NOTE: Robust Continual Test-time Adaptation Against Temporal Correlation

Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee
KAIST
Daejeon, South Korea
{taesik.gong,jongheonj,maxkim139,yewon.e.kim,jinwoos,profsj}@kaist.ac.kr

Abstract

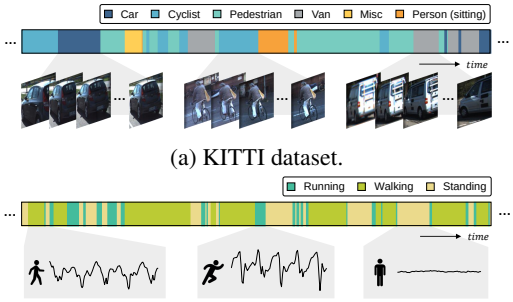
Test-time adaptation (TTA) is an emerging paradigm that addresses distributional shifts between training and testing phases without additional data acquisition or labeling cost; only unlabeled test data streams are used for continual model adaptation. Previous TTA schemes assume that the test samples are independent and identically distributed (i.i.d.), even though they are often temporally correlated (non-i.i.d.) in application scenarios, e.g., autonomous driving. We discover that most existing TTA methods fail dramatically under such scenarios. Motivated by this, we present a new test-time adaptation scheme that is robust against non-i.i.d. test data streams. Our novelty is mainly two-fold: (a) Instance-Aware Batch Normalization (IABN) that corrects normalization for out-of-distribution samples, and (b) Prediction-balanced Reservoir Sampling (PBRS) that simulates i.i.d. data stream from non-i.i.d. stream in a class-balanced manner. Our evaluation with various datasets, including real-world non-i.i.d. streams, demonstrates that the proposed robust TTA not only outperforms state-of-the-art TTA algorithms in the non-i.i.d. setting, but also achieves comparable performance to those algorithms under the i.i.d. assumption. Code is available at https://github.com/TaesikGong/NOTE.

1 Introduction

While deep neural networks (DNNs) have been successful in several applications, their performance degrades under distributional shifts between the training data and test data [32]. This distributional shift hinders DNNs from being widely deployed in many risk-sensitive applications, such as autonomous driving, medical imaging, and mobile health care, where new types of test data unseen during training could result in undesirable disasters. For instance, Tesla Autopilot has caused 12 "deaths" until recently [2]. To address this problem, *test-time adaptation* (TTA) aims to adapt DNNs to the target/unseen domain with only unlabeled test data streams, without any additional data acquisition or labeling cost. Recent studies reported that TTA is a promising, practical direction to mitigate distributional shifts [29, 33, 41, 4, 44].

Prior TTA studies typically assume (implicitly or explicitly) that a target test sample $\mathbf{x}_t$ at time t and the corresponding ground-truth label y_t (unknown to the learner) are independent and identically distributed (i.i.d.) following a target domain, i.e., $(\mathbf{x}_t, y_t)$ is drawn independently from a time-invariant distribution $P_{\mathcal{T}}(\mathbf{x}, y)$. However, the distribution of online test samples often changes across the time axis, i.e., $(\mathbf{x}_t, y_t) \sim P_{\mathcal{T}}(\mathbf{x}, y \mid t)$ in many applications; for instance, AI-powered self-driving car's object encounter will be dominated by cars while driving on the highway, but less dominated by them on downtown where other classes such as pedestrians and bikes are visible. In human activity recognition, some activities last for a short term (e.g., a fall down), whereas certain activities last longer (e.g., a sleep). Figure 1 illustrates that some data distributions in the real world, such as autonomous driving and human activity recognition, are often temporally correlated. Considering that most

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) KITTI dataset.

(b) HARTH dataset.

Figure 1: Illustration of test sample distributions along the time axis from real-world datasets: (a) autonomous driving (KITTI [9]) and (b) human activity recognition (HARTH [25]). They are temporally correlated.

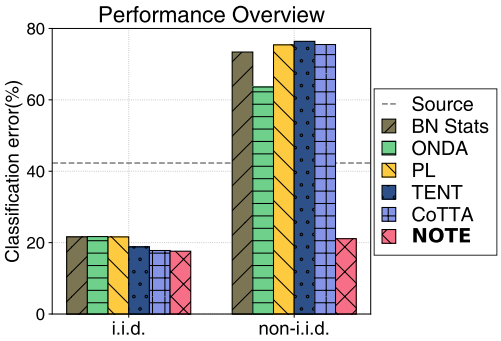


Figure 2: Average classification error (%) of existing TTA methods and our method (NOTE) on CIFAR10-C [13]. The error rates significantly increase under the non-i.i.d. setting compared with the i.i.d. setting. Lower is better.

existing TTA algorithms simply use an incoming batch of test samples for adaptation [29, 33, 41, 44], the model might be biased towards these imbalanced samples under the temporally correlated test streams. Figure 2 compares the performance of the state-of-the-art TTA algorithms under the i.i.d. and non-i.i.d.[1] conditions. While the TTA methods perform well under the i.i.d. assumption, their errors increase under the non-i.i.d. case. Adapting to temporally correlated test data results in overfitting to temporal distributions, which in turn harms the generalization of the model.

Motivated by this, we present a NOn-i.i.d. TEst-time adaptation scheme, *NOTE*, that consists of two components: (a) Instance-Aware Batch Normalization (IABN) and (b) Prediction-Balanced Reservoir Sampling (PBRS). First, we propose a novel normalization layer, IABN, that eliminates the dependence on temporally correlated data for adaptation while being robust to distribution shifts. IABN detects out-of-distribution instances *sample by sample* and corrects via instance-aware normalization. The key idea of IABN is synthesizing Batch Normalization (BN) [16] with Instance Normalization (IN) [37] in a unique way; it calculates how different the learned knowledge (BN) is from the current observation (IN) and corrects the normalization by the deviation between IN and BN. Second, we present PBRS that resolves the problem of overfitting to non-i.i.d. samples by mimicking i.i.d. samples from non-i.i.d. streams. By utilizing predicted labels of the model, PBRS aims for both time-uniform sampling and class-uniform sampling from the non-i.i.d. streams, and stores the 'simulated' i.i.d. samples in memory. With the i.i.d.-like batch in the memory, PBRS enables the model to adapt to the target domain without being biased to temporal distributions.

We evaluate NOTE with state-of-the-art TTA baselines [29, 33, 22, 27, 4, 44] on multiple datasets, including common TTA benchmarks (CIFAR10-C, CIFAR100-C, and ImageNet-C [13]) and real-world non-i.i.d. datasets (KITTI [9], HARTH [25], and ExtraSensory [38]). Our results suggest that NOTE not only significantly outperforms the baselines under non-i.i.d. test data, e.g., it achieves a 21.1% error rate on CIFAR10-C which is on average 15.1% lower than the state-of-the-art method [4], but also shows comparable performance even under the i.i.d. assumption, e.g., 17.6% error on CIFAR10-C where the best baseline [44] achieves 17.8% error. Our ablative study demonstrates the individual effectiveness of IABN and PBRS and further highlights their synergy when jointly used.

Finally, we summarize the key characteristics of NOTE. First, NOTE is a batch-free inference algorithm (requiring a single instance for inference), different from the state-of-the-art TTA algorithms [29, 33, 41, 4, 44] where a batch of test data is necessary for inference to estimate normalization statistics (mean and variance). Second, while some recent TTA methods leverage augmentations to improve performance at the cost of additional forwarding passes [35, 44], NOTE requires only a single forwarding pass. NOTE updates only the normalization statistics and affine parameters in IABN, which is, e.g., approximately 0.02% of the total trainable parameters in ResNet18 [12]. Third, NOTE's additional memory overhead is negligible. It merely stores predicted labels of test data to run PBRS. These characteristics make NOTE easy to apply to any existing AI system and particularly, are beneficial in latency-sensitive tasks such as autonomous driving and human health monitoring.

[1] We use the terms *temporally correlated* and *non-i.i.d.* interchangeably in the context of test-time adaptation.

2 Background

2.1 Problem setting: test-time adaptation with non-i.i.d. streams

Test-time adaptation. Let $\mathcal{D}_\mathcal{S} = \{\mathcal{X}^\mathcal{S}, \mathcal{Y}\}$ be the data from the source domain and $\mathcal{D}_\mathcal{T} = \{\mathcal{X}^\mathcal{T}, \mathcal{Y}\}$ be the data from the target domain to adapt to. Each data instance and the corresponding ground-truth label pair $(\mathbf{x}_i, y_i) \in \mathcal{X}^\mathcal{S} \times \mathcal{Y}$ in the source domain follows a probability distribution $P_\mathcal{S}(\mathbf{x}, y)$. Similarly, each target test sample and the corresponding label at test time t, $(\mathbf{x}_t, y_t) \in \mathcal{X}^\mathcal{T} \times \mathcal{Y}$, follows a probability distribution $P_\mathcal{T}(\mathbf{x}, y)$ where y_t is unknown for the learner. The standard covariate shift assumption in domain adaptation is defined as $P_\mathcal{S}(\mathbf{x}) \neq P_\mathcal{T}(\mathbf{x})$ and $P_\mathcal{S}(y|\mathbf{x}) = P_\mathcal{T}(y|\mathbf{x})$ [32]. Unlike traditional domain adaptation that uses $\mathcal{D}_\mathcal{S}$ and $\mathcal{X}^\mathcal{T}$ collected beforehand for adaptation, test-time adaptation (TTA) continually adapts a pre-trained model $f_\theta(\cdot)$ from $\mathcal{D}_\mathcal{S}$ by utilizing only $\mathbf{x}_t$ obtained at test time t.

TTA on non-i.i.d. streams. Note that previous TTA mechanisms typically assume that each target sample $(\mathbf{x}_t, y_t) \in \mathcal{X}^\mathcal{T} \times \mathcal{Y}$ is independent and identically distributed (i.i.d.) following a time-invariant distribution $P_\mathcal{T}(\mathbf{x}, y)$. However, the data obtained at test time is non-i.i.d. in many scenarios. By non-i.i.d., we refer to distribution changes over time, i.e., $(\mathbf{x}_t, y_t) \sim P_\mathcal{T}(\mathbf{x}, y \mid t)$, which is a practical setting in many real world applications [46].

2.2 Batch normalization in TTA

Batch Normalization (BN) [16] is a widely-used training technique in deep neural networks as it reduces the internal covariant shift problem. Let $\mathbf{f} \in \mathbb{R}^{B \times C \times L}$ denote a batch of feature maps in general, where B, C, and L denote the batch size, the number of channels, and the size of each feature map, respectively. Given the statistics of the feature maps for normalization, say mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$, BN is *channel-wise*, i.e., $\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \in \mathbb{R}^C$ and computes:

$$\mathrm{BN}(\mathbf{f}_{:,c,:}; \boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2) := \gamma \cdot \frac{\mathbf{f}_{:,c,:} - \boldsymbol{\mu}_c}{\sqrt{\boldsymbol{\sigma}_c^2 + \epsilon}} + \beta, \tag{1}$$

where γ and β are the affine parameters followed by the normalization, and $\epsilon > 0$ is a small constant for numerical stability.

Although a conventional way of computing BN in test-time is to set $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ as those estimated from *training* (or source) data, say $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\sigma}}^2$, the state-of-the-art TTA methods based on adapting BN layers [29, 33, 41, 44] instead use the statistics computed directly from the recent test batch to de-bias distributional shifts at test-time, i.e.:

$$\hat{\boldsymbol{\mu}}_c := \frac{1}{BL} \sum_{b,l} \mathbf{f}_{b,c,l}, \text{ and } \hat{\boldsymbol{\sigma}}_c^2 := \frac{1}{BL} \sum_{b,l} (\mathbf{f}_{b,c,l} - \hat{\boldsymbol{\mu}}_c)^2. \tag{2}$$

This practice is simple yet effective under distributional shifts and is thus adopted in many recent TTA studies [29, 33, 41, 44]. Based on the test batch statistics, they often further adapt the affine parameters via entropy minimization of the model outputs [41] or update the entire parameters with self-training [44].

3 Method

In the same vein as previous work [29, 33, 41], we focus on adapting BN layers in the given model to perform TTA, and this includes essentially two approaches: (a) re-calibrating (or adapting) channel-wise statistics for normalization (instead of using those learned from training), and (b) adapting the affine parameters (namely, γ and β) after the normalization with respect to a certain objective based on test samples, e.g., the entropy minimization of model outputs [41].

Under scenarios where test data are temporally correlated, however, naïvely adapting to the incoming batch of test samples [29, 33, 41, 44] could be problematic for both approaches: the batch is now more likely to (a) remove instance-wise variations that are actually useful to predict y, i.e., the "contents" rather than "styles" through normalization, and (b) include a bias in $p(y)$ rather than uniform, which can negatively affect the test-time adaptation objective such as entropy minimization.

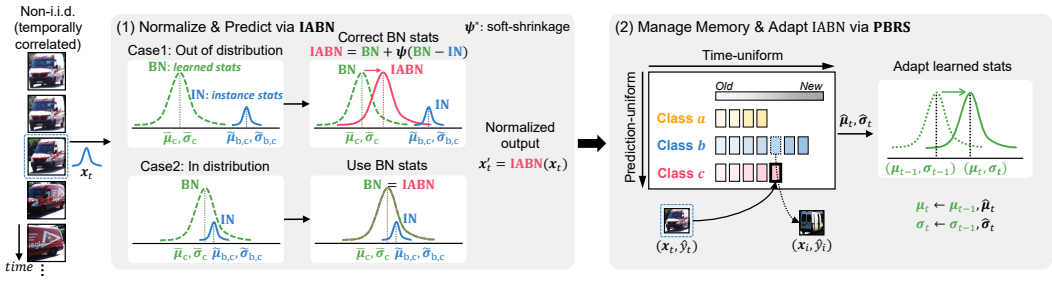


Figure 3: An overview of the proposed methodology: Instance-Aware Batch Normalization (IABN) and Prediction-Balanced Reservoir Sampling (PBRS). IABN aims to detect non-i.i.d. streams and in turn corrects the normalization for inference. PBRS manages data in a time- and prediction-uniform manner from non-i.i.d. data streams and gradually adapts IABNs with the balanced data afterwards.

We propose two approaches to tackle each of the failure modes of adapting BN under temporal correlation. Our method consists of two components: (a) Instance-Aware Batch Normalization (IABN) (Section §3.1) to overcome the limitation of BN under distribution shift and (b) Prediction-Balanced Reservoir Sampling (PBRS) (Section §3.2) to combat with the temporal correlation of test batches. Figure 3 illustrates the overall workflow of NOTE with IABN and PBRS.

3.1 Instance-Aware Batch Normalization

As described in Section §2.2, recent TTA algorithms rely solely on the test batch to re-calculate BN statistics. We argue that this common practice does not successfully capture the feature statistics to normalize the feature map $\mathbf{f} \in \mathbb{R}^{B \times C \times L}$ under temporal correlation in the test batch B. In principle, standardizing a given feature map $\mathbf{f}_{:,c,:}$ by the statistics $\hat{\boldsymbol{\mu}}_c, \hat{\boldsymbol{\sigma}}_c^2$ computed across B and L is posited on premise that averaging information across B can marginalize out uninformative instance-wise variations for predicting y. Under temporal correlation in B, however, this assumption is no longer valid, and averaging across B may not fully de-correlate useful information in $\mathbf{f}_{:,c,:}$ from $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$.

In an attempt to bypass such an "over-whitening" effect of using $\hat{\boldsymbol{\mu}}_c$ and $\hat{\boldsymbol{\sigma}}_c^2$ in test-time under temporal correlation, we propose correcting normalization statistics on a *per-sample basis*: specifically, instead of completely switching from the original statistics of $(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}^2)$ into $(\hat{\boldsymbol{\mu}}_c, \hat{\boldsymbol{\sigma}}_c^2)$, our proposed *Instance-Aware Batch Normalization* (IABN) considers the *instance-wise* statistics $\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2 \in \mathbb{R}^{B,C}$ of $\mathbf{f}$ similarly to Instance Normalization (IN) [37], namely:

$$\tilde{\boldsymbol{\mu}}_{b,c} := \frac{1}{L} \sum_l \mathbf{f}_{b,c,l} \text{ and } \tilde{\boldsymbol{\sigma}}_{b,c}^2 := \frac{1}{L} \sum_l (\mathbf{f}_{b,c,l} - \tilde{\boldsymbol{\mu}}_{b,c})^2. \tag{3}$$

We assume that $\tilde{\boldsymbol{\mu}}_{b,c}$ and $\tilde{\boldsymbol{\sigma}}_{b,c}^2$ follow the *sampling distribution* of a sample size L in $\mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}^2)$ as the population. Then the corresponding variances for the sample mean $\tilde{\boldsymbol{\mu}}_{b,c}$ and the sample variance $\tilde{\boldsymbol{\sigma}}_{b,c}^2$ can be calculated as:

$$s_{\tilde{\boldsymbol{\mu}},c}^2 := \frac{\bar{\boldsymbol{\sigma}}_c^2}{L} \text{ and } s_{\tilde{\boldsymbol{\sigma}}^2,c}^2 := \frac{2\bar{\boldsymbol{\sigma}}_c^4}{L-1}. \tag{4}$$

IABN corrects $(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}^2)$ only in cases when $\tilde{\boldsymbol{\mu}}_{b,c}$ (and $\tilde{\boldsymbol{\sigma}}_{b,c}^2$) significantly differ from $\bar{\boldsymbol{\mu}}_c$ (and $\bar{\boldsymbol{\sigma}}_c^2$). Specifically, we propose to use the following statistics for TTA:

$$\boldsymbol{\mu}_{b,c}^{\texttt{IABN}} := \bar{\boldsymbol{\mu}}_c + \psi(\tilde{\boldsymbol{\mu}}_{b,c} - \bar{\boldsymbol{\mu}}_c; \alpha s_{\tilde{\boldsymbol{\mu}},c}), \text{ and } (\boldsymbol{\sigma}_{b,c}^{\texttt{IABN}})^2 := \bar{\boldsymbol{\sigma}}_c^2 + \psi(\tilde{\boldsymbol{\sigma}}_{b,c}^2 - \bar{\boldsymbol{\sigma}}_c^2; \alpha s_{\tilde{\boldsymbol{\sigma}}^2,c}),$$

$$\text{where } \psi(x;\lambda) = \begin{cases} x - \lambda, & \text{if } x > \lambda \\ x + \lambda, & \text{if } x < -\lambda \\ 0, & \text{otherwise} \end{cases} \text{ is the soft-shrinkage function.} \tag{5}$$

$\alpha \geq 0$ is the hyperparameter of IABN that determines the confidence level of the BN statistics. A high value of α relies more on the learned statistics (BN), while a low value of α is in favor of the current statistics measured from the instance. Finally, the output of IABN can be described as:

Algorithm 1 Prediction-Balanced Reservoir Sampling

Input: target stream $\mathbf{x}_t \sim P_{\mathcal{T}}(\mathbf{x}|t)$, memory bank M of capacity N
$M[i] \leftarrow \phi$ for $i = 1, \cdots N$; and $n[c] \leftarrow 0$ for $c \in \mathcal{Y}$
for $t \in \{1, \cdots, T\}$ **do**
  $n[\hat{y}_t] \leftarrow n[\hat{y}_t] + 1$ // increase the number of samples encountered for the class
  $m[c] \leftarrow |\{(\mathbf{x}, y) \in M | y = c\}|$ for $c \in \mathcal{Y}$ // count instances per class in memory
  if $|M| < N$ **then** // if memory is not full
    Add $(\mathbf{x}_t, \hat{y}_t)$ to M
  else
    $C^* \leftarrow \arg\max_{c \in \mathcal{Y}} m[c]$ // get majority class(es)
    if $\hat{y}_t \notin C^*$ **then** // if the new sample is not majority ▷ Prediction-Balanced
      Randomly pick $M[i] := (\mathbf{x}_i, \hat{y}_i)$ where $\hat{y}_i \in C^*$
      $M[i] \leftarrow (\mathbf{x}_t, \hat{y}_t)$ // replace it with a new sample
    else ▷ Reservoir Sampling
      Sample $p \sim \mathrm{Uniform}(0, 1)$
      if $p < m[\hat{y}_t]/n[\hat{y}_t]$ **then**
        Randomly pick $M[i] := (\mathbf{x}_i, \hat{y}_i)$ where $\hat{y}_i = \hat{y}_t$
        $M[i] \leftarrow (\mathbf{x}_t, \hat{y}_t)$ // replace it with a new sample

$$\mathrm{IABN}(\mathbf{f}_{b,c,:}; \bar{\boldsymbol{\mu}}_c, \bar{\boldsymbol{\sigma}}_c^2; \tilde{\boldsymbol{\mu}}_{b,c}, \tilde{\boldsymbol{\sigma}}_{b,c}^2) := \gamma \cdot \frac{\mathbf{f}_{b,c,:} - \boldsymbol{\mu}_{b,c}^{\texttt{IABN}}}{\sqrt{(\boldsymbol{\sigma}_{b,c}^{\texttt{IABN}})^2 + \epsilon}} + \beta. \tag{6}$$

Observe that IABN becomes IN and BN when $\alpha = 0$ and $\alpha = \infty$, respectively. If one chooses too small $\alpha \geq 0$, IABN may remove useful features, e.g., styles, of input (as with IN), which can degrade the overall classification (or regression) performance [30]. Hence, it is important to choose an appropriate α. Nevertheless, we found that a good choice of α is not too sensitive across tested scenarios, where we chose $\alpha = 4$ for all experiments. This way, IABN can be robust to distributional shifts without the risk of eliminating crucial information to predict y.

3.2 Adaptation via Prediction-Balanced Reservoir Sampling

Temporally correlated distributions lead to an undesirable bias in $p(y)$, and thus adaptation with a batch of consecutive test samples negatively impacts the adaptation objective, such as entropy minimization [41]. To combat this imbalance, we propose Prediction-Balanced Reservoir Sampling (PBRS) that mimics i.i.d. samples from temporally correlated streams with the assistance of a small (e.g., a mini-batch size) memory. PBRS combines *time-uniform* sampling and *prediction-uniform* sampling to simulate i.i.d. samples from the non-i.i.d. streams. For time-uniform sampling, we adopt reservoir sampling (RS) [40], a proven random sampling algorithm to collect time-uniform data in a single pass on a stream without the prior knowledge of the total length of data. For prediction-uniform sampling, we first use the predicted labels to compute the majority class(es) in the memory. We then replace a random instance of the majority class(es) with a new sample. We detail the algorithm of PBRS as a pseudo-code in Algorithm 1. We found that these two heuristics can effectively balance samples among both time and class axes, which mitigates the bias in temporally correlated data.

With the stored samples in the memory, we update the normalization statistics and affine parameters in the IABN layers. Note that IABN assumes $\tilde{\boldsymbol{\mu}}_{b,c}$ and $\tilde{\boldsymbol{\sigma}}_{b,c}^2$ follow the sampling distribution of $\mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}^2)$ and corrects the normalization if $\tilde{\boldsymbol{\mu}}_{b,c}$ and $\tilde{\boldsymbol{\sigma}}_{b,c}^2$ are out of distribution. While IABN is resilient to distributional shifts to a certain extent, the assumption might not hold under severe distributional shifts. Therefore, we aim to find better estimates of $\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}^2$ in IABN under distributional shifts via PBRS. Specifically, we update the normalization statistics, namely the means $\boldsymbol{\mu}$ and variances $\boldsymbol{\sigma}^2$, via exponential moving average: (a) $\boldsymbol{\mu}_t = (1-m)\boldsymbol{\mu}_{t-1} + m\frac{N}{N-1}\hat{\boldsymbol{\mu}}_t$ and (b) $\boldsymbol{\sigma}_t^2 = (1-m)\boldsymbol{\sigma}_{t-1}^2 + m\frac{N}{N-1}\hat{\boldsymbol{\sigma}}_t^2$ where m is a momentum and N is the size of the memory. We further optimize the affine parameters, scaling factor γ and bias term β via a single backward pass with entropy minimization, similar to a previous study [41]. These parameters account for only around 0.02% of the total trainable parameters in ResNet18 [12]. The IABN layers are adapted with the

N samples in the memory every N test samples. We set the memory size N as 64 following the common batch size of existing TTA methods [33, 4, 41] to ensure a fair memory constraint.

3.3 Inference

NOTE infers each sample via a single forward pass with IABN layers. Note that NOTE requires only a single instance for inference, different from the state-of-the-art TTA methods [29, 33, 41, 4, 44] that require batches for every inference. Moreover, NOTE requires only one forwarding pass for inference, while multiple forward passes are required in other TTA methods that utilize augmentations [35, 44]. The batch-free single-forward inference of NOTE is beneficial in latency-sensitive tasks such as autonomous driving and human health monitoring. After inference, NOTE determines whether to store the sample and predicted label to the memory via PBRS.

4 Experiments

We implemented NOTE and the baselines via the PyTorch framework [31].[2] We ran all experiments with three random seeds and report the means and standard deviations. Additional experimental details, e.g., hyperparameters of the baselines, datasets, are specified in Appendix A.

Baselines. We consider the following baselines including the state-of-the-art test-time adaptation algorithms: **Source** evaluates the model trained from the source data directly on the target data without adaptation. Test-time normalization (**BN stats**) [29, 33] updates the BN statistics from a batch of test data. Online Domain Adaptation (**ONDA**) [27] adapts batch normalization statistics to target domains via a batch of target data with an exponential moving average. Pseudo-Label (**PL**) [22] optimizes the trainable parameters in BN layers via hard pseudo labels. We update the BN layers only in PL, as done in previous studies [41, 44]. Test entropy minimization (**TENT**) [41] updates the BN parameters via entropy minimization. Laplacian Adjusted Maximum-likelihood Estimation (**LAME**) [4] takes a more conservative approach; it modifies the classifier's output probability and not the internal parameters of the model itself. By doing so, it prevents the model parameters from over-adapting to the test batch. Continual test-time adaptation (**CoTTA**) [44] reduces the error accumulation by using weight-averaged and augmentation-averaged predictions. It avoids catastrophic forgetting by stochastically restoring a part of the neurons to the source pre-trained weights.

Adaptation and hyperparameters. We assume the model pre-trained with source data is available for TTA. In NOTE, we replaced BN with IABN during training. We set the test batch size as 64 and the adaptation epoch as one for adaptation, which is the most common setting among the baselines [33, 4, 41]. Similarly, we set the memory size N as 64 and adapt the model every 64 samples in NOTE to ensure a fair memory constraint. We conduct online adaptation and evaluation, where the model is continually updated. For the baselines, we adopt the best values for the hyperparameters reported in their papers or the official codes. We followed the guideline to tune the hyperparameters when such a guideline was available [44]. We use fixed values for the hyperparameters of NOTE, soft-shrinkage width $\alpha = 4$ and exponential moving average momentum $m = 0.01$, and update the affine parameters via the Adam optimizer [18] with a learning rate $l = 0.0001$ unless specified. We detailed hyperparameter information of the baselines in Appendix A.1.

4.1 Robustness under corruptions

Datasets. We use CIFAR10-C, CIFAR100-C, and ImageNet-C [13] datasets that are common TTA benchmarks for evaluating the robustness to corruptions [29, 33, 41, 44, 4]. Both CIFAR10/CIFAR100 [19] have 50,000/10,000 training/test data. ImageNet [7] has 1,281,167/50,000 training/test data. CIFAR10/CIFAR100/ImageNet have 10/100/1,000 classes, respectively. CIFAR10-C/CIFAR100-C/ImageNet-C apply 15 types of corruption to CIFAR10/CIFAR100/ImageNet test data. Similar to previous studies [29, 33, 41, 44], we use the most severe corruption level of 5. We use ResNet18 [12] as the backbone network and pre-trained it on the clean training data. Following prior studies [23, 15, 43, 42], we adopt Dirichlet distribution to generate synthetic non-i.i.d. test streams from the originally i.i.d. CIFAR10/100 data. The details are provided in Appendix A.2. We vary the

[2] https://github.com/TaesikGong/NOTE

Table 1: Average classification error (%) and their corresponding standard deviations on CIFAR10-C/100-C and ImageNet-C under temporally correlated (non-i.i.d.) and uniformly distributed (i.i.d.) test data stream. **Bold** fonts indicate the lowest classification errors, while Red fonts show performance degradation after adaptation. Values encompassed by parentheses refers to NOTE used directly with test batches (without using PBRS). Averaged over 3 runs.

	Temporally correlated test stream				**Uniformly distributed test stream**			
Method	CIFAR10-C	CIFAR100-C	ImageNet-C	Avg	CIFAR10-C	CIFAR100-C	ImageNet-C	Avg
Source	42.3 ± 1.1	66.6 ± 0.1	86.1 ± 0.0	65.0	42.3 ± 1.1	66.6 ± 0.1	86.1 ± 0.0	65.0
BN Stats [29]	73.4 ± 1.3	65.0 ± 0.3	96.9 ± 0.0	78.5	21.6 ± 0.4	46.6 ± 0.2	76.0 ± 0.0	48.1
ONDA [27]	63.6 ± 1.0	49.6 ± 0.3	89.0 ± 0.0	67.4	21.7 ± 0.4	46.5 ± 0.1	75.9 ± 0.0	48.0
PL [22]	75.4 ± 1.8	66.4 ± 0.4	98.9 ± 0.0	80.2	21.6 ± 0.2	43.1 ± 0.3	74.4 ± 0.2	46.4
TENT [41]	76.4 ± 2.7	66.9 ± 0.6	96.9 ± 0.0	80.1	18.8 ± 0.2	**40.3 ± 0.2**	76.0 ± 0.0	45.0
LAME [4]	36.2 ± 1.3	63.3 ± 0.3	82.7 ± 0.0	60.7	44.1 ± 0.5	68.8 ± 0.1	86.3 ± 0.0	66.4
CoTTA [44]	75.5 ± 0.7	64.2 ± 0.2	97.0 ± 0.0	78.9	17.8 ± 0.3	44.3 ± 0.2	71.5 ± 0.0	44.6
NOTE	**21.1 ± 0.6**	**47.0 ± 0.1**	**80.6 ± 0.1**	**49.6**	20.1 ± 0.5 (**17.6 ± 0.3**)	46.4 ± 0.0 (41.0 ± 0.2)	**70.3 ± 0.0** (71.7 ± 0.0)	45.6 (**43.4**)

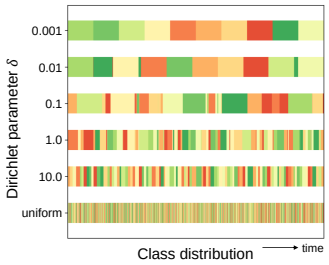
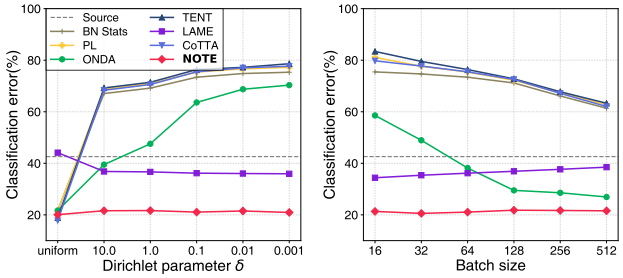


Figure 4: Illustration of synthetic non-i.i.d. streams sampled from Dirichlet distribution varying δ on CIFAR10-C. *uniform* denotes an i.i.d. condition. The lower the δ, the more temporally correlated the distribution.



(a) Effect of Dirichlet concentration parameter δ.

(b) Effect of batch size.

Figure 5: Average classification error (%) under the non-i.i.d. setting with CIFAR10-C dataset. We vary (a) the Dirichlet concentration parameter δ to investigate the impact of the degree of temporal correlation and (b) batch size to understand the behaviors of the TTA methods. Averaged over 3 runs. Lower is better.

Dirichlet concentration parameter δ to simulate diverse streams and visualize the resulting data in Figure 4. We use $\delta = 0.1$ as the default value unless specified. For ImageNet, we sort the test stream as the number of test samples per class is not enough for generating temporally correlated streams via Dirichlet distribution. We additionally provide an experiment with MNIST-C data [28] in the appendix, which shows similar takeaways to our experiments with CIFAR10-C/CIFAR100-C/ImageNet-C.

Overall result. Tables 1 shows the result under the temporally correlated (non-i.i.d.) data and the uniform (i.i.d.) data, respectively. We observe significant performance degradation in the baselines under the temporally correlated setting. For BN Stats, PL, TENT, and CoTTA, this degradation is particularly due to the dependence on the test batch for the re-calculation of the BN statistics. Updating the batch statistics from test data via exponential moving average (ONDA) also suffers from the temporally correlated data. This indicates relying on the test batch for re-calculating the BN statistics indeed cancels out meaningful instance-wise variations under temporal correlation. Interestingly, LAME works better in the non-i.i.d. setting than in the i.i.d. setting, which is consistent with previous reports [4]. The primary reason is, as stated by the authors, it "discourages deviations from the predictions of the pre-trained model," and thus it "does not noticeably help in i.i.d and class-balanced scenarios."

NOTE achieves on average 11.1% improvement over the best baseline (LAME) under the non-i.i.d. setting. With the i.i.d. assumption, NOTE still achieves comparable performance to the baselines. When we know the target samples are i.i.d., we can simply use the test batch without using PBRS. For this variant version of NOTE, we update IABN with incoming test batches directly using a ten times higher learning rate of 0.001 following previous work [41, 44]. We report the result of the variant

Table 2: Average classification error (%) and their corresponding standard deviations on three real test data streams: KITTI, HARTH, and ExtraSensory. **Bold** fonts indicate the lowest classification errors, while Red fonts show performance degradation after adaptation. Averaged over 3 runs.

	Real test stream			
Method	KITTI	HARTH	ExtraSensory	Avg
Source	12.3 ± 2.3	62.6 ± 8.5	50.2 ± 2.2	41.7
BN Stats [29]	35.4 ± 0.5	68.6 ± 1.1	56.0 ± 0.9	53.4
ONDA [27]	26.3 ± 0.5	69.3 ± 1.1	48.2 ± 1.5	47.9
PL [22]	39.0 ± 0.3	64.8 ± 0.6	56.0 ± 0.9	53.3
TENT [41]	39.6 ± 0.2	64.1 ± 0.7	56.0 ± 0.8	53.2
LAME [4]	11.3 ± 2.9	61.0 ± 10.0	50.7 ± 2.7	41.0
CoTTA [44]	35.4 ± 0.6	68.7 ± 1.1	56.0 ± 0.9	53.4
NOTE	**10.9 ± 3.6**	**51.0 ± 5.6**	**45.4 ± 2.6**	**35.8**

version of NOTE in the parentheses, which achieves on average 2.2% improvement further when the i.i.d. assumption is known.

Effect of the degree of temporal correlation. We also investigate the effect of the degree of temporal correlation for TTA algorithms. Figure 5a shows the result. The lower δ is, the severer temporal correlation becomes. The error rates of most of the baselines deteriorate as δ decreases, which shows that the existing TTA baselines are susceptible to temporally correlated data. NOTE shows consistent performance among all δ values, indicating its robustness under temporal correlation.

Effect of batch size. While we experiment with a widely-used value of 64 as the batch size (or memory size in NOTE), one might be curious about the impact of batch size under temporally correlated streams. Figure 5b shows the result with six different batch sizes. As shown, NOTE is not much affected by the batch size, while most of the baselines recovers performance degradation as the batch size increases. This is because a higher batch size has a better chance of adaptation with balanced samples under temporally correlated streams. Increasing the batch size, however, mitigates temporal correlation at the expense of inference latency and adaptation speed.

4.2 Real-distributions with domain shift

Datasets. We evaluate NOTE under three real-world distribution datasets: object detection in autonomous driving (KITTI [9]), human activity recognition (HARTH [25]), and user behavioral context recognition (ExtraSensory [38]). Additional dataset-specific details are in Appendix A.2.

KITTI is a well-known autonomous driving dataset that provides consecutive frames that contains natural temporal correlation in driving contexts. We adapt from KITTI to KITTI-Rain [11] - a dataset that converted KITTI images to rainy images. This contains 7,481/7,800 train/test samples with nine classes. We use ResNet50 [12] pre-trained on ImageNet [8] as the backbone network.

HARTH was collected from 22 users in free-living environments for seven days. Each user was equipped with two three-axial Axivity AX3 accelerometers for recording human activities. We use 15 users collectively as the source domain and the remaining seven users as each target domain, which entails natural domain shifts from source users to target users as different physical conditions make domain shifts across users. HARTH contains 82,544/39,377 train/test samples with 12 classes. We report the averaged error over all target domains. We use four one-dimensional convolutional layers followed by one fully-connected layer as the backbone network for HARTH.

The Extrasensory dataset collected users' own smartphone sensory data (motion sensors, audio, etc.) in the wild for seven days, aiming to capture people's authentic behaviors in their regular activities. We use 16 users as the source domain and seven users as target domains. ExtraSensory includes 17,777/4,862 train/test data with five classes. For ExtraSensory, we use two one-dimensional convolutional layers followed by one fully-connected layer as the backbone network. For both HARTH and ExtraSensory models, a single BN layer follows each convolutional layer.

Table 3: Average classification error (%) and corresponding standard deviations of varying ablation settings on CIFAR10-C/100-C under temporally correlated (non-i.i.d.) and uniformly distributed (i.i.d.) test data stream. **Bold** type indicates those of lowest classification error. Averaged over 3 runs.

	Temporally correlated test stream			**Uniformly distributed test stream**		
Method	CIFAR10-C	CIFAR100-C	Avg	CIFAR10-C	CIFAR100-C	Avg
Source	42.3 ± 1.1	66.6 ± 0.1	54.4	42.3 ± 1.1	66.6 ± 0.1	54.4
IABN	24.6 ± 0.6	54.5 ± 0.1	39.5	24.6 ± 0.6	54.5 ± 0.1	39.5
PBRS	27.5 ± 1.0	51.7 ± 0.2	39.6	25.8 ± 0.2	51.3 ± 0.1	38.5
IABN+RS	**20.5 ± 1.5**	48.2 ± 0.2	34.3	20.7 ± 0.6	48.3 ± 0.3	34.5
IABN+PBRS	21.1 ± 0.6	**47.0 ± 0.1**	**34.0**	**20.1 ± 0.5**	**46.4 ± 0.0**	**33.2**

Result. Table 2 shows the result for the real-world datasets. The overall trend is similar to the temporal correlation experiments with CIFAR10-C/CIFAR100-C datasets, which indicates that the real-world datasets are indeed temporally correlated. NOTE consistently reduces errors after adaptation under real-world distributions. We believe this demonstrates NOTE is a promising method to be utilized in various real-world ML applications with distributional shifts. We illustrate real-time classification error changes for real-world datasets in the appendix.

4.3 Ablation study

We conduct an ablative study to further investigate the effectiveness of the individual components. Table 3 shows the result under both i.i.d. and non-i.i.d. settings. Using IABN alone significantly reduces error rates over Source, demonstrating the effectiveness of correcting normalization for out-of-distribution samples. Using PBRS with BN shows comparable improvement with the IABN-only result. Note that there is only a marginal gap (around 1%) between the non-i.i.d. and i.i.d. results in PBRS. This indicates that PBRS could effectively simulate i.i.d. samples from non-i.i.d. streams. The joint use of IABN and PBRS outperforms using either of them, meaning that PBRS provides IABN with better estimates for the normalizing operation. In addition, PBRS is better than Reservoir Sampling (RS) that has been a strong baseline in continual learning [17, 5]. This shows storing prediction-balanced sampling in addition to time-uniform sampling leads to better adaptation in TTA. We also investigated the joint use of IN and PBRS with the combination of IABN and PBRS on CIFAR100-C, and the result shows that IABN+PBRS (47.0%) achieves lower error rate than IN+PBRS (52.5%) on CIFAR100-C under temporal correlation.

5 Related work

Test-time adaptation. Test-time adaptation (TTA) attempts to overcome distributional shifts with test data without the cost of data acquisition or labeling. TTA adapts to the target domain with only test data on the fly. Most existing TTA algorithms rely on a batch of test samples to adapt [29, 33, 41, 44] to re-calibrate BN layers on the test data. Simply using the statistics of test batch in BN layers improves the robustness under distributional shifts [29, 33]. ONDA [27] updates the BN statistics with test data via exponential moving average. TENT [41] further updates the scaling and bias parameters in BN layers via entropy minimization.

Latest TTA studies consider distribution changes of test data [4, 44]. LAME [4] corrects the output probabilities of a classifier rather than tweaking the model's inner parameters. By restraining the model from over-adapting to the test batch, LAME allows the model to be more robust under non-i.i.d. scenarios. However, LAME does not have noticeable performance gains in class-balanced, standard i.i.d. scenarios. The primary reason is, as stated by the authors, it "discourages deviations from the predictions of the pre-trained model," and thus it "does not noticeably help in i.i.d and class-balanced scenarios." CoTTA [44] aims to adapt to continually changing target environments via weight-averaged teacher model, weight-averaged augmentations, and stochastic restoring. However, CoTTA assumes i.i.d. test data within each domain and updates the entire model that increases computational costs.

There also exist works [35, 24] utilizing domain-specific self-supervision to resolve the distribution shift with test data, but are complementary to ours, i.e., we can also optimize the self-supervised loss

instead of the entropy loss, and not applicable to our setups of real test data stream as designing good self-supervision for these domains is highly non-trivial.

Replay memory. Replay memory is one of the major approaches in continual learning; it manages a buffer to replay previous data for future learning to prevent catastrophic forgetting. Reservoir sampling [40] is a random sampling algorithm that collects time-uniform samples from unknown sample streams with a single pass, and it has been proven to be a strong baseline in continual learning [17, 5]. GSS [1] stores samples to a memory in a way that maximizes the gradient direction among those samples. A recent study modifies reservoir sampling to balance classes under imbalanced data, when the labels are given [6]. Our memory management scheme (PBRS) is inspired by these studies to prevent catastrophic forgetting in test-time adaptations.

6 Discussion and conclusion

This paper highlights that real-world distributions often change across the time axis, and existing test-time adaptation algorithms mostly suffer from the non-i.i.d. test data streams. To address this problem, we present a NOn-i.i.d. TEst-time adaptation algorithm, NOTE. Our experiments evaluated robustness under corruptions and domain adaptation on real-world distributions. The results demonstrate that NOTE not only outperforms the baselines under the non-i.i.d./real distribution settings, but it also shows comparable performance under the i.i.d. assumption. We believe that the insights and findings from this study are a meaningful step toward the practical impact of the test-time adaptation paradigm.

Limitations. NOTE and most state-of-the-art TTA algorithms [29, 22, 27, 33, 41, 44] assume that the backbone networks are equipped with BN (or IABN) layers. While BN is a widely-used component in deep learning, several architectures, such as LSTMs [14] and Transformers [39], do not embed BN layers. A recent study uncovered that BN is advantageous in Vision Transformers [45], showing potential room to apply our idea to architectures without BN layers. However, more in-depth studies are necessary to identify the actual applicability of BN (or IABN) to those architectures. While LAME [4] is applicable to models without BN, its limitation is the performance drop in i.i.d. scenarios, as shown in both its paper and our evaluation. While NOTE shows its effectiveness in both non-i.i.d and i.i.d. scenarios, a remaining challenge is to design an algorithm that generalizes to any architecture. We believe the findings and contributions of our work could give valuable insights to future endeavors on this end.

Potential negative societal impacts. As TTA relies on unlabeled test samples and changes the model accordingly, the model is exposed to potential data-driven biases after adaptation, such as fairness issues [3] and adversarial attacks [36]. In some sense, the utility of TTA comes at the expense of exposure to threats. This vulnerability is another crucial problem that both ML researchers and practitioners need to take into consideration. In addition, TTA entails additional computations for adaptation with test data, which may have negative impacts on environments, e.g., increasing electricity consumption and carbon emissions [34]. Nevertheless, we believe NOTE would not exacerbate this issue as it is computationally efficient as mentioned in Section §1.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their constructive feedback and suggestions to improve this paper. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No.NRF-2020R1A2C1004062) and Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) (UD190031RD).

References

[1] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

[2] Elon Bachman and I Capulet. Digital record of tesla crashes resulting in death, May 2022.

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2, 2017.

[4] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2022.

[5] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *ICML Workshop: Multi-Task and Lifelong Reinforcement Learning*, 2019.

[6] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1952–1961. PMLR, 13–18 Jul 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

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

[9] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[10] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency, 2016.

[11] Shirsendu Sukanta Halder, Jean-François Lalonde, and Raoul de Charette. Physics-based rendering for improving robustness to rain, 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[13] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[17] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[18] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[19] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[20] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, 2010.

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, page 896, 2013.

[23] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

[24] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34, 2021.

[25] Aleksej Logacjov, Kerstin Bach, Atle Kongsvold, Hilde Bremseth Bårdstu, and Paul Jarle Mork. Harth: A human activity recognition dataset for machine learning. *Sensors*, 21(23), 2021.

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. 2017.

[27] Massimiliano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, and Barbara Caputo. Kitting in the wild through online domain adaptation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1103–1109. IEEE, 2018.

[28] Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

[29] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.

[30] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[33] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11539–11551. Curran Associates, Inc., 2020.

[34] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, nov 2020.

[35] 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 *International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[37] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[38] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16(4):62–74, 2017.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, mar 1985.

[41] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[42] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[43] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[44] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2022.

[45] Zhuliang Yao, Yue Cao, Yutong Lin, Ze Liu, Zheng Zhang, and Han Hu. Leveraging batch normalization for vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 413–422, 2021.

[46] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section §6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section §6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is available at `https://github.com/TaesikGong/NOTE`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section §4 and Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We ran the entire experiments with three different random seems (0,1,2) and reported the standard deviations.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental details

For the all experiments in the paper, we used three different random seeds (0, 1, 2) and report the average errors (and standard deviations). We ran our experiments on NVIDIA GeForce RTX 3090 GPUs.

A.1 Baseline details

We referred to the official implementations of the baselines. We use the reported best hyperparameters from their paper or code. We further tuned hyperparameters if there exists a hyperparameter selection guideline. Here, we provide additional details of the baseline implementations including hyperparameters.

PL. Following the previous studies [41, 44], we update the BN layers only in PL. We set the learning rate as $LR = 0.001$ as the same as [41].

ONDA. ONDA [27] has two hyperparameters, the update frequency N and the decay of the moving average m. The authors set $N = 10$ and $m = 0.1$ as the default values throughout the experiments, and we follow this choice unless specified.

TENT. TENT [41] set the learning rate as $LR = 0.001$ for all datasets except for ImageNet, and we follow this choice. We referred to the official code[3] for implementing TENT.

LAME. LAME [4] needs an affinity matrix and has hyperparameters related to it. We follow the authors' choice on hyperparameter selection specified in both the paper and in their official code. Namely, we use the kNN affinity matrix and with the value of k set as 5. We referred to the official code[4] for implementing LAME.

CoTTA. CoTTA [44] has three hyperparameters, augmentation confidence threshold p_{th}, restoration factor p, and exponential moving average (EMA) factor m. We follow the authors' choice for restoration factor ($p = 0.01$) and EMA factor ($\alpha = 0.999$). For augmentation confidence threshold, the authors provide a guideline to choose it, which is using 5% quantile for the softmax predictions' confidence on the source domains. We follow this guideline, which results in $p_{th} = 0.92$ for MNIST-C and CIFAR10-C, $p_{th} = 0.72$ for CIFAR100-C, and $p_{th} = 0.55$ for KITTI. For 1D time-series datasets (HARTH and ExtraSensory), augmentations are not provided from the authors, and it is non-trivial to select appropriate augmentations for them. We thus do not use augmentations for these datasets. We referred to the official code[5] for implementing CoTTA.

A.2 Dataset details

A.2.1 Robustness to corruptions

MNIST-C. MNIST-C [28] applies 15 corruptions to the MNIST [21] dataset. Specifically, the corruptions include Shot Noise, Impulse Noise, Glass Blur, Motion Blur, Shear, Scale, Rotate, Brightness, Translate, Stripe, Fog, Spatter, Dotted Line, Zigzag, and Canny Edges, as illustrated in Figure 6. Note that the result of this dataset is included only in the supplementary material. In total, MNIST-C has 60,000 clean training data and 150,000 corrupted test data (10,000 for each corruption type). We use ResNet18 [12] as the backbone network. We train it on the clean training data to generate source models, using stochastic gradient descent with momentum=0.9 and cosine annealing learning rate scheduling [26] for 100 epochs with an initial learning rate of 0.1.

CIFAR10-C/CIFAR100-C. CIFAR10-C/CIFAR100-C [13] are common TTA benchmarks for evaluating the robustness to corruptions [29, 33, 41, 44]. Both CIFAR10/CIFAR100 [19] have 50,000/10,000 training/test data. CIFAR10/CIFAR100 have 10/100 classes respectively. CIFAR10-C/CIFAR100-C apply 15 types of corruptions to CIFAR10/CIFAR100 test data: Gaussian Noise,

[3] https://github.com/DequanWang/tent

[4] https://github.com/fiveai/LAME

[5] https://github.com/qinenergy/cotta

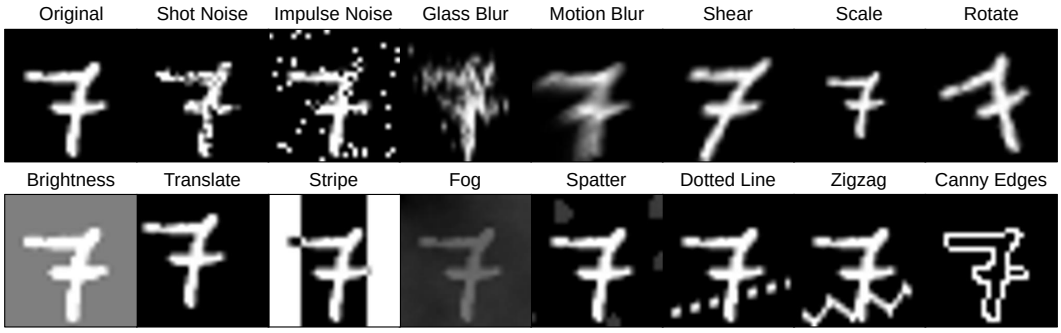


Figure 6: Illustration of the 15 corruption types in the MNIST-C dataset.

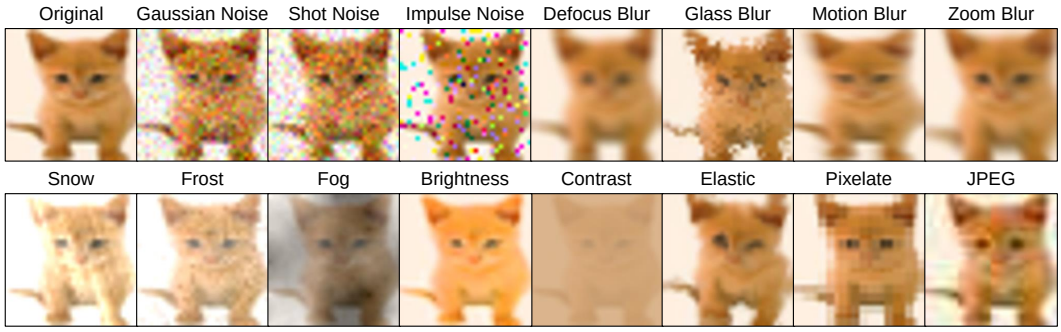
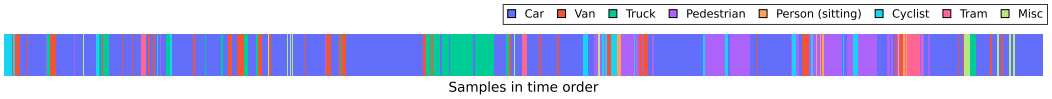


Figure 7: Illustration of the 15 corruption types in the CIFAR10-C/CIFAR100-C/ImageNet-C dataset.

Shot Noise, Impulse Noise, Defocus Blur, Frosted Glass Blur, Motion Blur, Zoom Blur, Snow, Frost, Fog, Brightness, Contrast, Elastic Transformation, Pixelate, and JPEG Compression, as illustrated in Figure 7. We use the most severe corruption level of 5, similar to previous studies [29, 33, 41, 44]. This results in total 150,000 test data for CIFAR10-C/CIFAR100-C, respectively. We use ResNet18 [12] as the backbone network. We train it on the clean training data to generate source models, using stochastic gradient descent with momentum=0.9 and cosine annealing learning rate scheduling [26] for 200 epochs with an initial learning rate of 0.1 and a batch size of 128.

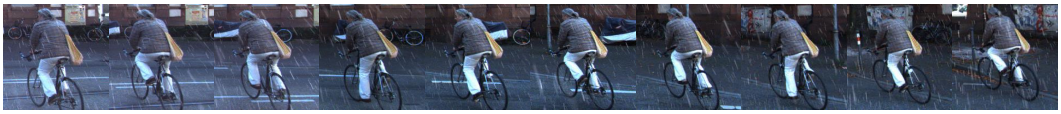
ImageNet-C. ImageNet-C is another common TTA benchmark for evaluating the robustness to corruptions [29, 33, 41, 44, 4]. ImageNet [7] has 1,281,167/50,000 training/test data. ImageNet-C applies the same 15 types of corruptions used in CIFAR10-C and CIFAR100-C. We use a pre-trained ResNet18 [12] on ImageNet training data and fine-tune it by replacing BN layers with IABN layers on the clean ImageNet training data. For fine-tuning, we use stochastic gradient descent with momentum=0.9 for 30 epochs with a fixed learning rate of 0.001 and a batch size of 256.

Temporally correlated streams via Dirichlet distribution. Note that most public vision datasets are not time-series data and existing TTA studies usually shuffled the order of these datasets resulting in i.i.d. streams, which might be unrealistic in real-world scenarios. To simulate non-i.i.d. streams from these "static" datasets, we utilize Dirichlet distribution that is widely used to simulate non-i.i.d. settings. [23, 15, 43, 42] Specifically, we simulate a non-i.i.d partition for T tokens on C classes. For each class c, we draw a T-dimensional vector $\mathbf{q}_c \sim Dir(\delta \mathbf{p})$, where $Dir(\cdot)$ denotes the Dirichlet distribution, $\mathbf{p}$ is a prior class distribution over T classes, and $\delta > 0$ is a concentration parameter. We assign data from each class to each token t, following proportion $\mathbf{q}_c[n]$. To simulate the nature of real-world online data where sequences are temporally correlated and data from same classes appear multiple times (e.g., walking, jogging, and then walking, see Figure 9 and 10 for illustrations), we concatenate the generated T tokens to create a synthetic non-i.i.d. sequential data. We use $\delta = 0.1$ as the default value if not specified.



(a) Visualization of the class distribution in the entire KITTI dataset.

(b) Original data with an interval of three frames.



(c) Rain data with an interval of three frames.

Figure 8: Illustration of the test stream of the KITTI dataset. We apply a 200mm/hr intensity of rain to the original data.

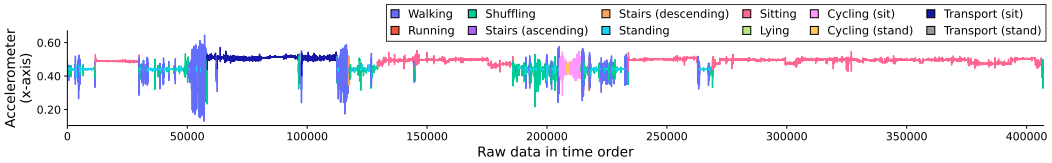
A.2.2 Real-distributions with domain shift

The following illustrates the summary and preprocessing steps of datasets collected in the real-world, or has resemblance to class distributions in the real-world.

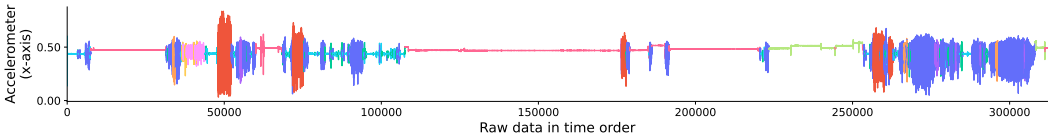
KITTI, KITTI-Rain. KITTI [9] is a well-known dataset used in numerous tasks such as object detection, object tracking, depth estimation, etc. It must be emphasized that the dataset was collected by driving around the city, in rural areas and on highways, which captures the real-world distribution. From the available tasks, we select object tracking task; to utilize its temporal correlation. In order to reduce the task to a single image classification task, we crop each frame with respect to the largest bounding box. Domain gap is introduced, through synthetic generation of corresponding "rainy" frames, hereby denoted as KITTI-Rain [11]. KITTI-Rain is generated via a two-step procedure: (1) generation of a depth-map estimation of each frame, and (2) generation of rainy images from vanilla frame and its corresponding depth map, as described in [11]. For the depth map generation, we used Monodepth [10], and for rainy image generation we use the source code available in [11]. The rain intensity is set to 200mm/hr for training and testing. The final source domain consists of 7,481 samples, and each of the target domain consists of 7,800 samples. We use ResNet50 [12] pre-trained on ImageNet [8] as the backbone network. We fine-tune it on the KITTI training data to generate source models, using the Adam optimizer [18] and cosine annealing learning rate scheduling [26] for 50 epochs with an initial learning rate of 0.1 and a batch size of 64.

HARTH. Human Activity Recognition Trondheim dataset [25] was collected from 22 users, with two three-axial Axivity AX3 accelerometers, each attached to the subject's thigh and lower back. HARTH was also collected in a free-living environment, labelled through recorded video. We set the source domain as the accelerometer data collected from the back (15 users), and set the target domain as one collected from the thigh (from the remaining seven users). We deem such setting to be natural, for one of the most dominant forms of domain shift in wearable sensory data is by the positioning of sensors on the human body [20]. We use a window size of 50, and min-max scaled (0-1) the data, following the original paper [25]. The final source domain consists of 82,544 samples, and each of the seven target domain consists of {S008: 8,140, S018: 6,241, S019: 5,846, S021: 5,910, S022: 6,448, S028: 3,271, S029: 3,521} samples. We use four one-dimensional convolutional layers followed by one fully-connected layers as the backbone network. We train it on the source data to generate source models, using stochastic gradient descent with momentum=0.9 for 100 epochs and cosine annealing learning rate scheduling [26] with an initial learning rate of 0.1 and a batch size of 64.

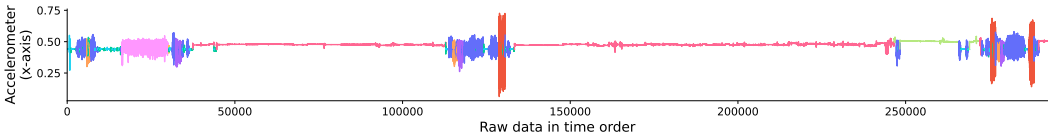
ExtraSensory. Extrasensory dataset [38] was collected from 60 users with the user's own smartphones, over a seven day period, in-the-wild, i.e. data was collected from users whom engaged in their regular natural behavior. As there were no constraints on the subject's activity, the distribution varied



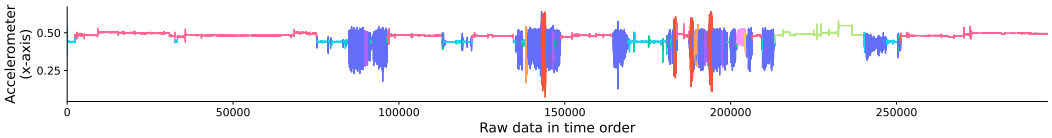
(a) S008.



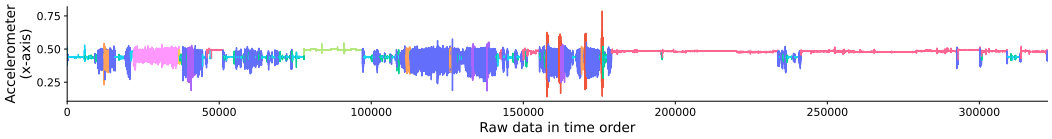
(b) S018.



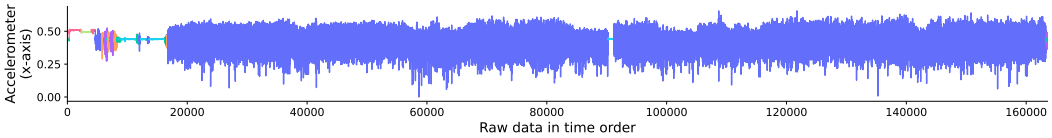
(c) S019.



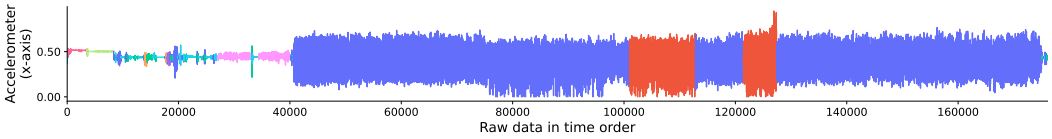
(d) S021.



(e) S022.



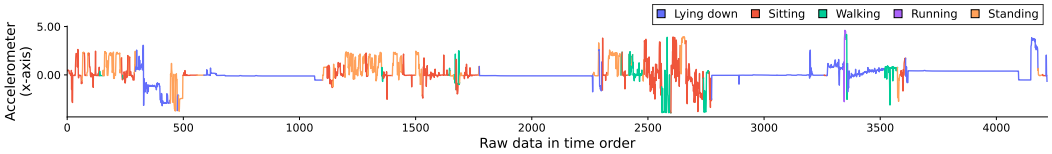
(f) S028.



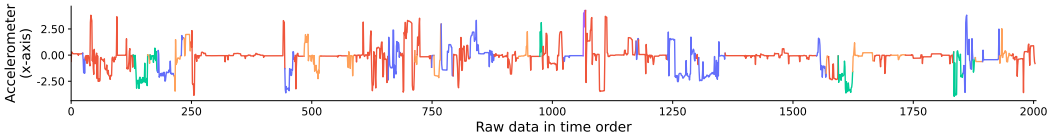
(g) S029.

Figure 9: Illustration of the target streams of the HARTH dataset. We specify x-axis accelerometer values only.

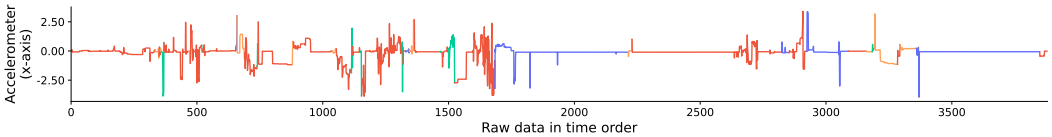
from user to user. We select five most frequently occurred, mutually exclusive activities (lying down, sitting, walking, standing, running), and omit other labels. We further process the data to only those consisting the following sensor modalities - accelerometer, gyroscope, magnetometer and audio. We used a window size of five, with no overlap, and standardly scale the datasets. After the pre-processing step, 23 users were left, and 16 of them were used as source domains and seven of them are used as target domains. The final source domain consists of 17,777 samples, and each of the seven target domain consists of {4FC32141-E888-4BFF-8804-12559A491D8C: 844, 59818CD2-24D7-4D32-B133-24C2FE3801E5: 401, 61976C24-1C50-4355-9C49-AAE44A7D09F6: 776, 797D145F-3858-4A7F-A7C2-A4EB721E133C: 463, A5CDF89D-02A2-4EC1-89F8-F534FDABDD96 : 734, C48CE857-



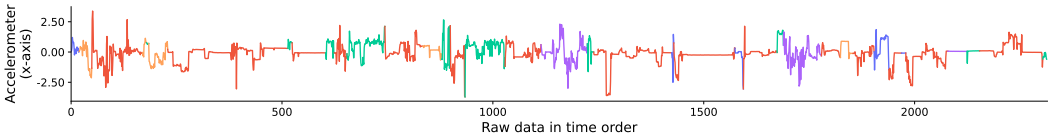
(a) 4FC.



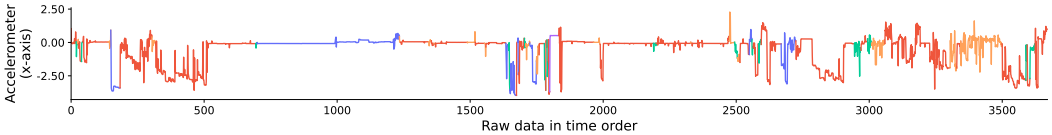
(b) 598.



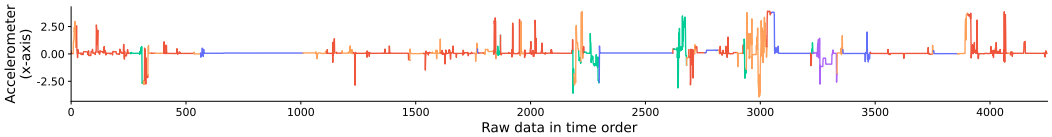
(c) 619.



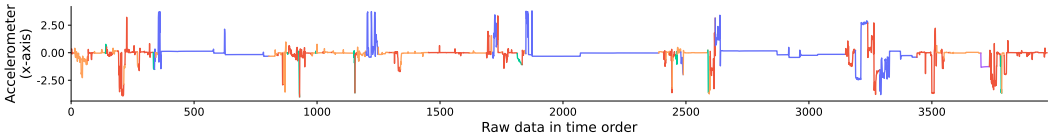
(d) 797.



(e) A5D.



(f) C48.

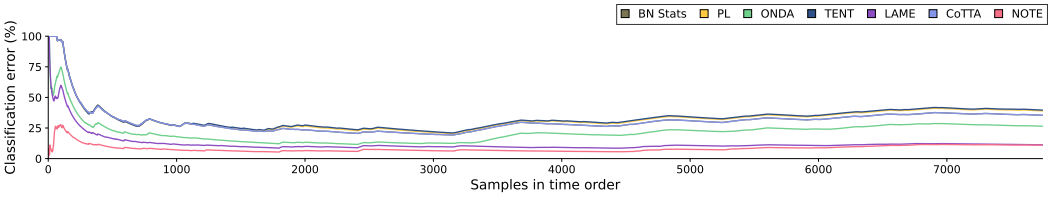


(g) D7D.

Figure 10: Illustration of the target streams of the Extrasensory dataset. We specify x-axis accelerometer values only. Due to the length of the name of each domain, denoted here with the first three characters.

A0DD-4DDB-BEA5-3A25449B2153 : 850, D7D20E2E-FC78-405D-B346-DBD3FD8FC92B: 794} samples. We use two one-dimensional convolutional layers followed by one fully-connected layers as the backbone network. We train it on the source data to generate source models, using stochastic gradient descent with momentum=0.9 for 100 epochs and cosine annealing learning rate scheduling [26] with an initial learning rate of 0.1 and a batch size of 64.

Error on the source domain. We also measure the domain gap between the source and the targets in the three real-distribution datasets: Table 4 for KITTI, Table 5 for HARTH, and Table 6 for



(a) Rain-200.

Figure 11: Illustration of the real-time cumulative classification error change of different methods on the KITTI dataset. The x-axis denotes the samples in order, whereas the y-axis denotes the error rate in percentage. Note that some lines are not clearly visible due to overlap.

Extrasensory. As shown, there is a clear performance degradation from the source domain to the target domain. For HARTH and ExtraSensory, the performance degradation was severe (30∼40%p increased error rates compared with Source), indicating the importance overcoming the domain shift problem in sensory applications.

Table 4: Average classification error (%) and their corresponding standard deviations on the KITTI dataset of the source model. Bold type indicates those of lowest classification error. Averaged over 3 runs.

Method	Src domain	Rain	Avg
Source	**7.4 ± 1.0**	12.3 ± 2.3	9.9

Table 5: Average classification error (%) and their corresponding standard deviations on the HARTH dataset of the source model. Bold type indicates those of lowest classification error. Averaged over 3 runs.

Method	Src domain	S008	S018	S019	S021	S022	S028	S029	Avg
Source	**11.7 ± 0.7**	86.2 ± 1.3	44.7 ± 2.1	50.4 ± 9.5	74.8 ± 3.8	72.0 ± 2.6	53.0 ± 24.0	57.0 ± 16.7	56.2

Table 6: Average classification error (%) and their corresponding standard deviations on the ExtraSensory dataset of the source model. Bold type indicates those of lowest classification error. Averaged over 3 runs.

Method	Src domain	4FC	598	619	797	A5C	C48	D7D	Avg
Source	**8.3 ± 0.7**	34.6 ± 2.5	40.1 ± 0.7	63.8 ± 5.7	45.3 ± 2.4	64.6 ± 3.7	39.6 ± 6.8	63.0 ± 3.9	44.9

B Domain-wise results

B.1 Robustness to corruptions

Table 7: Average classification error (%) and their corresponding standard deviations on CIFAR10-C with **temporally correlated test streams**, shown per corruption. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	74.0 ± 3.3	66.8 ± 3.5	75.3 ± 4.2	43.3 ± 2.7	48.0 ± 2.7	32.6 ± 1.2	35.2 ± 2.6	22.0 ± 0.4	33.0 ± 2.5	25.9 ± 0.9	8.5 ± 0.3	66.1 ± 1.8	23.4 ± 0.7	53.6 ± 0.7	26.8 ± 0.7	42.3
BN Stats [29]	77.2 ± 0.7	76.7 ± 1.0	78.9 ± 0.8	70.0 ± 1.7	78.6 ± 0.6	70.5 ± 1.5	71.1 ± 1.4	72.5 ± 1.4	71.9 ± 1.1	70.6 ± 1.6	68.7 ± 1.9	69.1 ± 1.9	75.1 ± 1.5	73.6 ± 1.4	76.8 ± 1.4	73.4
ONDA [27]	69.3 ± 1.0	68.5 ± 1.0	71.8 ± 0.6	58.5 ± 1.4	71.0 ± 0.2	59.9 ± 1.0	59.5 ± 1.0	62.4 ± 1.4	62.1 ± 1.0	59.6 ± 1.3	55.6 ± 1.4	58.4 ± 1.4	65.6 ± 1.0	63.9 ± 1.4	67.6 ± 1.1	63.6
PL [22]	78.3 ± 1.0	78.0 ± 1.5	80.4 ± 1.0	72.2 ± 1.6	80.1 ± 1.2	72.4 ± 2.2	73.1 ± 1.4	74.5 ± 2.5	73.9 ± 1.8	73.4 ± 1.7	71.5 ± 2.7	71.7 ± 2.5	77.3 ± 2.1	75.7 ± 1.5	78.6 ± 2.7	75.4
TENT [41]	79.0 ± 2.9	78.8 ± 2.8	80.6 ± 2.2	73.3 ± 1.7	80.5 ± 2.9	74.4 ± 2.4	74.5 ± 3.3	74.8 ± 2.2	75.0 ± 2.3	74.0 ± 2.2	72.3 ± 3.4	74.9 ± 3.2	78.2 ± 2.8	76.5 ± 2.9	79.0 ± 2.9	76.4
LAME [4]	73.6 ± 5.2	64.8 ± 4.6	74.8 ± 6.4	36.2 ± 4.4	37.7 ± 5.3	24.9 ± 1.6	27.9 ± 3.4	**12.4 ± 1.0**	22.4 ± 3.9	19.4 ± 0.9	**3.6 ± 0.3**	65.1 ± 1.5	**12.6 ± 0.8**	50.3 ± 0.9	**16.4 ± 1.2**	36.2
CoTTA [44]	77.0 ± 0.7	76.8 ± 0.6	79.0 ± 0.7	74.1 ± 0.9	79.6 ± 0.6	74.3 ± 0.5	74.0 ± 0.8	74.8 ± 1.1	73.3 ± 0.9	72.9 ± 0.5	72.2 ± 0.9	76.5 ± 0.8	76.5 ± 0.9	75.1 ± 0.8	76.6 ± 0.6	75.5
NOTE	**34.9 ± 1.6**	**32.3 ± 3.1**	**39.6 ± 2.5**	**13.6 ± 0.5**	**35.8 ± 1.9**	**11.8 ± 0.8**	**14.5 ± 0.5**	14.1 ± 0.6	**15.2 ± 1.3**	**14.2 ± 0.6**	7.7 ± 0.3	**7.6 ± 0.6**	20.8 ± 0.7	**27.7 ± 2.6**	26.4 ± 0.5	**21.1**

Table 8: Average classification error (%) and their corresponding standard deviations on CIFAR100-C with **temporally correlated test streams**, shown per corruption. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	88.1 ± 0.2	86.8 ± 0.6	93.7 ± 0.6	64.9 ± 0.4	79.7 ± 0.9	55.5 ± 0.3	57.7 ± 0.2	53.8 ± 0.4	66.3 ± 0.8	59.3 ± 0.4	33.0 ± 0.3	81.4 ± 0.4	49.2 ± 0.4	73.6 ± 1.1	55.5 ± 0.3	66.6
BN Stats [29]	73.9 ± 0.5	73.5 ± 0.4	77.2 ± 0.7	56.9 ± 0.2	72.3 ± 0.5	58.8 ± 0.3	57.9 ± 0.4	65.3 ± 0.4	65.0 ± 0.4	62.4 ± 0.6	55.6 ± 0.2	57.6 ± 0.4	64.6 ± 0.5	63.6 ± 0.3	71.0 ± 0.4	65.0
ONDA [27]	**63.0 ± 0.7**	**62.5 ± 0.4**	**68.0 ± 0.5**	37.3 ± 0.2	**60.0 ± 0.2**	40.0 ± 0.3	38.3 ± 0.1	49.6 ± 0.3	50.0 ± 0.6	45.2 ± 0.6	35.7 ± 0.2	40.9 ± 0.5	48.6 ± 0.5	**46.9 ± 0.3**	57.5 ± 0.2	49.6
PL [22]	71.9 ± 1.4	72.0 ± 0.5	76.3 ± 0.7	59.3 ± 0.8	73.8 ± 0.9	61.5 ± 0.9	59.9 ± 0.5	67.1 ± 0.9	66.7 ± 1.4	63.0 ± 1.0	57.9 ± 0.5	62.2 ± 1.5	67.6 ± 1.0	65.2 ± 0.3	71.1 ± 0.5	66.4
TENT [41]	71.8 ± 0.9	71.0 ± 0.4	76.4 ± 1.2	60.2 ± 0.6	75.0 ± 1.0	61.9 ± 0.9	60.2 ± 0.7	67.8 ± 0.5	67.8 ± 0.7	63.3 ± 1.1	58.4 ± 0.7	65.0 ± 1.8	68.4 ± 0.9	65.0 ± 0.2	71.8 ± 0.1	66.9
LAME [4]	89.0 ± 1.1	87.1 ± 0.8	94.5 ± 0.7	62.3 ± 1.2	79.7 ± 1.2	49.4 ± 1.0	52.8 ± 0.3	46.6 ± 0.4	63.9 ± 1.9	55.6 ± 1.2	**25.2 ± 0.6**	82.4 ± 0.2	**40.8 ± 0.5**	71.9 ± 1.4	**47.8 ± 0.7**	63.3
CoTTA [44]	68.6 ± 0.3	67.9 ± 0.4	71.4 ± 0.4	60.7 ± 0.4	69.9 ± 0.4	60.8 ± 0.5	60.2 ± 0.2	64.0 ± 0.3	62.9 ± 0.5	63.2 ± 0.6	56.7 ± 0.2	65.6 ± 0.3	64.5 ± 0.3	60.9 ± 0.0	65.3 ± 0.1	64.2
NOTE	66.2 ± 0.8	64.2 ± 1.6	72.6 ± 0.4	**37.2 ± 0.8**	61.1 ± 0.7	**35.4 ± 0.3**	**37.4 ± 0.4**	**40.0 ± 0.4**	**42.5 ± 0.3**	**43.4 ± 0.5**	29.4 ± 0.1	**32.1 ± 0.5**	44.3 ± 0.4	47.5 ± 0.6	51.3 ± 0.3	**47.0**

Table 9: Average classification error (%) and their corresponding standard deviations on ImageNet-C with **temporally correlated test streams**, shown per corruption. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	98.4 ± 0.0	97.7 ± 0.0	98.4 ± 0.0	90.6 ± 0.0	92.5 ± 0.0	89.8 ± 0.0	81.8 ± 0.0	89.5 ± 0.0	85.0 ± 0.0	86.4 ± 0.0	51.1 ± 0.0	97.2 ± 0.0	85.3 ± 0.0	76.9 ± 0.0	71.7 ± 0.0	86.1
BN Stats	98.3 ± 0.0	98.1 ± 0.0	98.4 ± 0.0	98.7 ± 0.0	98.8 ± 0.0	97.8 ± 0.0	96.6 ± 0.0	96.2 ± 0.0	96.0 ± 0.0	95.1 ± 0.0	93.1 ± 0.0	98.6 ± 0.0	96.3 ± 0.0	95.6 ± 0.0	96.1 ± 0.0	96.9
ONDA	95.1 ± 0.0	94.7 ± 0.0	95.0 ± 0.0	96.2 ± 0.0	96.1 ± 0.0	92.5 ± 0.0	87.2 ± 0.0	87.4 ± 0.0	87.8 ± 0.0	82.7 ± 0.0	71.0 ± 0.0	96.4 ± 0.0	84.9 ± 0.0	81.7 ± 0.0	86.1 ± 0.0	89.0
PL	99.3 ± 0.0	99.3 ± 0.0	99.4 ± 0.0	99.5 ± 0.0	99.4 ± 0.0	99.5 ± 0.0	98.8 ± 0.0	99.1 ± 0.0	99.2 ± 0.0	98.1 ± 0.0	97.3 ± 0.1	99.8 ± 0.0	98.4 ± 0.0	98.5 ± 0.0	98.5 ± 0.0	98.9
TENT	98.3 ± 0.0	98.1 ± 0.0	98.4 ± 0.0	98.7 ± 0.0	98.8 ± 0.0	97.8 ± 0.0	96.6 ± 0.0	96.2 ± 0.0	96.0 ± 0.0	95.1 ± 0.0	93.1 ± 0.0	98.6 ± 0.0	96.3 ± 0.0	95.6 ± 0.0	96.1 ± 0.0	96.9
LAME	98.1 ± 0.0	97.1 ± 0.0	98.0 ± 0.0	**87.9 ± 0.0**	**90.9 ± 0.0**	87.1 ± 0.0	**78.3 ± 0.0**	87.1 ± 0.0	80.2 ± 0.0	81.5 ± 0.0	**39.8 ± 0.0**	96.4 ± 0.0	82.5 ± 0.0	70.7 ± 0.0	**64.9 ± 0.0**	82.7
CoTTA	98.2 ± 0.0	98.1 ± 0.0	98.3 ± 0.0	98.8 ± 0.0	98.8 ± 0.0	97.7 ± 0.0	96.8 ± 0.0	96.6 ± 0.1	96.3 ± 0.0	95.3 ± 0.0	93.5 ± 0.0	98.8 ± 0.0	96.5 ± 0.0	95.6 ± 0.0	96.2 ± 0.0	97.0
NOTE	**94.7 ± 0.1**	**93.7 ± 0.3**	**94.5 ± 0.1**	91.2 ± 0.1	91.0 ± 0.2	**83.3 ± 0.1**	79.0 ± 0.2	**79.0 ± 0.4**	**78.7 ± 0.3**	**66.3 ± 0.6**	48.0 ± 0.4	**94.1 ± 0.1**	**76.9 ± 0.6**	**62.6 ± 0.7**	76.6 ± 0.6	**80.6**

Table 10: Average classification error (%) and their corresponding standard deviations on MNIST-C with **temporally correlated test streams**, shown per corruption. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs.

Method	*Shot*	*Impulse*	*Glass*	*Motion*	*Shear*	*Scale*	*Rotate*	*Brightness*	*Translate*	*Stripe*	*Fog*	*Spatter*	*Dotted line*	*Zigzag*	*Canny edges*	Avg
Source	3.7 ± 0.7	27.3 ± 5.5	20.4 ± 6.4	4.6 ± 0.5	2.2 ± 0.5	5.1 ± 1.0	6.5 ± 1.0	21.1 ± 22.9	13.8 ± 1.4	17.4 ± 17.0	66.6 ± 14.7	3.8 ± 0.4	3.7 ± 0.4	18.2 ± 3.0	26.4 ± 11.4	16.1
BN Stats [29]	72.0 ± 0.6	75.2 ± 0.8	73.7 ± 1.0	72.1 ± 0.8	71.2 ± 1.1	71.4 ± 0.6	71.2 ± 0.3	71.6 ± 0.6	78.5 ± 0.2	72.3 ± 1.2	70.8 ± 1.2	71.6 ± 0.9	73.8 ± 0.7	74.6 ± 0.6	72.3 ± 0.3	72.8
ONDA [27]	53.3 ± 3.0	59.9 ± 3.0	59.2 ± 3.3	54.1 ± 3.5	51.6 ± 2.2	53.9 ± 2.5	54.6 ± 2.0	50.5 ± 2.3	65.2 ± 2.1	57.5 ± 0.7	54.8 ± 2.9	54.2 ± 3.0	55.4 ± 2.8	61.0 ± 2.2	56.7 ± 2.1	56.1
PL [22]	73.7 ± 1.0	76.4 ± 0.4	75.3 ± 0.5	74.7 ± 1.1	72.7 ± 0.9	73.3 ± 1.6	73.7 ± 0.9	73.7 ± 1.0	78.7 ± 0.3	74.1 ± 1.4	75.8 ± 2.6	72.5 ± 0.8	75.8 ± 0.6	76.9 ± 1.4	74.5 ± 0.1	74.8
TENT [41]	74.7 ± 1.1	78.1 ± 0.9	76.6 ± 0.6	76.1 ± 0.7	75.8 ± 1.1	73.7 ± 1.3	75.2 ± 1.1	75.4 ± 0.3	78.9 ± 0.2	76.7 ± 1.8	81.4 ± 1.7	73.9 ± 0.5	77.3 ± 0.7	79.2 ± 2.0	75.8 ± 1.0	76.6
LAME [4]	**1.1 ± 0.3**	17.0 ± 8.7	**12.5 ± 6.5**	**1.1 ± 0.3**	**0.4 ± 0.2**	**1.5 ± 0.6**	**2.3 ± 0.6**	17.2 ± 26.0	**6.0 ± 2.3**	**12.3 ± 17.2**	68.3 ± 15.8	**0.7 ± 0.3**	**0.7 ± 0.4**	13.2 ± 3.4	22.1 ± 12.3	11.8
CoTTA [44]	76.9 ± 0.5	79.4 ± 0.4	79.1 ± 0.5	77.6 ± 0.6	75.4 ± 0.4	76.2 ± 1.3	77.6 ± 0.2	76.0 ± 0.5	81.6 ± 0.9	76.8 ± 0.6	78.0 ± 0.4	77.6 ± 0.6	79.3 ± 0.4	80.6 ± 1.0	77.6 ± 0.5	78.0
NOTE	3.9 ± 1.3	**13.8 ± 2.4**	14.3 ± 1.5	3.3 ± 2.4	1.7 ± 0.2	3.8 ± 0.7	6.5 ± 0.3	**0.9 ± 0.0**	8.0 ± 1.2	14.4 ± 8.1	**1.6 ± 0.3**	3.9 ± 0.4	4.5 ± 1.2	**12.6 ± 2.5**	**13.4 ± 3.9**	**7.1**

Table 11: Average classification error (%) and their corresponding standard deviations on CIFAR10-C with **uniformly distributed test streams**, shown per domain. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs. NOTE* indicates NOTE used directly with test batches (without using PBRS)

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	74.0 ± 3.3	66.8 ± 3.5	75.3 ± 4.2	43.3 ± 2.7	48.0 ± 2.7	32.6 ± 1.2	35.2 ± 2.6	22.0 ± 0.4	33.0 ± 2.5	25.9 ± 0.9	8.5 ± 0.3	66.1 ± 1.8	23.4 ± 0.7	53.6 ± 0.7	26.8 ± 0.7	42.3
BN Stats [29]	33.1 ± 0.9	31.1 ± 1.0	39.8 ± 0.9	12.3 ± 0.4	34.8 ± 0.3	13.7 ± 0.3	12.6 ± 0.4	18.3 ± 0.7	19.9 ± 0.6	14.5 ± 0.6	9.3 ± 0.3	13.0 ± 0.3	23.3 ± 0.3	20.8 ± 0.2	28.0 ± 0.6	21.6
ONDA [27]	33.4 ± 0.6	31.3 ± 0.9	40.0 ± 1.1	12.3 ± 0.4	34.6 ± 0.7	13.7 ± 0.3	12.4 ± 0.5	18.3 ± 0.6	19.8 ± 0.8	14.3 ± 0.4	9.1 ± 0.0	14.0 ± 0.2	23.3 ± 0.4	20.9 ± 0.2	28.0 ± 0.7	21.7
PL [22]	29.4 ± 1.1	26.3 ± 1.0	36.8 ± 1.6	13.7 ± 0.4	36.5 ± 1.1	14.0 ± 1.0	13.5 ± 0.2	19.7 ± 0.8	21.2 ± 0.6	15.6 ± 1.5	10.0 ± 0.6	14.8 ± 0.2	24.5 ± 2.0	20.1 ± 0.9	27.4 ± 1.3	21.6
TENT [41]	25.3 ± 0.8	23.1 ± 1.1	32.1 ± 1.2	**11.7 ± 0.6**	33.1 ± 3.0	13.2 ± 1.1	**11.2 ± 0.1**	15.9 ± 0.3	18.8 ± 0.7	**12.9 ± 0.8**	8.6 ± 0.3	14.4 ± 0.6	21.7 ± 0.9	**16.5 ± 0.8**	23.6 ± 0.7	18.8
LAME [4]	78.2 ± 3.6	70.6 ± 4.0	80.5 ± 4.5	46.6 ± 1.9	48.0 ± 3.8	34.2 ± 0.4	37.4 ± 1.5	20.8 ± 0.8	30.5 ± 4.1	26.9 ± 1.8	9.8 ± 0.2	71.9 ± 1.0	24.2 ± 0.9	56.4 ± 0.8	25.8 ± 0.9	44.1
CoTTA [44]	**23.1 ± 0.7**	**21.5 ± 0.6**	**28.0 ± 0.3**	**11.7 ± 0.5**	**29.2 ± 0.6**	13.3 ± 0.6	12.0 ± 0.5	16.6 ± 0.2	16.6 ± 0.3	13.8 ± 0.4	8.8 ± 0.2	14.9 ± 0.5	**20.6 ± 0.7**	17.3 ± 0.5	**19.9 ± 0.4**	17.8
NOTE	33.5 ± 1.7	30.0 ± 1.6	38.2 ± 0.9	12.6 ± 0.8	34.4 ± 0.8	**11.5 ± 0.5**	12.9 ± 0.6	**14.1 ± 0.2**	15.2 ± 0.8	14.0 ± 0.6	**7.4 ± 0.2**	7.8 ± 0.2	20.7 ± 0.3	24.7 ± 0.7	24.2 ± 0.4	20.1
NOTE*	23.8 ± 0.7	23.0 ± 0.9	31.1 ± 0.3	11.8 ± 0.6	30.9 ± 1.3	11.8 ± 0.4	11.9 ± 0.7	15.3 ± 1.3	**14.0 ± 0.7**	13.3 ± 0.7	8.6 ± 0.2	**7.5 ± 0.3**	21.2 ± 0.3	16.9 ± 0.6	23.0 ± 1.2	**17.6**

Table 12: Average classification error (%) and their corresponding standard deviations on CIFAR100-C with **uniformly distributed test streams**, shown per domain. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs. NOTE* indicates NOTE used directly with test batches (without using PBRS)

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	88.1 ± 0.2	86.8 ± 0.6	93.7 ± 0.6	64.9 ± 0.4	79.7 ± 0.9	55.5 ± 0.3	57.7 ± 0.2	53.8 ± 0.4	66.3 ± 0.8	59.3 ± 0.4	33.0 ± 0.3	81.4 ± 0.4	49.2 ± 0.4	73.6 ± 1.1	55.5 ± 0.3	66.6
BN Stats [29]	60.9 ± 0.8	59.9 ± 0.6	65.7 ± 0.8	33.7 ± 0.4	57.6 ± 0.4	36.5 ± 0.2	35.2 ± 0.4	46.7 ± 0.3	46.9 ± 0.4	42.8 ± 0.7	32.3 ± 0.4	35.6 ± 0.5	45.8 ± 0.3	43.6 ± 0.3	55.5 ± 0.2	46.6
ONDA [27]	60.8 ± 0.9	60.2 ± 0.5	66.0 ± 0.6	33.9 ± 0.4	57.5 ± 0.4	36.3 ± 0.4	34.6 ± 0.4	46.5 ± 0.3	47.2 ± 0.3	42.1 ± 0.6	32.1 ± 0.5	36.4 ± 0.4	45.5 ± 0.1	43.4 ± 0.8	55.1 ± 0.1	46.5
PL [22]	52.2 ± 0.9	50.3 ± 1.0	59.4 ± 0.9	33.5 ± 0.5	54.0 ± 0.6	35.7 ± 0.3	33.1 ± 0.5	42.8 ± 0.9	44.5 ± 1.6	39.2 ± 1.3	30.9 ± 0.2	35.5 ± 0.2	45.5 ± 1.0	39.9 ± 0.3	50.4 ± 1.3	43.1
TENT [41]	**48.7 ± 0.8**	**47.2 ± 0.6**	**55.6 ± 0.9**	**31.5 ± 0.2**	**50.9 ± 0.5**	33.5 ± 0.4	**31.7 ± 0.2**	39.6 ± 0.3	41.0 ± 0.1	36.8 ± 0.7	29.4 ± 0.3	33.6 ± 0.4	**42.3 ± 0.6**	**36.8 ± 0.5**	**46.4 ± 0.5**	**40.3**
LAME [4]	91.0 ± 1.0	89.5 ± 1.0	95.2 ± 0.7	68.1 ± 0.9	82.7 ± 1.1	57.1 ± 0.5	60.2 ± 0.3	54.7 ± 0.3	68.9 ± 1.2	61.8 ± 0.6	33.7 ± 0.5	85.2 ± 0.4	50.3 ± 0.2	76.7 ± 1.3	56.2 ± 0.5	68.8
CoTTA [44]	52.8 ± 0.7	51.0 ± 0.4	56.9 ± 0.6	35.8 ± 0.4	53.9 ± 0.2	37.9 ± 0.5	36.8 ± 0.1	45.2 ± 0.5	44.5 ± 0.1	44.0 ± 0.2	32.2 ± 0.5	41.3 ± 1.4	46.1 ± 0.1	39.7 ± 0.3	46.9 ± 0.7	44.3
NOTE	65.6 ± 1.0	62.6 ± 0.7	72.0 ± 0.2	36.8 ± 0.7	60.5 ± 0.7	34.9 ± 0.5	36.7 ± 0.2	39.6 ± 0.2	41.7 ± 0.6	42.3 ± 0.3	**28.6 ± 0.2**	32.3 ± 0.9	43.8 ± 0.2	47.7 ± 0.4	50.9 ± 0.2	46.4
NOTE*	51.8 ± 1.0	50.0 ± 0.3	60.7 ± 0.4	32.6 ± 0.2	54.4 ± 0.3	**33.0 ± 0.2**	33.5 ± 0.4	**38.5 ± 0.3**	**38.6 ± 0.1**	**36.7 ± 0.3**	29.7 ± 0.5	**27.3 ± 0.3**	43.2 ± 0.4	37.1 ± 0.2	47.6 ± 0.9	41.0

Table 13: Average classification error (%) and their corresponding standard deviations on ImageNet-C with **temporally correlated test streams**, shown per corruption. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	98.4 ± 0.0	97.7 ± 0.0	98.4 ± 0.0	90.6 ± 0.0	92.5 ± 0.0	89.8 ± 0.0	81.8 ± 0.0	89.5 ± 0.0	85.0 ± 0.0	86.4 ± 0.0	51.1 ± 0.0	97.2 ± 0.0	85.3 ± 0.0	76.9 ± 0.0	71.7 ± 0.0	86.1
BN Stats	89.4 ± 0.0	88.5 ± 0.1	89.2 ± 0.2	90.8 ± 0.0	90.0 ± 0.0	81.3 ± 0.0	69.8 ± 0.2	72.6 ± 0.1	73.8 ± 0.0	62.6 ± 0.0	44.3 ± 0.3	92.1 ± 0.0	64.5 ± 0.1	60.3 ± 0.1	70.7 ± 0.0	76.0
ONDA	89.2 ± 0.0	88.2 ± 0.0	89.0 ± 0.1	90.9 ± 0.1	90.0 ± 0.1	81.6 ± 0.1	69.5 ± 0.0	72.6 ± 0.1	73.7 ± 0.0	62.7 ± 0.1	43.9 ± 0.0	92.1 ± 0.0	64.3 ± 0.0	60.1 ± 0.1	70.0 ± 0.0	75.9
PL	89.8 ± 1.9	86.1 ± 0.9	88.5 ± 1.6	93.0 ± 1.1	92.5 ± 0.6	82.2 ± 0.0	64.6 ± 0.3	70.2 ± 0.6	79.7 ± 0.4	**55.8 ± 0.2**	43.9 ± 0.1	97.2 ± 0.5	**57.8 ± 0.1**	**52.7 ± 0.2**	**60.5 ± 0.1**	74.4
TENT	91.1 ± 2.4	89.7 ± 1.6	91.0 ± 2.5	93.1 ± 3.2	92.2 ± 3.2	84.7 ± 4.9	72.4 ± 3.5	73.3 ± 1.1	78.7 ± 6.9	59.8 ± 4.0	44.5 ± 0.5	95.2 ± 4.3	61.6 ± 4.3	56.4 ± 5.6	67.4 ± 4.7	76.5
LAME	98.6 ± 0.0	97.8 ± 0.0	98.6 ± 0.0	90.7 ± 0.0	92.6 ± 0.0	89.9 ± 0.0	81.9 ± 0.0	89.8 ± 0.0	85.0 ± 0.0	86.5 ± 0.0	51.1 ± 0.0	97.3 ± 0.0	85.6 ± 0.0	77.0 ± 0.0	71.7 ± 0.0	86.3
CoTTA	**85.7 ± 0.2**	**84.6 ± 0.1**	**85.4 ± 0.0**	87.8 ± 0.3	86.4 ± 0.2	74.6 ± 0.0	**64.2 ± 0.2**	67.9 ± 0.0	69.7 ± 0.2	56.1 ± 0.1	**42.7 ± 0.0**	88.5 ± 0.8	60.0 ± 0.0	54.2 ± 0.1	64.9 ± 0.1	71.5
NOTE	87.6 ± 0.1	85.7 ± 0.1	87.2 ± 0.2	**83.3 ± 0.2**	**83.2 ± 0.2**	**73.6 ± 0.0**	65.4 ± 0.2	**65.0 ± 0.0**	**68.6 ± 0.1**	57.9 ± 0.0	43.5 ± 0.1	**75.9 ± 0.1**	61.2 ± 0.1	54.1 ± 0.0	62.8 ± 0.1	**70.3**
NOTE*	89.5 ± 0.4	87.9 ± 0.2	88.9 ± 0.3	84.6 ± 0.2	83.7 ± 0.2	74.4 ± 0.1	66.6 ± 0.1	66.1 ± 0.2	71.2 ± 0.1	58.2 ± 0.1	44.7 ± 0.1	78.8 ± 0.1	61.2 ± 0.2	54.8 ± 0.0	64.8 ± 0.1	71.7

Table 14: Average classification error (%) and their corresponding standard deviations on MNIST-C with **uniformly distributed test streams**, shown per domain. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs. NOTE* indicates NOTE used directly with test batches (without using PBRS)

Method	*Shot*	*Impulse*	*Glass*	*Motion*	*Shear*	*Scale*	*Rotate*	*Brightness*	*Translate*	*Stripe*	*Fog*	*Spatter*	*Dotted line*	*Zigzag*	*Canny edges*	Avg
Source	3.7 ± 0.7	27.3 ± 5.5	20.4 ± 6.4	4.6 ± 0.5	2.2 ± 0.5	5.1 ± 1.0	6.5 ± 1.0	21.1 ± 22.9	13.8 ± 1.4	17.4 ± 17.0	66.6 ± 14.7	3.8 ± 0.4	3.7 ± 0.4	18.2 ± 3.0	26.4 ± 11.4	16.1
BN Stats [29]	2.9 ± 0.7	7.0 ± 1.6	9.1 ± 1.0	3.0 ± 0.8	2.0 ± 0.3	3.8 ± 0.2	6.1 ± 0.7	1.1 ± 0.1	12.5 ± 0.8	6.5 ± 2.6	2.2 ± 0.5	3.3 ± 0.3	2.5 ± 0.2	11.4 ± 0.2	6.7 ± 0.9	5.3
ONDA [27]	2.6 ± 0.6	6.5 ± 1.4	8.6 ± 1.0	2.8 ± 0.8	1.8 ± 0.2	3.5 ± 0.2	5.7 ± 0.7	1.0 ± 0.1	11.7 ± 1.1	6.1 ± 2.6	2.6 ± 0.9	3.0 ± 0.4	2.2 ± 0.2	11.0 ± 0.4	6.2 ± 0.8	5.0
PL [22]	1.6 ± 0.3	3.5 ± 0.7	4.8 ± 0.8	1.7 ± 0.0	1.5 ± 0.0	2.3 ± 0.1	4.9 ± 0.7	0.8 ± 0.1	6.8 ± 0.8	2.7 ± 0.6	1.0 ± 0.0	2.2 ± 0.3	1.7 ± 0.2	5.3 ± 0.4	3.9 ± 0.9	3.0
TENT [41]	1.4 ± 0.1	2.8 ± 0.4	**3.8 ± 0.5**	1.5 ± 0.0	1.2 ± 0.0	1.8 ± 0.1	3.6 ± 0.2	**0.7 ± 0.1**	4.6 ± 0.7	**1.9 ± 0.2**	0.8 ± 0.0	**1.7 ± 0.1**	**1.3 ± 0.1**	**4.5 ± 0.6**	**3.1 ± 0.5**	2.3
LAME [4]	3.0 ± 0.8	30.7 ± 8.3	18.9 ± 5.8	3.4 ± 0.5	1.9 ± 0.3	4.2 ± 0.5	6.3 ± 0.9	25.9 ± 29.8	13.9 ± 1.9	18.5 ± 21.2	78.2 ± 9.8	3.3 ± 0.7	3.2 ± 0.3	19.3 ± 3.2	28.0 ± 12.7	17.2
CoTTA [44]	2.6 ± 0.6	6.6 ± 1.7	8.7 ± 0.9	2.7 ± 0.7	1.8 ± 0.3	3.2 ± 0.0	5.6 ± 0.8	1.0 ± 0.1	14.3 ± 1.1	7.7 ± 6.0	1.9 ± 0.5	2.9 ± 0.3	2.2 ± 0.1	13.6 ± 1.4	6.1 ± 0.6	5.4
NOTE	2.5 ± 0.8	10.7 ± 1.9	10.9 ± 2.0	2.0 ± 0.3	1.5 ± 0.0	2.4 ± 0.1	5.5 ± 0.3	0.9 ± 0.1	5.5 ± 0.2	12.1 ± 5.7	1.2 ± 0.1	2.8 ± 0.3	3.0 ± 0.1	10.9 ± 1.6	9.1 ± 0.4	5.4
NOTE*	**1.3 ± 0.2**	**2.7 ± 0.1**	**3.8 ± 0.5**	**1.3 ± 0.1**	**1.1 ± 0.1**	**1.6 ± 0.0**	**3.5 ± 0.1**	**0.7 ± 0.0**	**2.8 ± 0.0**	2.2 ± 0.1	**0.7 ± 0.1**	**1.7 ± 0.4**	1.4 ± 0.2	4.8 ± 1.1	3.5 ± 0.1	**2.2**

B.2 Real distributions with domain shift

Since the adaptation is done from a single source domain to a single target domain in KITTI, no further per-domain tables are specified here.

Table 15: Average classification error (%) and their corresponding standard deviations on HARTH with **real test streams**, shown per domain. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Averaged over 3 runs.

Method	S008	S018	S019	S021	S022	S028	S029	Avg
Source	86.2 ± 1.3	44.7 ± 2.1	50.4 ± 9.5	74.8 ± 3.8	72.0 ± 2.6	53.0 ± 24.0	57.0 ± 16.7	62.6
BN Stats [29]	70.3 ± 1.4	73.8 ± 1.3	68.1 ± 3.0	64.9 ± 0.9	68.5 ± 0.3	65.5 ± 0.5	69.4 ± 1.4	68.6
ONDA [27]	75.3 ± 4.0	60.4 ± 0.9	63.1 ± 4.6	67.9 ± 0.4	70.0 ± 3.8	73.6 ± 0.7	74.5 ± 4.4	69.3
PL [22]	60.4 ± 1.3	71.4 ± 1.5	62.9 ± 1.9	61.8 ± 1.2	63.1 ± 0.4	64.5 ± 0.8	69.4 ± 2.0	64.8
TENT [41]	**59.5 ± 0.3**	71.0 ± 1.6	62.2 ± 1.9	**61.1 ± 1.1**	**61.7 ± 0.4**	64.1 ± 0.5	69.3 ± 2.1	64.1
LAME [4]	85.5 ± 1.7	43.4 ± 2.0	48.8 ± 10.9	73.2 ± 3.8	70.7 ± 2.6	51.2 ± 29.4	54.1 ± 20.6	61.0
CoTTA [44]	70.4 ± 1.4	73.8 ± 1.3	68.2 ± 2.9	64.9 ± 1.0	68.5 ± 0.2	65.5 ± 0.5	69.4 ± 1.4	68.7
NOTE	84.8 ± 0.7	**32.9 ± 1.8**	**36.3 ± 10.9**	69.1 ± 2.4	67.1 ± 1.2	**30.0 ± 13.8**	**36.6 ± 9.8**	**51.0**

Table 16: Average classification error (%) and their corresponding standard deviations on Extrasensory with **real test streams**, shown per domain. **Bold** type indicates those of lowest classification error, while those with Red font are performance degradation after adaptation. Due to the length of the name of each domain, denoted here with the first three characters. Averaged over 3 runs.

Method	4FC	598	619	797	A5D	C48	D7D	Avg
Source	34.6 ± 2.5	40.1 ± 0.7	63.8 ± 5.7	45.3 ± 2.4	64.6 ± 3.7	39.6 ± 6.8	63.0 ± 3.9	50.2
BN Stats[29]	61.7 ± 4.2	50.1 ± 5.1	51.6 ± 1.5	59.4 ± 1.1	54.4 ± 1.0	52.4 ± 2.8	62.6 ± 2.9	56.0
ONDA [27]	36.3 ± 3.5	44.0 ± 2.2	**50.8 ± 2.4**	56.1 ± 1.9	59.7 ± 2.7	43.5 ± 5.9	**46.7 ± 4.2**	48.2
PL [22]	62.2 ± 4.3	50.0 ± 5.1	51.7 ± 1.8	59.2 ± 1.1	53.9 ± 1.1	52.3 ± 2.9	62.8 ± 3.0	56.0
TENT [41]	62.1 ± 4.6	49.8 ± 5.0	51.6 ± 1.9	59.4 ± 1.2	53.9 ± 1.0	52.2 ± 2.9	62.8 ± 3.0	56.0
LAME [4]	**33.1 ± 2.4**	**37.8 ± 0.4**	68.0 ± 8.8	**37.1 ± 6.7**	73.2 ± 2.6	39.0 ± 7.6	66.4 ± 4.0	50.7
CoTTA [44]	61.7 ± 4.2	50.0 ± 4.9	51.6 ± 1.5	59.4 ± 1.1	54.4 ± 1.0	52.4 ± 2.8	62.6 ± 2.9	56.0
NOTE	41.7 ± 5.9	40.7 ± 0.8	55.5 ± 10.8	45.8 ± 4.6	**45.8 ± 10.4**	**32.9 ± 1.1**	55.5 ± 10.4	**45.4**

B.3 Ablation study

Table 17: Average classification error (%) and their corresponding standard deviations of varying ablation settings on CIFAR10-C with **temporally correlated test streams**, shown per domain. **Bold** type indicates those of lowest classification error. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	74.0 ± 3.3	66.8 ± 3.5	75.3 ± 4.2	43.3 ± 2.7	48.0 ± 2.7	32.6 ± 1.2	35.2 ± 2.6	22.0 ± 0.4	33.0 ± 2.5	25.9 ± 0.9	8.5 ± 0.3	66.1 ± 1.8	23.4 ± 0.7	53.6 ± 0.7	26.8 ± 0.7	42.3
IABN	44.5 ± 2.7	41.3 ± 2.3	48.0 ± 1.9	16.3 ± 1.0	39.9 ± 0.1	13.8 ± 0.7	16.1 ± 0.7	14.9 ± 0.3	17.8 ± 0.6	16.3 ± 0.6	7.6 ± 0.2	8.8 ± 0.3	22.5 ± 0.3	34.0 ± 1.2	26.7 ± 0.6	24.6
PBRS	45.2 ± 3.0	38.5 ± 4.9	46.8 ± 3.3	24.5 ± 2.2	38.2 ± 2.8	19.1 ± 0.9	20.0 ± 0.2	16.5 ± 0.2	19.1 ± 2.4	16.5 ± 0.4	**7.1 ± 0.7**	34.4 ± 3.0	21.5 ± 0.5	39.8 ± 4.7	**25.2 ± 0.4**	27.5
IABN + RS	**33.7 ± 6.4**	**30.0 ± 6.7**	**37.6 ± 2.9**	**13.6 ± 0.3**	**34.9 ± 1.9**	12.4 ± 1.2	**14.5 ± 1.7**	**13.9 ± 1.1**	**15.0 ± 3.1**	**14.0 ± 1.3**	7.2 ± 0.0	**7.4 ± 0.7**	21.1 ± 0.9	**26.2 ± 4.4**	25.9 ± 1.1	**20.5**
IABN + PBRS	34.9 ± 1.6	32.3 ± 3.1	39.6 ± 2.5	**13.6 ± 0.5**	35.8 ± 1.9	**11.8 ± 0.8**	**14.5 ± 0.5**	14.1 ± 0.6	15.2 ± 1.3	14.2 ± 0.6	7.7 ± 0.3	7.6 ± 0.6	**20.8 ± 0.7**	27.7 ± 2.6	26.4 ± 0.5	21.1

Table 18: Average classification error (%) and their corresponding standard deviations of varying ablation settings on CIFAR100-C with **temporally correlated test streams**, shown per domain. **Bold** type indicates those of lowest classification error. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	88.1 ± 0.2	86.8 ± 0.6	93.7 ± 0.6	64.9 ± 0.4	79.7 ± 0.9	55.5 ± 0.3	57.7 ± 0.2	53.8 ± 0.4	66.3 ± 0.8	59.3 ± 0.4	33.0 ± 0.3	81.4 ± 0.4	49.2 ± 0.4	73.6 ± 1.1	55.5 ± 0.3	66.6
IABN	79.3 ± 0.7	77.2 ± 0.7	84.2 ± 1.0	45.0 ± 0.6	69.6 ± 0.3	40.9 ± 0.3	43.1 ± 0.6	42.5 ± 0.4	48.6 ± 0.3	52.5 ± 0.5	30.4 ± 0.1	40.5 ± 0.7	47.6 ± 0.5	59.8 ± 1.1	56.2 ± 0.4	54.5
PBRS	68.8 ± 0.6	66.2 ± 0.4	73.3 ± 0.9	46.2 ± 0.6	64.9 ± 1.5	41.8 ± 0.6	41.7 ± 0.3	44.2 ± 0.4	48.5 ± 0.7	44.7 ± 0.2	**28.3 ± 0.2**	60.1 ± 0.4	**44.2 ± 0.4**	51.9 ± 0.8	**50.5 ± 0.5**	51.7
IABN + RS	66.8 ± 2.1	65.2 ± 0.3	73.1 ± 1.0	38.7 ± 0.4	63.0 ± 0.9	36.6 ± 0.0	38.0 ± 0.2	41.9 ± 0.8	43.9 ± 0.4	44.6 ± 0.5	29.5 ± 0.3	33.5 ± 0.7	46.0 ± 0.5	49.9 ± 0.9	52.4 ± 0.4	48.2
IABN + PBRS	**66.2 ± 0.8**	**64.2 ± 1.6**	**72.6 ± 0.4**	**37.2 ± 0.8**	**61.1 ± 0.7**	**35.4 ± 0.3**	**37.4 ± 0.4**	**40.0 ± 0.4**	**42.5 ± 0.3**	**43.4 ± 0.5**	29.4 ± 0.1	**32.1 ± 0.5**	44.3 ± 0.4	**47.5 ± 0.6**	51.3 ± 0.3	**47.0**

Table 19: Average classification error (%) and their corresponding standard deviations of varying ablation settings on CIFAR10-C with **uniformly distributed test streams**, shown per domain. **Bold** type indicates those of lowest classification error. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	74.0 ± 3.3	66.8 ± 3.5	75.3 ± 4.2	43.3 ± 2.7	48.0 ± 2.7	32.6 ± 1.2	35.2 ± 2.6	22.0 ± 0.4	33.0 ± 2.5	25.9 ± 0.9	8.5 ± 0.3	66.1 ± 1.8	23.4 ± 0.7	53.6 ± 0.7	26.8 ± 0.7	42.3
IABN	44.5 ± 2.7	41.4 ± 2.3	48.1 ± 1.9	16.3 ± 1.0	39.9 ± 0.1	13.9 ± 0.7	16.2 ± 0.7	14.9 ± 0.3	17.9 ± 0.6	16.4 ± 0.5	7.6 ± 0.2	8.8 ± 0.3	22.5 ± 0.4	34.1 ± 1.2	26.7 ± 0.6	24.6
PBRS	43.4 ± 0.8	37.9 ± 0.6	46.2 ± 1.5	21.8 ± 2.0	36.8 ± 1.0	18.1 ± 0.3	17.6 ± 0.8	16.1 ± 0.1	19.3 ± 0.5	15.2 ± 0.3	**7.1 ± 0.4**	32.5 ± 1.5	**20.0 ± 0.2**	30.7 ± 0.7	**23.8 ± 0.1**	25.8
IABN + RS	33.8 ± 1.6	31.1 ± 0.9	40.4 ± 1.3	13.3 ± 0.7	35.6 ± 0.2	11.8 ± 0.6	13.2 ± 0.3	14.6 ± 0.3	**14.9 ± 0.6**	14.7 ± 0.4	7.7 ± 0.2	8.1 ± 0.4	22.3 ± 0.5	**24.6 ± 1.9**	25.1 ± 1.2	20.7
IABN + PBRS	**33.5 ± 1.7**	**30.0 ± 1.6**	**38.2 ± 0.9**	**12.6 ± 0.8**	**34.4 ± 0.8**	**11.5 ± 0.5**	**12.9 ± 0.6**	**14.1 ± 0.2**	15.2 ± 0.8	**14.0 ± 0.6**	7.4 ± 0.2	**7.8 ± 0.2**	20.7 ± 0.3	24.7 ± 0.7	24.2 ± 0.4	**20.1**

Table 20: Average classification error (%) and their corresponding standard deviations of varying ablation settings on CIFAR100-C with **uniformly distributed test streams**, shown per domain. **Bold** type indicates those of lowest classification error. Averaged over 3 runs.

Method	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Avg
Source	88.1 ± 0.2	86.8 ± 0.6	93.7 ± 0.6	64.9 ± 0.4	79.7 ± 0.9	55.5 ± 0.3	57.7 ± 0.2	53.8 ± 0.4	66.3 ± 0.8	59.3 ± 0.4	33.0 ± 0.3	81.4 ± 0.4	49.2 ± 0.4	73.6 ± 1.1	55.5 ± 0.3	66.6
IABN	79.3 ± 0.6	77.2 ± 0.6	84.3 ± 1.0	45.0 ± 0.5	69.6 ± 0.2	40.9 ± 0.3	43.1 ± 0.6	42.5 ± 0.4	48.6 ± 0.3	52.5 ± 0.5	30.5 ± 0.1	40.5 ± 0.7	47.6 ± 0.5	59.8 ± 1.1	56.2 ± 0.4	54.5
PBRS	68.6 ± 1.0	66.0 ± 0.3	72.9 ± 0.3	45.3 ± 0.3	64.1 ± 0.8	40.9 ± 0.5	41.6 ± 0.5	43.7 ± 0.2	47.9 ± 0.2	44.2 ± 0.3	**28.3 ± 0.3**	59.9 ± 0.7	44.2 ± 0.5	51.1 ± 1.6	**50.4 ± 0.6**	51.3
IABN + RS	67.1 ± 1.2	65.6 ± 0.3	74.0 ± 0.4	39.0 ± 0.3	61.4 ± 1.3	36.5 ± 0.1	38.7 ± 0.8	41.4 ± 0.2	44.0 ± 0.4	45.0 ± 0.2	30.0 ± 0.2	34.0 ± 0.2	46.0 ± 1.4	48.8 ± 1.3	52.5 ± 0.5	48.3
IABN + PBRS	**65.6 ± 1.0**	**62.6 ± 0.7**	**72.0 ± 0.2**	**36.8 ± 0.7**	**60.5 ± 0.7**	**34.9 ± 0.5**	**36.7 ± 0.2**	**39.6 ± 0.2**	**41.7 ± 0.6**	**42.3 ± 0.3**	28.6 ± 0.2	**32.3 ± 0.9**	**43.8 ± 0.2**	**47.7 ± 0.4**	50.9 ± 0.2	**46.4**

C Replacing BN with IABN during test time

Table 21: Average classification error (%) and corresponding standard deviations of varying ablation settings on CIFAR10-C/100-C under temporally correlated (non-i.i.d.) and uniformly distributed (i.i.d.) test data stream. IABN* refers to replacing BN with IABN during test time (no pre-training with IABN layers). **Bold** type indicates those of lowest classification error. Averaged over 3 runs.

	Temporally correlated test stream			**Uniformly distributed test stream**		
Method	CIFAR10-C	CIFAR100-C	Avg	CIFAR10-C	CIFAR100-C	Avg
Source	42.3 ± 1.1	66.6 ± 0.1	54.4	42.3 ± 1.1	66.6 ± 0.1	54.4
IABN*	27.1 ± 0.4	60.8 ± 0.1	44.0	27.1 ± 0.4	60.8 ± 0.2	44.0
IABN	24.6 ± 0.6	54.5 ± 0.1	39.5	24.6 ± 0.6	54.5 ± 0.1	39.5
IABN*+PBRS	24.9 ± 0.2	55.9 ± 0.2	40.4	23.2 ± 0.4	55.3 ± 0.1	39.3
IABN+PBRS	**21.1 ± 0.6**	**47.0 ± 0.1**	**34.0**	**20.1 ± 0.5**	**46.4 ± 0.0**	**33.2**

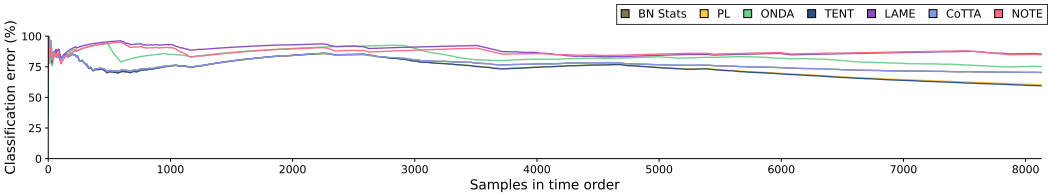
For pre-trained models with BN layers such as ResNet [12], NOTE needs to re-train the model by replacing BN layers with IABN layers in order to utilize the effectiveness of IABN. This requires additional computational cost of re-training, which might make it inconvenient to utilize off-the-shelf models. We further investigate whether simply switching BN to IABN without re-training still leads to performance gain.

Table 21 shows the result of this experiment, where IABN* refers to replacing BN with IABN during test time. We note that IABN* still shows significant reduction of errors under CIFAR10-C and CIFAR100-C datasets compared with BN (Source). We interpret this as the normalization correction in IABN is valid to some extent without re-training of the model. We notice that IABN* outperforms the baselines in CIFAR10-C with 27.1% error, while the second best (LAME) shows 36.2% error 1. In addition, IABN* also shows improvement combined with PBRS. This implies that IABN can be used without re-training the model, which aligns with the fully test-time adaptation paradigm introduced in a recent study [41].

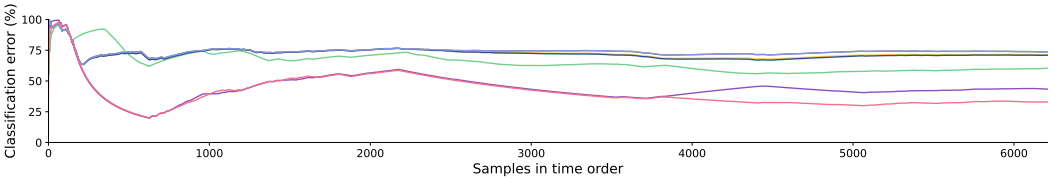
D License of assets

Datasets KITTI dataset (CC-BY-NC-SA 3.0), KITTI-rain dataset (CC-BY-NC-SA 3.0), CIFAR10, 100 (MIT License), ImageNet-C (Apache 2.0), MNIST-C (CC-BY-NC-SA 4.0), HARTH datset (MIT License), and the Extrasensory dataset (CC-BY-NC-SA 4.0)

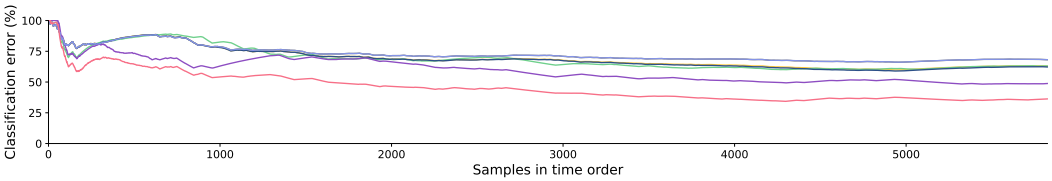
Codes Code for rain augmentation on KITTI dataset (Apache 2.0), torch-vision for ResNet18 and ResNet50 (Apache 2.0), code for depth estimation used in rain augmentation on KITTI dataset (UCLB ACP-A License), code for generating Dirichlet distributions (Apache 2.0), official repository of CoTTA (MIT License), official repository of TENT (MIT License), and the official repository of LAME (CC BY-NC-SA 4.0).



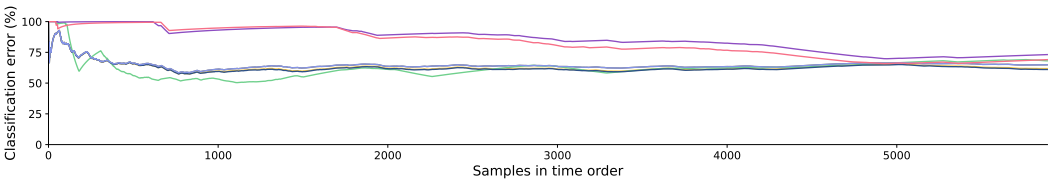
(a) S008.



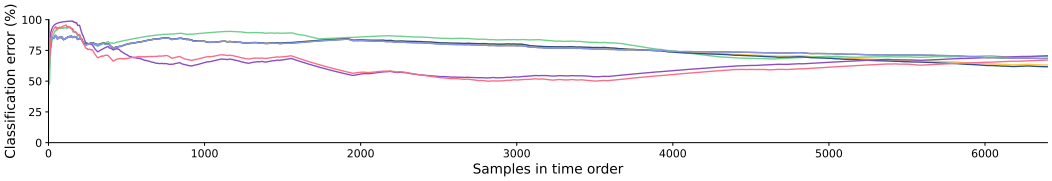
(b) S018.



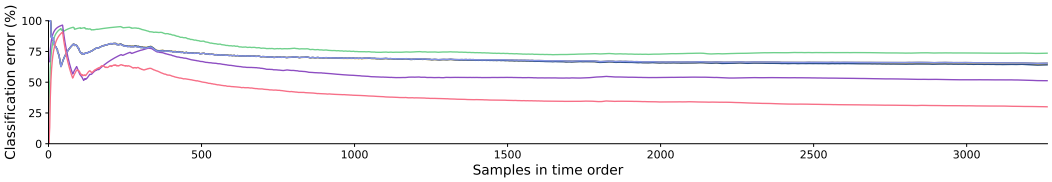
(c) S019.



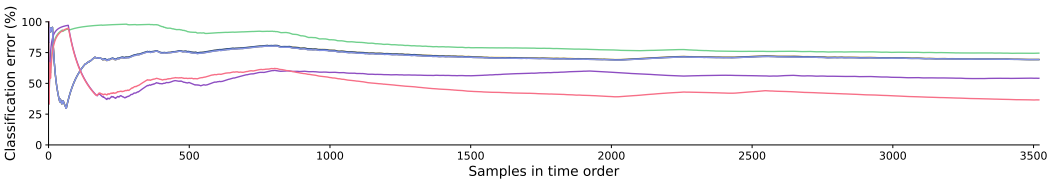
(d) S021.



(e) S022.

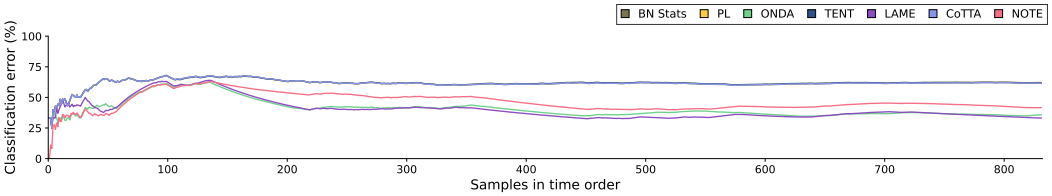


(f) S028.

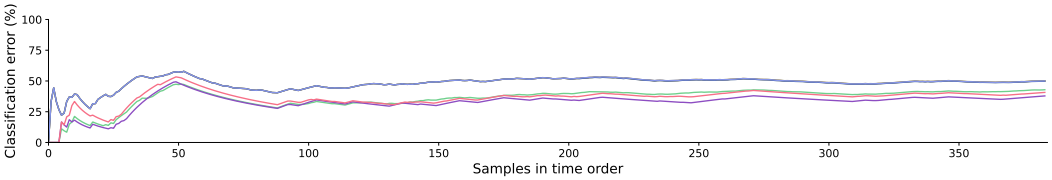


(g) S029.

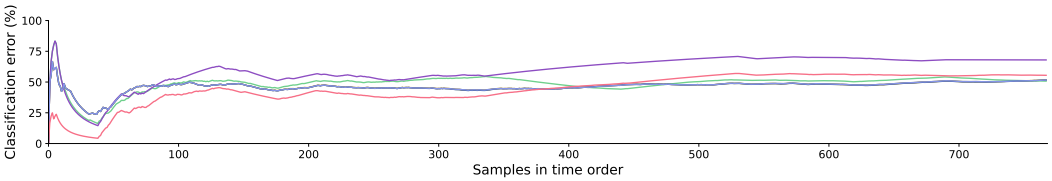
Figure 12: Illustration of the real-time cumulative classification error change of different methods on the HARTH dataset. The x-axis denotes the samples in order, whereas the y-axis denotes the error rate in percentage. Note that some lines are not clearly visible due to overlap.



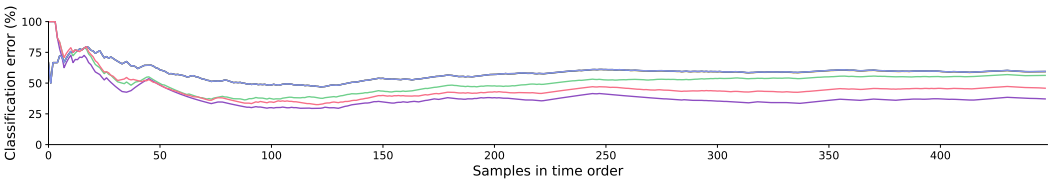
(a) 4FC.



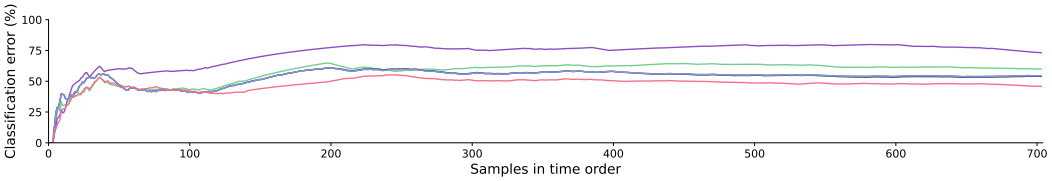
(b) 598.



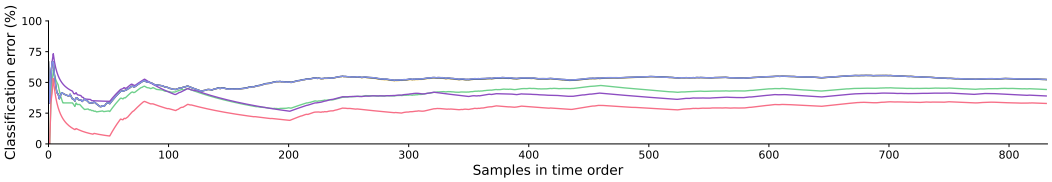
(c) 619.



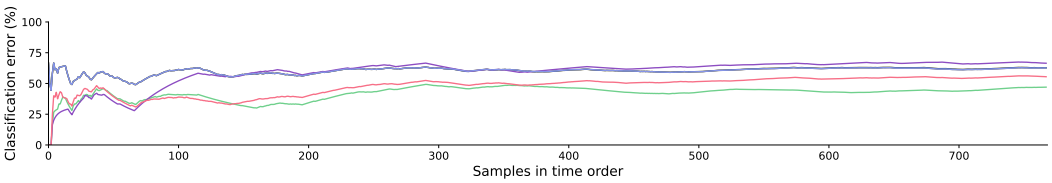
(d) 797.



(e) A5D.



(f) C48.



(g) D7D.

Figure 13: Illustration of the real-time cumulative classification error change of different methods on the Extrasensory dataset. The x-axis denotes the samples in order, whereas the y-axis denotes the error rate in percentage. Note that some lines are not clearly visible due to overlap.