improve. This degradation seems to be directly proportional to the amount of data a method misses according to its $\mathcal{C}(g)$.

Table 9: **Episodic Error Rate on ImageNet-3DCommonCorruptions.** We report the error rate of different TTA methods on ImageNet-3DCC (Kar et al., 2022) benchmark under both the realistic and offline setups. A lower error rate indicates a better TTA method. The highlighted numbers indicate a better performance per method across setups. Episodic means the model will adapt to one corruption at a time. The model is reset back to the base model when moving to the next corruption. The offline setup corresponds to reproducing the reported performance of every method. The first sub-table corresponds to methods that incur none or few additional computations, i.e.,$\mathcal{C}(g) = 1$. We show that methods generally perform worse in the more realistic setup. The more computationally complex the TTA method is, the fewer data it will adapt to, and the worse its performance.

| Method | Realistic | Depth of field | | Noise | | | Lighting | Weather | Video | | | Camera motion | | Avg. | Δ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Near focus | Far focus | Color quant. | ISO noise | Low light | Flash | Fog 3D | Bit error | H.265 ABR | H.265 CRF | XY-mot. blur | Z-mot. blur | | |
| Source | ✓ | 46.9 | 55.6 | 82.5 | 94.0 | 71.7 | 78.7 | 75.3 | 88.6 | 70.6 | 65.4 | 82.0 | 75.3 | 73.9 | - |
| AdaBN | ✓ | 45.2 | 55.0 | 71.8 | 76.8 | 64.1 | 80.8 | 75.0 | 91.8 | 80.9 | 76.7 | 79.1 | 67.5 | 72.1 | - |
| LAME | ✓ | 45.3 | 55.0 | 71.9 | 76.9 | 64.1 | 80.8 | 75.1 | 91.8 | 80.9 | 76.8 | 79.2 | 67.6 | 72.1 | - |
| BN | ✓ | 43.9 | 54.3 | 72.3 | 76.6 | 60.9 | 80.1 | 72.4 | 90.9 | 78.7 | 73.8 | 76.9 | 65.6 | 70.5 | - |
| PL | ✗ | 39.8 | 49.8 | 65.5 | 72.6 | 48.9 | 79.0 | 66.1 | 97.5 | 92.1 | 86.2 | 88.7 | 57.6 | 70.3 | (-1.6) |
| | ✓ | 41.0 | 51.3 | 66.5 | 71.5 | 52.8 | 77.4 | 68.1 | 95.6 | 86.0 | 78.7 | 77.0 | 59.2 | 68.7 | |
| SHOT | ✗ | 43.0 | 53.6 | 67.1 | 64.2 | 51.9 | 81.1 | 73.2 | 97.2 | 83.5 | 77.8 | 77.3 | 60.1 | 69.2 | (-2.2) |
| | ✓ | 41.7 | 51.4 | 64.4 | 63.8 | 51.6 | 77.5 | 71.6 | 95.1 | 79.9 | 74.6 | 73.7 | 58.5 | 67.0 | |
| SHOT-IM | ✗ | 42.2 | 52.7 | 66.6 | 63.7 | 51.0 | 81.0 | 72.1 | 97.0 | 83.3 | 77.6 | 75.6 | 59.2 | 68.5 | (-1.9) |
| | ✓ | 41.2 | 51.2 | 64.4 | 63.3 | 51.3 | 77.5 | 70.9 | 94.9 | 79.4 | 74.1 | 72.3 | 58.3 | 66.6 | |
| TENT | ✗ | 39.9 | 49.6 | 62.4 | 62.2 | 50.7 | 75.6 | 68.5 | 91.6 | 75.7 | 70.2 | 70.4 | 57.0 | 64.5 | (+2.3) |
| | ✓ | 41.7 | 51.4 | 65.5 | 67.2 | 54.7 | 77.4 | 70.1 | 90.7 | 76.8 | 71.9 | 74.0 | 60.8 | 66.8 | |
| SAR | ✗ | 40.3 | 50.0 | 62.0 | 61.2 | 50.6 | 73.8 | 65.8 | 90.1 | 73.9 | 68.8 | 69.1 | 56.8 | 63.5 | (+6.9) |
| | ✓ | 44.9 | 54.7 | 71.1 | 75.4 | 62.6 | 80.3 | 73.8 | 91.7 | 80.5 | 76.1 | 78.6 | 66.9 | 71.4 | |
| ETA | ✗ | 38.7 | 47.9 | 59.1 | 56.7 | 46.8 | 71.0 | 62.1 | 90.6 | 72.8 | 67.3 | 64.7 | 52.9 | 60.9 | (+2.3) |
| | ✓ | 39.7 | 49.3 | 61.6 | 60.7 | 50.0 | 73.5 | 65.2 | 90.3 | 74.4 | 69.1 | 68.8 | 55.9 | 63.2 | |
| CoTTA | ✗ | 40.8 | 50.9 | 66.3 | 68.3 | 54.6 | 77.2 | 68.0 | 90.2 | 76.4 | 71.1 | 73.1 | 60.4 | 66.4 | (+9.2) |
| | ✓ | 55.4 | 63.1 | 74.1 | 77.0 | 64.7 | 83.4 | 78.1 | 93.7 | 84.0 | 80.3 | 81.7 | 71.9 | 75.6 | |
| TTAC-NQ | ✗ | 40.7 | 50.5 | 61.0 | 61.1 | 51.5 | 72.8 | 66.6 | 93.8 | 81.1 | 74.7 | 75.7 | 59.1 | 65.7 | (+7.9) |
| | ✓ | 49.9 | 57.0 | 69.3 | 72.3 | 58.9 | 79.8 | 76.3 | 95.8 | 86.5 | 83.0 | 84.6 | 69.8 | 73.6 | |
| EATA | ✗ | 38.6 | 47.8 | 59.2 | 56.6 | 46.9 | 71.2 | 62.2 | 90.9 | 72.5 | 67.4 | 64.6 | 52.9 | 60.9 | (+2.2) |
| | ✓ | 39.8 | 49.3 | 61.6 | 60.5 | 49.9 | 73.5 | 64.8 | 90.6 | 73.7 | 69.1 | 68.6 | 55.7 | 63.1 | |

## C. Single Model Evaluation Scheme

In Section 3.2, we assume $f_{\theta_t}$ can generate predictions whenever $g$ is occupied with adapting to a batch. This setup assumes the capacity to concurrently deploy two models. However, this assumption might be unfair to methods with $\mathcal{C}(g) = 1$, since it allows expensive methods to skip batches without large penalties. We thus also study the case where only one model can be deployed.

Studying this setup requires establishing a policy on how samples missed by the TTA method $g$ are treated. That is, when $g$ is busy adapting, all skipped samples still must be predicted without access to $f_{\theta_t}$. Depending on the application, this prediction could leverage prior knowledge about the problem *e.g.* temporal correlation across samples, or the bias of the distribution. In our setup, we consider the most strict scenario in which, whenever $g$ is busy, a random classifier generates predictions for the incoming samples. This naive design choice results from our evaluation on ImageNet-based datasets, which contain images whose classes display no bias nor temporal correlation. We conduct episodic evaluation, similar to Section 4.1, on ImageNet-C dataset. We average the error rates per corruption category (*e.g.* averaging error rates for gaussian, shot, and impulse noises) and present the results of this study in Table 10. We draw the following observation.

**Single model evaluation strongly favors methods with $\mathcal{C}(g) = 1$.** We observe that all models that are slower than the stream are heavily penalized to the point that using the original pre-trained model becomes a better alternative. However, methods that can be as fast as the stream, like AdaBN or BN, become the best alternative due to their speed. This result encourages more research toward developing efficient TTA methods that have negligible additional computational overhead.

## D. Results on ResNet18

In our experiments in the main paper, we focused on the standard ResNet18-architecture, following the common practice in the literature. Here, and for completeness, we extend our results to cover the smaller and more efficient ResNet18 architecture. Teble 11 summarizes the episodic evaluation of 6 TTA methods on ImageNet-C dataset. Similar to our conclusions in the episodic evaluation section in the main paper, more expensive adaptation methods degrade more under our realistic evaluation scheme.

Table 10: **Per Corruption Category Average Error Rate Using Single Model Evaluation on ImageNet-C.** We report the average error rate per corruption category of different TTA methods under single model realistic evaluation mode on ImageNet-C. Single model mode assumes the deployment of a single model $g$ instead of two under a constant speed stream $\mathcal{S}$. We assume the most extreme scenario, that is if a model $g$ is occupied adapting to a batch, the incoming batch is fed to a random classifier. We observe that the best TTA methods to use in this scenario are AdaBN (Li et al., 2016) and BN (Schneider et al., 2020), which simply adapt the BN statistics.

| Method | Realistic | Noise | Blur | Weather | Digital | Avg. |
|---|---|---|---|---|---|---|
| Source | ✓ | 97.7 | 83.8 | 69.1 | 81.4 | 82.0 |
| AdaBN | ✓ | 84.5 | 76.1 | 54.9 | 62.7 | 68.5 |
| BN | ✓ | 84.1 | 73.1 | 54.2 | 59.9 | 66.7 |
| SHOT | ✓ | 92.6 | 91.3 | 87.0 | 88.5 | 89.7 |
| TENT | ✓ | 91.9 | 89.4 | 83.0 | 85.0 | 87.0 |
| SAR | ✓ | 95.6 | 94.0 | 90.1 | 91.3 | 92.6 |
| EATA | ✓ | 89.4 | 87.6 | 82.0 | 83.2 | 85.3 |
| TTAC-NQ | ✓ | 96.6 | 96.9 | 96.3 | 96.4 | 96.5 |

Table 11: Evaluating different TTA methods with ResNet-18 architecture on ImageNet-C. We report the average error rate across all different types of corruptions (lower is better). TTA methods generally perform worse in the more realistic setup. The more computationally complex the TTA method is, the less data it will adapt to, and the worse is its performance.

| Method | Basic | BN | SHOT | Tent | EATA | SAR |
|---|---|---|---|---|---|---|
| Current | 85.4 | 70.1 | 64.4 | 64.9 | 59.7 | 63.8 |
| Realistic | 85.4 | 70.1 | 64.5 | 68.3 | 63.2 | 69.5 |
| Diff | - | - | 0.1 | 3.4 | 3.5 | 5.7 |

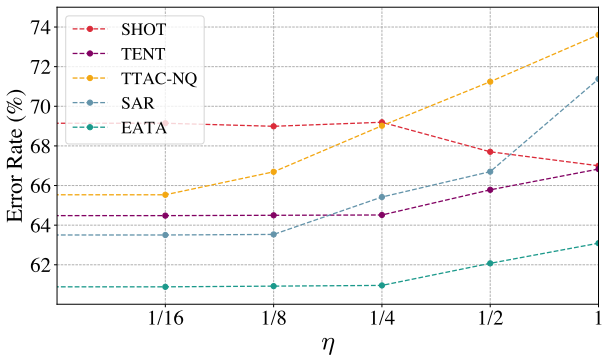



Figure 8: **Average Error Rate on ImageNet-3DCC Under Slower Stream Speeds.** We report the average error rate for several TTA methods on ImageNet-3DCC under slower stream speeds. In our proposed online model evaluation, the stream speed $r$ is normalized by the time needed for a forward pass using the base model. We evaluate different TTA methods under a stream with speed $\eta r$ with $\eta \in (0, 1]$. An $\eta = 1/16$ means the stream is 16 times slower than the forward pass of the base model. We report the standard deviation across 3 random seeds. Different TTA methods degrade differently when varying $\eta$.