





MemBN: Robust Test-Time Adaptation via Batch Norm with Statistics Memory

Juwon Kang¹, Nayeong Kim², Jungseul Ok², and Suha Kwak²

¹ GENGENAI, South Korea

² Pohang University of Science and Technology (POSTECH), South Korea
gjw0917@gengen.ai, {nayeong.kim, jungseul.ok, suha.kwak}@postech.ac.kr
<http://cvlab.postech.ac.kr/research/MemBN>

Abstract. Test-time adaptation (TTA) has emerged as a promising approach to dealing with latent distribution shifts between training and testing data. However, most of existing TTA methods often struggle with small input batches, as they heavily rely on batch statistics that become less reliable as batch size decreases. In this paper, we introduce memory-based batch normalization (MemBN) to enhance the robustness of TTA across a wide range of batch sizes. MemBN leverages statistics memory queues within each batch normalization layer, accumulating the latest test batch statistics. Through dedicated memory management and aggregation algorithms, it enables to estimate reliable statistics that well represent the data distribution of the test domain in hand, leading to improved performance and robust test-time adaptation. Extensive experiments under a large variety of TTA scenarios demonstrate MemBN’s superiority in terms of both accuracy and robustness.

Keywords: Test-time adaptation · Batch normalization

1 Introduction

Although deep neural networks (DNNs) have driven remarkable advances in a variety of vision tasks, they still suffer from severe performance degradation when there exists a distribution shift between training and test domains [6, 9, 19, 34, 35, 37]. In practice, such a distribution shift frequently occurs due to environment (*e.g.*, weather, time, or geolocation) changes in the real world. Moreover, it is impractical to forecast which domains a model will face in testing or to collect labeled training data for every possible test domain in advance.

In this context, test-time adaptation (TTA), the task of adapting a pre-trained model to latent test domains using unlabeled input data during testing *on the fly*, has been extensively studied [2, 13, 26, 37, 38, 47, 49]. Regarding that the motivation behind TTA was the need for domain adaptation in more realistic settings, one may expect that ideally TTA has to work in typical test environments where we need to process a stream of test data given a limited computational resource. Despite recent remarkable advances in TTA, however, most of the prior arts have been evaluated in settings that largely differ from real-world scenarios [3, 7, 15, 37, 38]. Indeed, a standard benchmark considers TTA processing test

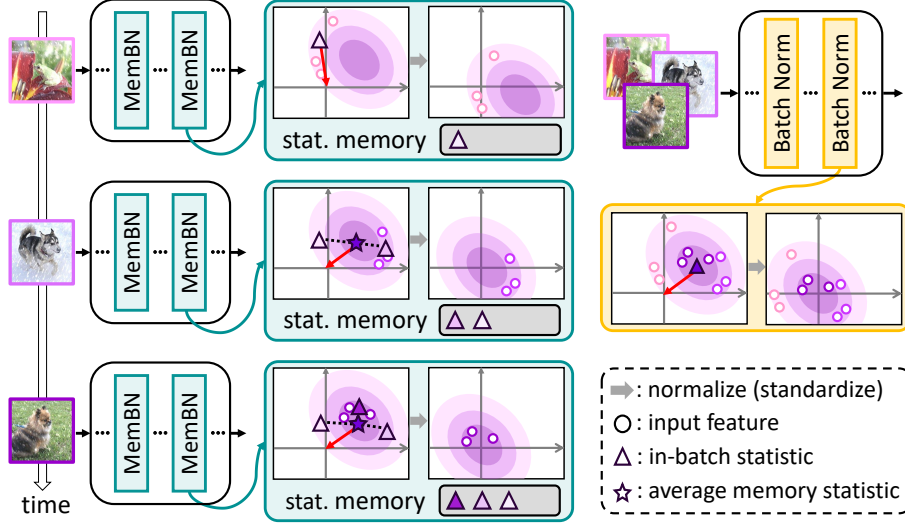


Fig. 1: The core concept of MemBN. MemBN layer retains in-batch statistics of test data and aggregates those stored in memory to derive more reliable statistics. As more statistics are progressively stored, the average memory statistics well approximate the feature distribution of the test data, similar to those derived from large input batches in batch normalization (as illustrated on the right).

data batch-by-batch given a large batch size (*e.g.*, 200), which is often infeasible due to the limited memory of edge devices and real-time constraints.

Unfortunately, most of existing TTA methods exhibit significant performance drops when handling small input batches [20, 25, 37, 46]. This issue arises mainly because they rely on normalization statistics estimated from the test batch in hand, *i.e.*, in-batch statistics, for batch normalization (BN) [12] to alleviate the distribution shift issue. When the size of input batch is small, such in-batch statistics fail to represent the distribution of test data and vary significantly for different batches, leading to performance degradation and unreliable test-time training. Recent studies have addressed this challenge by virtually increasing the size of input batches through data augmentation [25, 46] or by interpolation between normalization statistics of the entire training data and the in-batch statistics [20]. However, the former methods impose additional computation, and the latter still suffers from in-batch statistics of small test batches.

In this paper, we propose a **Memory-based Batch Normalization**, dubbed **MemBN**, to enhance the robustness of TTA across a wide range of batch sizes. The key idea is to introduce and manage *statistics memory queues* in each BN layer for accumulating statistics of the latest test batches, as illustrated in Fig. 1. Without any increase in input batch size during the feedforward pass, our method effectively simulates statistics of large batches by aggregating the statistics previously stored in the memory. In addition, we introduce a simple yet effective

memory reset strategy, allowing a model to respond flexibly to new domains in continually changing environments. In short, the combination of the memory queues and the memory reset strategy enables our method to calculate reliable statistics that well represent the data distribution of the current domain.

We further improve the robustness of our method by employing a linear combination of the source statistics and the average memory statistics as the normalization statistics. At the core of this technique lies a *layer-wise adaptive weighting* policy, which assigns a larger weight to the source statistics if the current in-batch statistics are closer to them in certain layers. This improves robustness by actively exploiting the source statistics, which are reliable as accumulated from large-scale training data, when the current test domain is little deviated from the source domain.

Our method with MemBN was evaluated under a large variety of TTA scenarios, specifically, on three datasets for classification [9] and four datasets for semantic segmentation [24, 29, 30, 42], in single domain [9], continual domain [38], and temporally correlated [7] TTA settings, and with a wide range of test batch sizes. Extensive experiments on these benchmarks demonstrate the superiority of MemBN. The contribution of this paper is three-fold:

- We propose a novel memory-based normalization method to TTA, which derives reliable statistics by leveraging the statistics memory that retains the latest batches from test domain.
- We present an adaptive weighting scheme interpolating source statistics and average memory statistics to adapt to test domain according to domain shift.
- Our approach surpasses existing TTA methods across diverse benchmarks and batch sizes, significantly outperforming them at small batch sizes.

2 Related work

Test-Time Adaptation (TTA). Early TTA methods focused on utilizing self-supervised learning using pretext tasks for unlabeled test data [21, 35]. However, these methods require modifying the training process or model architecture to incorporate auxiliary self-supervised learning tasks. On the other hand, spurred by Tent [37], fully test-time adaptation has been explored for adapting pretrained networks to the test domain in real-time without altering the training phase [2, 3, 13–16, 38, 50]. AdaContrat [3] and CoTTA [38] adopt momentum encoders, while Jang *et al.* [14] utilized nearest neighbor information for refining pseudo labels of test data. Recently, other studies have suggested to utilize additional lightweight information to prevent trained models from degenerating during the adaptation process. Niu *et al.* [26] calculated Fisher information [17] to constrain important model parameters and Kang *et al.* [15] leveraged class similarity and condensed data of training data. Also, TTAC [33] focused on clustering source domain data and aligning the feature distribution of test data with that of the source domain. Unlike these methods that require preprocessing steps to obtain additional information, our method allows fully test-time adaptation since it solely relies on statistics storage of the test domain without any preprocessing.

More practical scenarios for TTA. Although TTA has been proposed to tackle domain shifts in more realistic scenarios, experimental settings of early approaches in TTA are often insufficient to reflect real-world scenarios. One such issue is that conventional TTA methods [3, 26, 32, 37, 38] have been evaluated with large input batches and exhibit significant performance drops with small batch sizes, which are more prevalent in practical scenarios. This issue has been addressed recently by encouraging transformation invariance with augmented inputs [25, 46] and by a robust normalization method [20, 45]. However, these methods either impose a computational burden with augmented inputs or still depend on unreliable in-batch statistics of small test batches. Our method is free from this limitation in that it allows the model to efficiently derive more reliable statistics through the use of multiple statistics stored in memory. Meanwhile, as real-world data often exhibit high temporal correlations, recent studies [7, 43] have proposed robust TTA methods tailored for such temporally correlated test streams. Our method is also evaluated in this context, demonstrating the robustness across various scenarios and a wide range of batch sizes.

Normalization in TTA. Many TTA methods [3, 15, 26, 37, 38] rely on normalization using test batch statistics in the BN layer during test time, as introduced by Nado *et al.* [23]. However, this normalization scheme degrades performance in realistic scenarios, such as small batch inputs or temporally correlated test streams. One popular way of alleviating this issue involves interpolating source and test batch statistics with methods such as manual balancing coefficient tuning [7, 16, 31, 41], adjustments based on domain shift sensitivity [20], or dynamic adaption according to the distances of statistics during test time [45]. On the other hand, Zhao *et al.* [48] introduced a method to store and re-normalize test batch statistics using Exponential Moving Average (EMA) for class-imbalanced data streams. Our method ensures that multiple statistics calculated from the current domain are stored in memory to compute more reliable statistics for that specific domain. Additionally, it effectively interpolates between source and average memory statistics, harnessing the advantages from both of them.

3 Preliminary: Batch Norm in TTA

We first review the process of batch normalization (BN). Let $\mathbf{X} \in \mathbb{R}^{B \times C \times H \times W}$ be the input feature map of size (batch size B) \times (channel C) \times (height H) \times (width W). For simplicity, we omit the index of the BN layer. The in-batch statistics, channel-wise mean and variance $\boldsymbol{\mu}_{\text{in}}, \boldsymbol{\sigma}_{\text{in}}^2 \in \mathbb{R}^C$, of \mathbf{X} are computed as:

$$\boldsymbol{\mu}_{\text{in},c} = \frac{1}{BHW} \sum_{b=1}^B \sum_{h=1}^H \sum_{w=1}^W \mathbf{X}_{b,c,h,w}, \quad (1)$$

$$\boldsymbol{\sigma}_{\text{in},c}^2 = \frac{1}{BHW} \sum_{b=1}^B \sum_{h=1}^H \sum_{w=1}^W (\mathbf{X}_{b,c,h,w} - \boldsymbol{\mu}_{\text{in},c})^2, \quad (2)$$

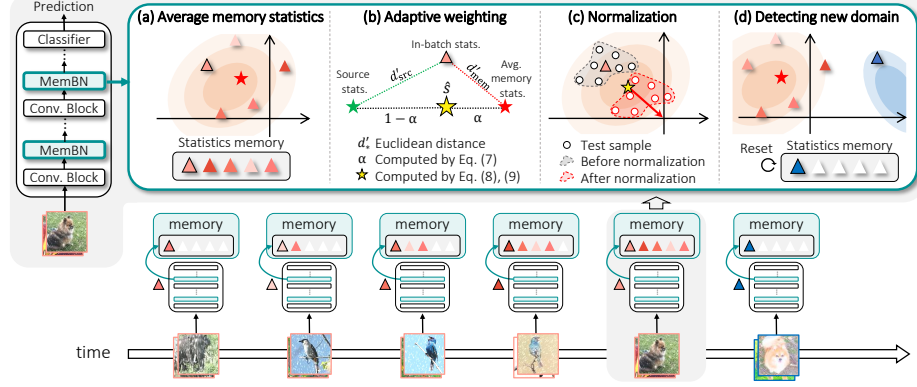


Fig. 2: Overview of MemBN. (a) Updating the memory queues with in-batch statistics of current inputs (triangles) and calculating average memory statistics (red star). This update occurs sequentially as each new input batch is processed. (b) Calculating layer-wise adaptive weight α and deriving normalization statistics (yellow star) by mixing average memory statistics and source statistics. (c) Normalizing the input features using the derived normalization statistics. (d) Resetting the memory queues when detecting a shift to a new domain, to prevent the outdated statistics from affecting the normalization process within the new domain.

where c is the channel index. In the traditional BN [12], the source statistics, μ_{src} and σ_{src}^2 , are estimated by exponential moving average of μ_{in} and σ_{in}^2 over training (or source) data, respectively. These statistics are then used to normalize the input feature \mathbf{X} in BN during testing:

$$\tilde{\mathbf{X}} = \mathbf{a} \circ \frac{\mathbf{X} - \mu_{\text{src}}}{\sqrt{\sigma_{\text{src}}^2 + \epsilon}} + \mathbf{b}, \quad (3)$$

where \mathbf{a} and \mathbf{b} denote the affine parameters of BN, and the basic operations of vectors are element-wise throughout this paper.

In the context of test-time adaptation, the in-batch statistics, μ_{in} of Eq. (1) and σ_{in}^2 of Eq. (2), that are directly computed from current test batch are in general utilized instead of the source statistics, μ_{src} and σ_{src}^2 , to mitigate the distribution gap between the source and target domain in the feature map [3, 15, 23, 26, 37, 38]. However, as observed in prior research [7, 25, 31, 45, 46, 48], the use of in-batch statistics that are entirely influenced by the current samples often leads to performance degradation, particularly when input batches are too small to estimate reliable statistics. This issue significantly undermines the robustness of TTA [20, 31, 37].

4 Method

To improve the robustness of TTA to small input batches, we propose aggregating normalization statistics in multiple test batches. This method aims to

derive reliable statistics, similar to those obtained from large input batches, by accumulating statistics of recently observed test samples. In realizing this idea, our method, MemBN, introduces dedicated memory for storing feature statistics computed from individual batches in each BN layer, along with a memory reset scheme. These statistics stored in the memory are then aggregated in order to effectively alleviate the issues of small batches by simulating large batches. Furthermore, we introduce an online adaptive weighting method to adjust weights between the aggregated statistics and source statistics according to the degree of domain shift.

An overview of MemBN is illustrated in Fig. 2. The remainder of this section describes how to maintain the statistics in memory and calculate average memory statistics (Sec. 4.1), our domain-shift aware weighting scheme (Sec. 4.2), and the overall process of MemBN (Sec. 4.3).

4.1 Statistics Memory Management

To obtain reliable normalization statistics by aggregating in-batch statistics (μ_{in} of Eq. (1) and σ_{in}^2 of Eq. (2)) of multiple test batches, we deploy memory queues of identical length that retain a few latest in-batch statistics for each BN. To be specific, during the testing phase, the in-batch statistics, μ_{in} and σ_{in} , are stored in separate memory queues, \mathcal{M}_{μ} and \mathcal{M}_{σ} , respectively, in the first-in first-out manner. Average memory statistics, μ_{mem} and σ_{mem} , are then computed by

$$\mu_{\text{mem},c} = \frac{1}{|\mathcal{M}_{\mu}|} \sum_{\mu \in \mathcal{M}_{\mu}} \mu_c, \quad (4)$$

$$\sigma_{\text{mem},c}^2 = \frac{1}{|\mathcal{M}_{\sigma}|} \sum_{\sigma \in \mathcal{M}_{\sigma}} \sigma_c^2 + \frac{1}{|\mathcal{M}_{\mu}|} \sum_{\mu \in \mathcal{M}_{\mu}} \mu_c^2 - \frac{1}{|\mathcal{M}_{\mu}|^2} \left(\sum_{\mu \in \mathcal{M}_{\mu}} \mu_c \right)^2. \quad (5)$$

These equations are derived in the supplement. By leveraging multiple statistics in the memory queues, μ_{mem} and σ_{mem} better approximate the feature distribution of test data than the in-batch statistics. In other words, this aggregation method mitigates the risk of computing inadequate statistics, thereby particularly useful for dealing with small batches and enhancing the robustness of TTA across a wide range of batch sizes. These average memory statistics are subsequently used for feature map normalization, in conjunction with an adaptive weighting scheme detailed in Sec. 4.2.

This aggregation method works properly per se when the test domain remains static. However, it may encounter challenges when the domain changes during testing, *e.g.*, in continually changing environments [38]. This issue arises because statistics stored in a part of the memory may have been computed from previous domains and thus may be irrelevant to the current domain.

To address this issue, we propose to detect such situations by comparing current in-batch statistics with those previously stored in memory, *i.e.*, in-memory statistics. For $\mathbf{s} \in \{\mu, \sigma\}$, let $d_i^{\mathbf{s}}$ be the Euclidean distance from average memory

statistics \mathbf{s}_{mem} to the i -th in-memory statistics \mathbf{s}_i , and $d_{\text{in}}^{\mathbf{s}}$ the distance from \mathbf{s}_{mem} to the current in-batch statistics \mathbf{s}_{in} , where $d_*^{\mathbf{s}} = \|\mathbf{s}_* - \mathbf{s}_{\text{mem}}\|_2$. Utilizing these distances, if the following inequality holds, we consider that a shift to a new domain has occurred:

$$(d_{\text{in}}^{\mathbf{s}} - \bar{d}^{\mathbf{s}}) \left(\frac{1}{|\mathcal{M}_{\mathbf{s}}|} \sum_{i=1}^{|\mathcal{M}_{\mathbf{s}}|} (d_i^{\mathbf{s}} - \bar{d}^{\mathbf{s}})^2 \right)^{-\frac{1}{2}} > k \quad (6)$$

where $\bar{d}^{\mathbf{s}} = \frac{1}{|\mathcal{M}_{\mathbf{s}}|} \sum_{i=1}^{|\mathcal{M}_{\mathbf{s}}|} d_i^{\mathbf{s}}$ and k is a hyperparameter. The first term quantifies the difference between the distance of the in-batch statistics and the average distance of in-memory elements from the average memory statistics, while the second term measures the standard deviation of the in-memory elements' distances. This use of the standard score allows to use a single hyper-parameter k to detect domain shifts across all layers. When $d_{\text{in}}^{\mathbf{s}}$ significantly exceeds the variation among the previously observed distances $d_i^{\mathbf{s}}$, the current data are likely drawn from a new domain and thus a shift towards a new domain may happen at that time. Once such a discrepancy is identified, all memories of all BN layers are reset so that outdated statistics from previous domains do not adversely affect the current one. This method is empirically validated in Sec. 5.5.

4.2 Adaptive Weighting

Beyond merely utilizing the average memory statistics, we introduce an adaptive weighting scheme to combine them with the source statistics. Prior TTA methods underscore the effectiveness of merging in-batch statistics with source statistics, enabling not only adaptation to test domains but also leveraging well-established source knowledge [20, 31, 41, 45]. Also, as suggested in previous work [18], it has been observed that different layers of a model exhibit different sensitivities to the type of domain shifts. Motivated by these observations, we propose to dynamically adjust the mixing ratio between source and average memory statistics to derive normalization statistics. Considering that the average memory statistics are aggregated to approximate the overall feature distribution of the test data, we regard them as representative information depicting the characteristics of the current test domain. We then assess the degree of domain shift by the distances from the in-batch statistics to source statistics and average memory statistics.

In detail, our method measures the degree of domain shift by the Euclidean distances from in-batch statistics to the average memory and source statistics, d'_{mem} and d'_{src} , respectively, by $d'_* = \|\boldsymbol{\mu}_{\text{in}} - \boldsymbol{\mu}_*\|_2 + \|\boldsymbol{\sigma}_{\text{in}}^2 - \boldsymbol{\sigma}_*^2\|_2$. The adaptive weighting factor α is then derived from these distances per BN as follows:

$$\alpha = \left(1 - \frac{d'_{\text{src}}}{d'_{\text{mem}} + d'_{\text{src}}} \right)^{\gamma}, \quad (7)$$

where γ is a hyperparameter for the gamma correction [28]. Then, the source and average memory statistics are combined by the following equations:

$$\hat{\boldsymbol{\mu}} = \alpha \boldsymbol{\mu}_{\text{src}} + (1-\alpha) \boldsymbol{\mu}_{\text{mem}}, \quad (8)$$

$$\hat{\boldsymbol{\sigma}}^2 = \alpha \boldsymbol{\sigma}_{\text{src}}^2 + (1-\alpha) \boldsymbol{\sigma}_{\text{mem}}^2 + \alpha(1-\alpha)(\boldsymbol{\mu}_{\text{src}} - \boldsymbol{\mu}_{\text{mem}})^2. \quad (9)$$

A smaller distance to source statistics (d'_{src}) increases α , giving more weight to source statistics, while a smaller distance to average memory statistics (d'_{mem}) decreases α , thereby reducing the reliance on source statistics. It is noteworthy that our method assigns weights layer-by-layer, based on the degree of domain shift, without requiring additional training before testing unlike previous work [20].

4.3 Normalization in MemBN

By computing the average memory statistics and applying the adaptive weighting, we calculate the normalization statistics $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ as in Eq. (8) and (9), and substitute them for $\boldsymbol{\mu}_{\text{src}}$ and $\boldsymbol{\sigma}_{\text{src}}$ in Eq. (3). However, as suggested in prior studies [11, 48], when a feature map is normalized using constant statistics rather than those derived from the feature map itself, the gradient no longer propagates through the normalization factors. This leads to gradient descent optimization no longer taking normalization into account, which can potentially result in unbounded growth in model parameters.

To address this problem, we incorporate the batch renormalization technique [48] and proceed with the overall normalization process as follows:

$$\text{ReNorm}(\mathbf{X}; \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}) = \frac{\mathbf{X} - \boldsymbol{\mu}_{\text{in}}}{\boldsymbol{\sigma}_{\text{in}}} \circ \frac{\text{sg}(\boldsymbol{\sigma}_{\text{in}})}{\hat{\boldsymbol{\sigma}}} + \frac{\text{sg}(\boldsymbol{\mu}_{\text{in}}) - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}}, \quad (10)$$

$$\text{MemBN}(\mathbf{X}; \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}, \mathbf{a}, \mathbf{b}) = \mathbf{a} \circ \text{ReNorm}(\mathbf{X}; \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}) + \mathbf{b}, \quad (11)$$

where $\text{sg}(\cdot)$ denotes the stop gradient operation. This technique contributes to more stable optimization and diminishes the sensitivity to the learning rate. The overall process of MemBN is described in the supplement.

5 Experiments

5.1 Experimental setting

Evaluation benchmarks. MemBN was first evaluated on three common image corruption benchmarks [9]: CIFAR10-C, CIFAR100-C, and ImageNet-C. These datasets consist of 15 corruption types, and each type has five severity levels with level 5 as the most severe condition. Our method was also validated on semantic segmentation benchmarks for domain generalization, including Cityscapes [5], BDD-100K [42], Mapillary [24], GTA5 [29] and SYNTHIA [30].

Baselines. We compared our method with normalization-based and optimization-based methods. MemBN was first evaluated alongside the following normalization-based methods: (1) TBN [23] using in-batch statistics of test data for normalization. (2) AdaptiveBN [31], (3) α -BN [41], and (4) MixNorm [10] that all blend

source and test (in-batch) statistics using predefined coefficients. (5) group normalization (GN) [40] that divides the channels into groups and normalizing each group independently. (6) TTN [20] that interpolates source statistics and test batch statistics according to the domain-shift sensitivity of each BN layer. As for optimization-based methods, MemBN was incorporated and compared with the following techniques: (7) TENT [37] that updates BN affine parameters via entropy minimization. (8) CoTTA [38] estimating accurate pseudo labels by leveraging momentum encoders and multiple augmented inputs. (9) TeSLA [36] that utilized a flipped cross entropy loss and knowledge distillation with adversarial augmentations. (10) EATA [26] incorporating a data selection strategy for reliable samples and a Fisher regularizer to prevent forgetting issue. (11) NOTE [7] that combines source and test instance statistics with prediction-balanced memory bank to simulate an i.i.d. data stream from a non-i.i.d. stream; we set the size of memory bank to the test batch size for fairness in memory constraint. In detail, TENT and CoTTA are built on TBN.

Evaluation scenarios. For the classification task, MemBN was evaluated not only for adaptation to a single domain but also in the continual TTA setting, where the test domain changes continually over time, and with temporally correlated test streams, *i.e.*, the non-i.i.d. setting of NOTE [7]. Our evaluation was conducted across a diverse range of test batch sizes, from 200 to 1. For the segmentation task, following the previous work [20], a segmentation model pre-trained on Cityscapes is adapted to the validation splits of real-world datasets, including Cityscapes, BDD-100K, and Mapillary, as well as to synthetic datasets such as GTA5 and SYNTHIA with test batch sizes of 2 and 1.

Implementation details. MemBN was evaluated under the conventional TTA setting where test data is streamed online and the pretrained model is adapted on the fly. For the hyperparameter of MemBN, we keep each statistics memory length N at 128 and the parameter for adaptive weighting γ at 0.3 across all settings. The memory reset parameter k is set to 6.5 for batch sizes of 200 and 64, and to 10 for all the other batch sizes. More details are in the supplement.

5.2 Experiment on image classification

Single domain TTA. In Table 1, our method is compared with previous TTA baselines on CIFAR-C. Conventional BN (CBN) maintained consistent performance across various test batch sizes as it does not perform adaptation during test time, but its performance does not excel. TBN outperformed CBN at larger batch sizes but experienced a significant performance drop as the batch size decreased, even performing worse than CBN. This demonstrates that the uncertainty of the test batch statistics obtained from smaller batch sizes significantly impacts the performance of TTA. Methods such as AdaptiveBN, α -BN, and MixNorm, which mix TBN and CBN using fixed coefficient, demonstrated better performance than CBN and TBN across most batch sizes. However, these methods also suffered performance degradation at batch sizes 4, 2, and 1. TTN, blending TBN and CBN based on domain sensitivity, surpassed BN-based methods at larger batch sizes but couldn’t avoid degradation at small batch sizes

Table 1: Single domain adaptation on CIFAR10-C and CIFAR100-C. Error rate (\downarrow) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 [44] for each test batch size. We mark the best and second-best performance in **bold** and underline, respectively.

		CIFAR10-C							CIFAR100-C						
Method		200	64	16	4	2	1	Avg.	200	64	16	4	2	1	Avg.
Norm.	Source (CBN)	18.27	18.27	18.27	18.27	18.27	18.27	18.27	46.75	46.75	46.75	46.75	46.75	46.75	46.75
	TBN [23]	14.49	15.02	17.10	26.28	35.65	90.00	33.09	39.25	40.21	44.03	59.10	80.65	99.04	60.38
	AdaptiveBN [31]	<u>12.21</u>	<u>12.31</u>	12.89	14.51	15.79	16.14	13.98	<u>36.56</u>	36.85	38.19	41.18	43.26	<u>44.01</u>	<u>40.01</u>
	α -BN [41]	13.78	13.77	13.89	14.54	<u>15.16</u>	15.47	14.44	39.72	39.85	39.99	41.34	<u>42.66</u>	45.64	41.53
	MixNorm [10]	13.85	14.41	14.23	14.60($B=5$)	-	<u>15.09</u>	14.44	-	-	-	-	-	-	-
	TTN [20]	11.67	11.80	12.13	<u>13.93</u>	15.83	17.99	<u>13.89</u>	35.58	35.84	<u>36.73</u>	<u>41.08</u>	46.67	57.71	42.27
	MemBN (Ours)	12.68	12.41	<u>12.16</u>	12.30	12.46	12.88	12.48	36.97	<u>36.70</u>	36.58	36.98	37.56	38.75	37.26
Optim.	TENT [37]	12.08	14.78	16.90	25.61	35.69	90.00	32.51	35.52	39.90	43.78	59.02	80.68	99.02	59.65
	TENT + TTN [20]	<u>11.28</u>	11.52	<u>12.04</u>	<u>13.95</u>	<u>15.84</u>	<u>17.94</u>	<u>13.77</u>	<u>35.16</u>	35.57	<u>36.55</u>	<u>41.18</u>	<u>46.63</u>	<u>58.33</u>	<u>42.24</u>
	TENT + MemBN (Ours)	11.08	<u>11.63</u>	11.22	11.33	11.86	12.15	11.55	35.14	<u>35.61</u>	34.94	36.02	37.31	37.37	36.07

Table 2: Single domain adaptation on ImageNet-C. Error rate (\downarrow) averaged over 15 corruption types with severity level 5 using ResNet-50 [8] for each test batch size. We mark the best performance in **bold**. The results of using pre-trained models with GN instead of BN are indicated by \dagger .

		Test batch size					Avg. Error
Method		64	16	4	2	1	
Source (CBN)		93.34	93.34	93.34	93.34	93.34	93.34
Norm.	TBN [23]	74.24	76.81	85.74	95.35	99.86	86.40
	AdaptiveBN [31]	77.86	81.47	86.71	90.15	91.11	85.46
	α -BN [41]	86.06	86.32	87.16	88.33	90.45	87.66
	GN † [40]	69.38	69.38	69.38	69.38	69.38	69.38
	TTN [20]	72.21	73.18	76.98	81.52	88.49	78.48
	MemBN (Ours)	67.07	67.08	67.38	67.78	68.54	67.57
Optim.	TENT [37]	66.56	72.61	93.37	99.46	99.90	86.41
	TENT+GN † [40]	67.31	67.41	68.34	68.96	69.44	68.29
	TENT + TTN [20]	71.42	72.45	76.66	81.89	91.00	78.68
	TENT + MemBN (Ours)	65.56	63.60	66.55	67.62	67.88	66.24
	TeSLA [36]	56.34	62.77	78.23	92.47	99.80	77.92
	TeSLA + MemBN (Ours)	55.90	58.89	65.19	66.02	67.31	62.66
	EATA [26]	52.38	58.54	81.72	93.27	99.86	77.15
	EATA + MemBN (Ours)	51.18	51.78	55.93	59.51	73.72	58.42

2 and 1. In contrast, MemBN consistently achieved superior performance in terms of average accuracy across all batch sizes, both in normalization-based and optimization-based methods, indicating the effectiveness of simulating large batch sizes through the memory of test statistics. Particularly, it is worth noting that while TTN requires an additional training phase (referred to as the post-training phase) to optimize interpolation weights between CBN and TBN before testing, our method consistently outperforms it without requiring any training phase for a pretrained model.

Table 2 presents the results on ImageNet-C. TBN, AdaptiveBN, α -BN, and TTN experienced performance degradation at small batch sizes (4, 2, and 1) consistent with Table 1. MemBN achieved state-of-the-art by a considerable margin across all batch sizes, demonstrating its scalability and effectiveness on large-scale dataset. While GN has been known for its robustness with small batch sizes and domain shifts [27], MemBN was consistently superior to GN.

Table 3: Continual domain adaptation on CIFAR10-C and CIFAR100-C. Error rate (\downarrow) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 [44] for each batch size. We mark the best and second-best performance in **bold** and underline, respectively.

Method		CIFAR10-C							CIFAR100-C						
		200	64	16	4	2	1	Avg.	200	64	16	4	2	1	Avg.
Norm.	Source (CBN)	18.27	18.27	18.27	18.27	18.27	18.27	18.27	46.75	46.75	46.75	46.75	46.75	46.75	46.75
	TBN [23]	14.48	15.04	17.38	26.23	35.43	90.00	33.09	39.31	40.20	44.13	59.29	80.42	99.04	60.40
	TTN [20]	11.67	11.80	<u>12.13</u>	<u>13.93</u>	<u>15.83</u>	<u>17.99</u>	<u>13.89</u>	35.58	35.84	<u>36.73</u>	<u>41.08</u>	<u>46.67</u>	<u>57.71</u>	<u>42.27</u>
	MemBN (Ours)	<u>12.33</u>	<u>12.13</u>	12.02	12.10	12.19	12.29	12.18	<u>36.91</u>	<u>36.66</u>	36.59	36.85	36.79	36.99	36.80
Optim.	CoTTA [38]	12.46	14.60	21.26	45.69	58.87	90.00	40.48	39.75	42.20	52.94	73.69	87.66	98.99	65.87
	TENT [37]	12.54	13.52	15.69	26.23	35.77	90.00	32.29	<u>36.11</u>	37.90	43.78	58.71	81.76	99.04	59.55
	TENT + TTN [20]	<u>11.44</u>	<u>11.60</u>	<u>12.08</u>	<u>16.14</u>	<u>18.36</u>	<u>22.40</u>	<u>15.33</u>	43.50	<u>37.60</u>	<u>38.28</u>	<u>44.60</u>	<u>54.29</u>	<u>80.63</u>	<u>49.82</u>
	TENT + MemBN (Ours)	11.38	11.30	11.47	11.55	12.14	12.11	11.66	35.13	35.75	35.46	36.32	37.03	37.56	36.36

Table 4: Temporally correlated (non-i.i.d.) domain adaptation on ImageNet-C. Error rate (\downarrow) averaged over 15 corruption types with severity level 5 is reported using ResNet-18 [8] for each test batch size. We mark the best performance in **bold**.

Method		Test batch size					Avg. Error
		64	16	4	2	1	
Norm.	Source (CBN)	86.10	86.10	86.10	86.10	86.10	86.10
	TBN [23]	96.50	98.40	99.00	99.30	99.90	98.62
	NOTE [7]	79.77	81.72	85.37	88.40	83.41	83.74
	MemBN (Ours)	74.52	76.48	78.62	80.64	83.23	78.70
Optim.	TENT [37]	96.90	98.38	99.08	99.51	99.85	98.66
	TENT + NOTE [7]	76.09	75.57	81.24	89.55	98.37	84.16
	TENT + MemBN (Ours)	74.22	75.46	78.22	81.02	85.27	79.02

Interestingly, even with a large batch size of 64, MemBN improves performance over previous work. This indicates that while reliable statistics computation typically requires batch sizes larger than 64 for large-scale datasets, our method effectively mitigates this issue and enhances overall performance across a wide range of batch sizes. In addition, to confirm the applicability of MemBN with optimization-based TTA methods other than TENT [37], we incorporated it to TeSLA [36] and EATA [26]. MemBN is seamlessly integrated to these methods and improved the performance across most batch sizes, indicating its versatility across various optimization-based TTA methods.

Continual TTA. Results in the continual TTA setting are reported in Table 3. CoTTA, utilizing multiple augmented inputs and an additional momentum teacher model, suffers significant performance degradation under this setting with small batch sizes. Also, TTN exhibited performance degradation at extremely smaller batch sizes despite being designed to perform well with smaller batch sizes. On the contrary, MemBN consistently presents superior performance without the need for additional inputs or models. This demonstrates the efficacy and efficiency of its memory-based strategy.

Temporally correlated TTA. As shown in Table 4, our method is also effective even under the temporally correlated (*i.e.*, non-i.i.d.) test data on ImageNet-C. The proposed method achieved state-of-the-art performance in both normalization-based and optimization-based methods across most batch sizes. In such scenarios,

Table 5: Adaptation on DG benchmarks in semantic segmentation. mIoU(\uparrow) on four unseen domains using ResNet-50 based DeepLabV3+ [4]. We mark the best performance in **bold**.

Batch size	Method (Cityscapes \rightarrow)	BDD-100K	Mapillary	GTA5	SYNTHIA	Cityscapes
	Source [4]	43.50	54.37	43.71	22.78	77.51
2	TENT [37]	43.30	47.80	43.57	25.92	72.93
	TENT + TTN [20]	47.89	57.84	46.18	27.29	75.04
	TENT + MemBN (Ours)	50.34	57.87	46.39	27.90	77.23
1	TENT [37]	40.78	44.00	39.52	25.33	69.92
	TENT + TTN [20]	-	-	-	-	-
	TENT + MemBN (Ours)	50.94	57.74	46.20	27.03	77.42

as noted by [7], most methods [23, 37, 38] suffer from a severe performance drop, due to the inaccurate and biased estimation of normalization statistics from class-imbalanced data. Our approach, however, alleviates this issue by aggregating multiple statistics from previous data, enabling it to benefit from a similar effect of more class-balanced data. The results on CIFAR10-C and CIFAR100-C are reported in the supplement.

5.3 Experiment on semantic segmentation

Table 5 presents the quantitative results of semantic segmentation by evaluating DeepLabV3+ [4] pretrained with the Cityscapes training set on the validation splits of real-world datasets (BDD-100k, Mapillary, and Cityscapes) and synthetic datasets (GTA5 and SYNTHIA). Our method achieved the best in mIoU across the five datasets for both batch sizes 2 and 1. Notably, while TENT experienced a significant performance drop at batch size 1, MemBN demonstrated superior performance for both batch sizes, showcasing its robustness.

Moreover, these results show the outstanding versatility of MemBN, in terms of both tasks (*i.e.*, segmentation as well as classification) and types of domain shifts (*i.e.*, real versus synthetic and diverse weather conditions as well as common corruption). Interestingly, while the previous best method, TTN, exhibited a decrease in performance on Cityscapes compared to the source model, MemBN demonstrated comparable performance. These overall results demonstrate the adaptability of our method in diverse application scenarios.

5.4 Ablation study

Table 6 illustrates the impact of each proposed method in the continual TTA setting on CIFAR10-C and CIFAR100-C. We first compared TBN’s performance (row 1) with statistics memory of size 1, updated by the Exponential Moving Average (EMA) of test batch statistics (row 2). Interestingly, there are substantial performance improvements in average performance, even by referring to previous statistics in a simple way. After introducing a statistics memory with a capacity of 128 and utilizing average memory statistics (row 3), we observed a performance decline at large batch sizes but an improvement at smaller ones. This

Table 6: Ablation study on each component of our method in continual domain adaptation on CIFAR10-C and CIFAR100-C. Error rate (\downarrow) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 [44] for each batch size. We mark the best and second-best performance in **bold** and underline, respectively.

Method				CIFAR10-C								CIFAR100-C							
	Memory size	Memory reset	Adaptive weight	200	64	16	4	2	1	Avg.		200	64	16	4	2	1	Avg.	
Norm.	0	-	-	14.48	15.04	17.38	26.23	35.43	90.00	33.09		39.31	40.20	44.13	59.29	80.42	99.04	60.40	
	1 (EMA update)	-	-	14.63	14.45	14.73	16.18	18.19	23.59	16.96		39.34	39.19	39.75	42.13	45.35	53.19	43.16	
	128	-	-	21.05	18.14	15.31	15.71	<u>15.11</u>	19.08	17.40		49.54	45.90	40.79	40.49	41.57	46.79	44.18	
	128	O	-	<u>14.36</u>	<u>14.31</u>	<u>14.43</u>	<u>14.88</u>	15.43	<u>17.03</u>	<u>15.07</u>		<u>38.92</u>	<u>38.89</u>	<u>39.09</u>	<u>39.88</u>	<u>41.02</u>	<u>43.20</u>	<u>40.17</u>	
	128	O	O	12.33	12.13	12.02	12.10	12.19	12.29	12.18		36.91	36.66	36.59	36.85	36.79	36.99	36.80	
Optim.	0	-	-	12.54	13.52	15.69	26.23	35.77	90.00	32.29		<u>36.11</u>	37.90	43.78	58.71	81.76	99.04	59.55	
	1 (EMA update)	-	-	12.62	12.03	12.06	14.12	16.10	24.60	15.25		36.40	<u>35.46</u>	36.80	39.42	44.60	55.25	41.32	
	128	-	-	17.43	14.35	12.47	12.92	<u>13.29</u>	<u>16.79</u>	14.54		46.21	41.11	38.44	37.41	<u>39.60</u>	44.33	41.18	
	128	O	-	<u>12.37</u>	<u>11.88</u>	<u>11.74</u>	<u>12.85</u>	13.46	17.73	<u>13.34</u>		36.28	35.14	<u>36.08</u>	<u>37.17</u>	39.79	<u>44.30</u>	<u>38.13</u>	
	128	O	O	11.38	11.30	11.47	11.55	12.14	12.11	11.66		35.13	35.75	35.46	36.32	37.03	37.56	36.36	

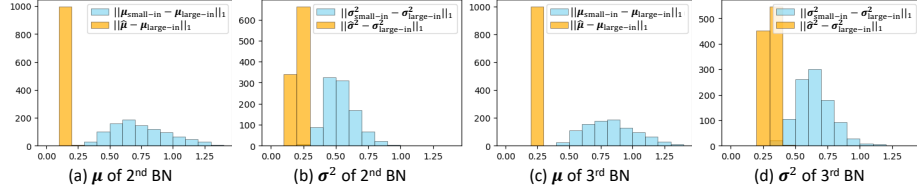


Fig. 3: Histogram of distances between in-batch statistics of a small batch size and those of a large batch size, and between normalization statistics of MemBN and in-batch statistics of the large batch size. These are computed using WideResNet-40-2 [44].

phenomenon is attributed to EMA prioritizing recent batch statistics, thereby enhancing the model’s adaptability to the current domain. However, this emphasis on recent statistics led to unreliable statistics estimation for small inputs. Conversely, employing specific-length memory statistics without reset introduced some interference at large inputs but was beneficial for achieving stability at small inputs. Then, adding the memory reset scheme (row 4) consistently improved performance across most batch sizes, highlighting the effectiveness of the proposed memory management scheme in adapting to domain changes while obtaining stable statistics. Specifically, we achieved performance comparable to TBN with batch sizes of 200 or 64, even with as few as 16 in this case. Finally, our final method (row 5), which includes adaptive weighting, outperformed all other variants, demonstrating the effectiveness of interpolating between average memory statistics and source statistics.

5.5 In-depth analysis

Analysis on normalization statistics of MemBN. To demonstrate MemBN’s ability to capture more reliable statistics, we measure L1 distances between in-batch statistics of a small batch size (4) and those of a large batch size (200), as well as between the normalization statistics of MemBN and in-batch statistics of the large batch size. Figure 3 illustrates these distances for 4k test data streams.

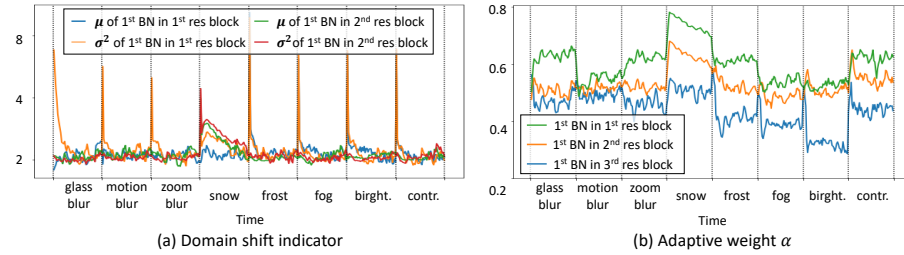


Fig. 4: (a) Domain shift indicator of Eq. (6). (b) Adaptive weight α . These values are computed using WideResNet-40-2 on CIFAR10-C in the continual TTA setting.

It reveals that the normalization statistics of MemBN align more closely with in-batch statistics of large batches than with those of small batches.

Detection of domain shift by Eq. (6). Recalling that the left term in Eq. (6) quantifies how different the current batch statistics are from those in memory, it can be interpreted as an indicator of domain shift. Figure 4 (a) shows the value of it under the continual TTA setting. The value soars right after encountering any domain shift. This enables prompt memory reset and adaptation.

Analysis on adaptive weight α . Figure 4 (b) illustrates how the trend of α changes with domain shifts and how the scale of α varies across layers in the continual TTA setting. While the weight in each layer exhibited distinct values, there is a tendency to maintain a certain level of consistency within each domain.

6 Conclusion

We have proposed a novel memory-based normalization method, MemBN, that significantly enhances the robustness of TTA across a wide range of batch sizes. By storing the latest statistics from the test domain in memory and implementing an effective memory reset scheme, our method is capable of deriving more reliable statistics and simulating those of large batches without any increase in the input size. In addition, our method performs layer-wise adaptive weighting between average memory statistics and source statistics, further enhancing TTA performance based on the degree of domain shift. The proposed method is a fully test-time adaptation method that does not require any preprocessing prior to testing, but merely stores in-batch statistics during testing, imposing a minimal overhead, and thus can operate effectively even under real-world scenarios.

Limitation: Our method exhibits a limitation in its applicability to layer normalization [1], commonly used in Vision Transformer (ViT). Despite the widespread use of transformer backbones, there are also studies showing that CNNs with BN are as competitive as ViTs [22, 39]. Therefore, we believe that recent studies targeting BNs [7, 20] as well as our work, remain valuable.

Acknowledgement: This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics (SRFC-IT1801-52), Samsung Electronics Co., Ltd (IO201210-07948-01), the NRF grant (NRF-2021R1A2C3012728) and the IITP grants funded by Ministry of Science and ICT, Korea (No.RS-2021-II210739).

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Boudiaf, M., Mueller, R., Ben Ayed, I., Bertinetto, L.: Parameter-free online test-time adaptation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8344–8353 (2022)
3. Chen, D., Wang, D., Darrell, T., Ebrahimi, S.: Contrastive test-time adaptation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 295–305 (2022)
4. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
6. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: Proc. International Conference on Machine Learning (ICML). pp. 1180–1189. PMLR (2015)
7. Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., Lee, S.J.: Note: Robust continual test-time adaptation against temporal correlation. In: Proc. Neural Information Processing Systems (NeurIPS). vol. 35, pp. 27253–27266 (2022)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
9. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: Proc. International Conference on Learning Representations (ICLR) (2019)
10. Hu, X., Uzunbas, G., Chen, S., Wang, R., Shah, A., Nevatia, R., Lim, S.N.: MixNorm: Test-Time Adaptation Through Online Normalization Estimation (arXiv:2110.11478) (Oct 2021)
11. Ioffe, S.: Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: Proc. Neural Information Processing Systems (NeurIPS). vol. 30 (2017)
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. International Conference on Machine Learning (ICML) (2015)
13. Iwasawa, Y., Matsuo, Y.: Test-time classifier adjustment module for model-agnostic domain generalization. In: Proc. Neural Information Processing Systems (NeurIPS). vol. 34 (2021)
14. Jang, M., Chung, S.Y., Chung, H.W.: Test-time adaptation via self-training with nearest neighbor information. In: Proc. International Conference on Learning Representations (ICLR) (2023)
15. Kang, J., Kim, N., Kwon, D., Ok, J., Kwak, S.: Leveraging proxy of training data for test-time adaptation. In: Proc. International Conference on Machine Learning (ICML) (2023)
16. Khurana, A., Paul, S., Rai, P., Biswas, S., Aggarwal, G.: Sita: Single image test-time adaptation. arXiv preprint arXiv:2112.02355 (2021)

17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. vol. 114, pp. 3521–3526. National Acad Sciences (2017)
18. Lee, Y., Chen, A.S., Tajwar, F., Kumar, A., Yao, H., Liang, P., Finn, C.: Surgical fine-tuning improves adaptation to distribution shifts. In: Proc. International Conference on Learning Representations (ICLR) (2023)
19. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 5542–5550 (2017)
20. Lim, H., Kim, B., Choo, J., Choi, S.: Ttn: A domain-shift aware batch normalization in test-time adaptation. In: Proc. International Conference on Learning Representations (ICLR) (2023)
21. Liu, Y., Kothari, P., van Delft, B., Bellot-Gurlet, B., Mordan, T., Alahi, A.: Ttt++: When does self-supervised test-time training fail or thrive? In: Proc. Neural Information Processing Systems (NeurIPS). vol. 34 (2021)
22. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11976–11986 (2022)
23. Nado, Z., Padhy, S., Sculley, D., D’Amour, A., Lakshminarayanan, B., Snoek, J.: Evaluating prediction-time batch normalization for robustness under covariate shift (arXiv:2006.10963) (Jan 2021)
24. Neuhold, G., Ollmann, T., Rota Bulò, S., Kotschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 4990–4999 (2017)
25. Nguyen, A.T., Nguyen-Tang, T., Lim, S.N., Torr, P.H.: Tipi: Test time adaptation with transformation invariance. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 24162–24171 (2023)
26. Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., Tan, M.: Efficient test-time model adaptation without forgetting. In: Proc. International Conference on Machine Learning (ICML). pp. 16888–16905. PMLR (2022)
27. Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., Tan, M.: Towards stable test-time adaptation in dynamic wild world. In: Proc. International Conference on Learning Representations (ICLR) (2023)
28. Poynton, C.: Digital video and HD: Algorithms and Interfaces. Elsevier (2012)
29. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: Proc. European Conference on Computer Vision (ECCV). pp. 102–118. Springer (2016)
30. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3234–3243 (2016)
31. Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., Bethge, M.: Improving robustness against common corruptions by covariate shift adaptation. In: Proc. Neural Information Processing Systems (NeurIPS). vol. 33, pp. 11539–11551 (2020)
32. Song, J., Lee, J., Kweon, I.S., Choi, S.: Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11920–11929 (2023)

33. Su, Y., Xu, X., Jia, K.: Revisiting realistic test-time training: Sequential inference and adaptation by anchored clustering. In: Proc. Neural Information Processing Systems (NeurIPS) (2022)
34. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: Proc. European Conference on Computer Vision (ECCV). pp. 443–450. Springer (2016)
35. Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., Hardt, M.: Test-time training with self-supervision for generalization under distribution shifts. In: Proc. International Conference on Machine Learning (ICML). pp. 9229–9248 (2020)
36. Tomar, D., Vray, G., Bozorgtabar, B., Thiran, J.P.: TeSLA: Test-time self-learning with automatic adversarial augmentation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20341–20350 (2023)
37. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: Proc. International Conference on Learning Representations (ICLR) (2021)
38. Wang, Q., Fink, O., Van Gool, L., Dai, D.: Continual test-time domain adaptation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7201–7211 (2022)
39. Wang, Z., Bai, Y., Zhou, Y., Xie, C.: Can cnns be more robust than transformers? In: Proc. International Conference on Learning Representations (ICLR) (2023)
40. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
41. You, F., Li, J., Zhao, Z.: Test-time batch statistics calibration for covariate shift. arXiv preprint arXiv:2110.04065 (2021)
42. Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2636–2645 (2020)
43. Yuan, L., Xie, B., Li, S.: Robust test-time adaptation in dynamic scenarios. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15922–15932 (2023)
44. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
45. Zhang, J., Qi, L., Shi, Y., Gao, Y.: Domainadaptor: A novel approach to test-time adaptation. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 18971–18981 (2023)
46. Zhang, M., Levine, S., Finn, C.: Memo: Test time robustness via adaptation and augmentation. In: Proc. Neural Information Processing Systems (NeurIPS). vol. 35, pp. 38629–38642 (2022)
47. Zhang, Y., Wang, X., Jin, K., Yuan, K., Zhang, Z., Wang, L., Jin, R., Tan, T.: Adanpc: Exploring non-parametric classifier for test-time adaptation. In: Proc. International Conference on Machine Learning (ICML). pp. 41647–41676. PMLR (2023)
48. Zhao, B., Chen, C., Xia, S.T.: DELTA: DEGRADATION-FREE FULLY TEST-TIME ADAPTATION. In: Proc. International Conference on Learning Representations (ICLR) (2023)
49. Zhao, H., Liu, Y., Alahi, A., Lin, T.: On pitfalls of test-time adaptation. In: Proc. International Conference on Machine Learning (ICML) (2023)
50. Zou, Y., Zhang, Z., Li, C.L., Zhang, H., Pfister, T., Huang, J.B.: Learning instance-specific adaptation for cross-domain segmentation. In: Proc. European Conference on Computer Vision (ECCV). pp. 459–476. Springer (2022)