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

Supplementary material

Juwon Kang[1], Nayeong Kim[2], Jungseul Ok[2], and Suha Kwak[2]

[1] GENGENAI, South Korea
[2] Pohang University of Science and Technology (POSTECH), South Korea
http://cvlab.postech.ac.kr/research/MemBN

A Algorithm

Overall process of MemBN is depicted in Algorithm 1.

Algorithm 1 Overall procedure of **MemBN**

Define PUSH(e, $\mathcal{M}$): push element e to queue $\mathcal{M}$ and pop the oldest element in queue $\mathcal{M}$ if exceeds the limit
Input: capacity of memory queue N, the number of BN layer L,
statistics memory queues $\mathcal{M}_{\boldsymbol{\mu}}^1, ..., \mathcal{M}_{\boldsymbol{\mu}}^L$, $\mathcal{M}_{\boldsymbol{\sigma}}^1, ..., \mathcal{M}_{\boldsymbol{\sigma}}^L$
for $l \leftarrow 1$ to L **do**
  Compute in-batch statistics $\boldsymbol{\mu}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{in}}$ with input $\mathbf{X}^l$ at l-th BN layer
  If detect new domain by Eq. (6)
    then reset $\mathcal{M}_{\boldsymbol{\mu}}^1, ..., \mathcal{M}_{\boldsymbol{\mu}}^L$, $\mathcal{M}_{\boldsymbol{\sigma}}^1, ..., \mathcal{M}_{\boldsymbol{\sigma}}^L$
  PUSH($\boldsymbol{\mu}_{\text{in}}$, $\mathcal{M}_{\boldsymbol{\mu}}^l$), PUSH($\boldsymbol{\sigma}_{\text{in}}$, $\mathcal{M}_{\boldsymbol{\sigma}}^l$)
  Compute average memory statistics $\boldsymbol{\mu}_{\text{mem}}, \boldsymbol{\sigma}_{\text{mem}}^2$ by Eq. (4) and (5)
  Compute adaptive weight α by Eq. (7)
  Compute normalization statistics $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}^2$ by Eq. (8) and (9)
  $\tilde{\mathbf{X}}^l = \text{MemBN}(\mathbf{X}^l; \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}, \mathbf{a}, \mathbf{b})$
end for

B Implementation details

For the classification task, we follow the default optimizer of other optimization-based TTA methods when combined with each of them. We conduct a grid search for the learning rate with values of 1e-4, 1e-5, 1e-6, and 1e-7, and use a constant learning rate without scheduling. For the semantic segmentation task, We conduct a grid search for the learning rate with values of 1e-4 and 1e-5, and keep the other hyperparameters consistent with TTN [3].

Table S1: Single domain adaptation on ImageNet-C. Error rate (↓) averaged over 15 corruption types with severity level 5 is reported using ResNet-50 backbone for each test batch size. We mark the best and second-best performance in **bold** and underline, respectively.

Method	Test batch size					Avg. Error
	64	16	4	2	1	
ETA [5]	51.90	56.37	79.89	93.20	99.86	76.24
ETA + TIPI [4]	**51.00**	-	-	75.40	-	-
ETA + **Ours (MemBN)**	51.03	**51.87**	**55.97**	**59.43**	**67.64**	**57.19**

Table S2: Teomporally-correlated (non-i.i.d.) domain adaptation on ImageNet-C. Error rate (↓) averaged over 15 corruption types with severity level 5 is reported using ResNet-50 backbone for each test batch size.

Method	Test batch size					Avg. Error
	64	16	4	2	1	
DELTA [9]	59.78	73.24	88.13	93.34	96.51	82.20
Tent + **Ours (MemBN)**	**57.43**	**64.09**	**67.51**	**70.27**	**74.00**	**66.66**

C Additional experiments

Applying our method to ETA [5]. To demonstrate the applicability of our method to different TTA approaches, we incorporated it into ETA [5], which selectively performs test-time entropy minimization for reliable and non-redundant samples. Simultaneously, our method was compared with TIPI [4], which utilizes augmented inputs for enhancing robustness of TTA for small input batches. The experiments were evaluated in a single-domain TTA setting on ImageNet-C with the highest severity level. Table S1 illustrates that our method consistently enhanced the robustness of TTA across a wide range of batch sizes. Compared to TIPI, our method achieved comparable performance at batch size of 64 and significantly surpassed it at batch size of 2 (results for other batch sizes are omitted as they were not reported in the paper).

Comparison to DELTA [9] in non-i.i.d. TTA on ImageNet-C. We conducted additional experiments with latest TTA method DELTA [9] in non-i.i.d. TTA on ImageNet-C. Following the experimental setting of DELTA, our method was evaluated using ResNet50 and with a concentration $\delta = 0.1$ for dependently sampled test stream. As shown in Table S2, our method consistently outperforms DELTA, with a substantial margin at small match sizes. This is because, although DELTA aggregates previous test statistics like ours, it employs an EMA update that substantially degrades performance when input batches are small, as already confirmed in Table 6 of the main paper.

Temporally correlated (non-i.i.d.) TTA on CIFAR-C [2]. We conducted additional experiments to identify the robustness of our method under a temporally correlated TTA setting on CIFAR10-C and CIFAR100-C. Following the evaluation protocol of NOTE [1], we simulated non-i.i.d. test streams using a

Table S3: Temporally correlated (non-i.i.d.) domain adaptation on CIFAR10-C and CIFAR100-C. Error rate (↓) averaged over 15 corruptions with severity level 5 using ResNet-18 for each test batch size. We mark the best and second-best performance in **bold** and underline, respectively. † denotes using prediction balanced reservoir sampling (PBRS) proposed in NOTE [1] to simulate i.i.d. data stream. Grey color indicates our default setting.

Method	CIFAR10-C							CIFAR100-C						
	200	64	16	4	2	1	**Avg.**	200	64	16	4	2	1	**Avg.**
Source (CBN)	42.56	42.56	42.56	42.56	42.56	42.56	42.56	66.60	66.60	66.60	66.60	66.60	66.60	66.60
TENT [6]	68.65	72.74	74.51	74.52	73.38	83.44	74.54	53.08	64.25	79.20	86.47	97.27	99.03	79.88
TENT + NOTE [1]	**20.79**	**20.71**	**19.00**	31.51	49.17	65.59	29.08	48.41	47.23	**43.53**	**43.25**	46.99	58.90	48.05
TENT + **Ours (MemBN)**	26.46	26.06	25.51	29.56	36.79	41.69	31.01	**43.95**	**44.13**	44.58	45.58	46.45	**48.77**	**45.58**
TENT + **Ours (MemBN)**†	24.90	22.80	21.80	**23.30**	**29.10**	**39.10**	**26.83**	47.03	44.70	44.65	45.13	**45.83**	54.00	46.89

Table S4: Non-i.i.d. and continual domain adaptation on CIFAR10-C and CIFAR100-C. Error rate (↓) averaged over 15 corruptions with severity level 5 using WideResNet28 and ResNet29 on CIFAR10-C and CIFAR100-C, respectively. Dirichlet concentration parameter δ is set to 0.1 for non-i.i.d. test stream.

Method	CIFAR10-C						CIFAR100-C					
	64	16	4	2	1	**Avg.**	64	16	4	2	1	**Avg.**
Source (CBN)	43.50	43.50	**43.50**	**43.50**	**43.50**	43.50	46.40	46.40	46.40	46.40	46.40	46.40
TENT [6]	75.52	78.77	79.65	78.57	90.00	80.50	51.82	71.93	84.77	90.15	98.91	79.52
RoTTA	**24.68**	43.02	71.21	78.21	81.49	59.72	35.00	36.35	47.28	63.34	79.21	52.24
TENT + **Ours (MemBN)**	26.34	**32.75**	44.38	49.61	52.63	**41.14**	**32.97**	**33.79**	**34.07**	**34.85**	**37.28**	**34.59**

Dirichlet distribution with a concentration parameter, δ, set to 0.1. NOTE proposes a robust TTA approach for non-i.i.d. test streams by integrating a memory bank to save input data for Prediction-Balanced Reservoir Sampling (PBRS) along with an instance-aware batch normalization. We evaluated the performance of our method in the presence and absence of PBRS. Table S3 presents the results on CIFAR10-C and CIFAR100-C using WideResNet-40-2 backbone. Without PBRS, our method outperformed NOTE on CIFAR100-C, albeit with lower performance on CIFAR10-C. However, when PBRS was integrated, our method achieved the best performance on CIFAR10-C. This indicates that the impact of PBRS was pronounced in settings with a smaller number of classes. Conversely, in scenarios with a larger number of classes (*e.g.*, ImageNet), our method demonstrated effectiveness even without the support of PBRS.

Temporally correlated (non-i.i.d.) and continual TTA on CIFAR-C [2]. To compare with the latest state-of-the-art method in more realistic setting, we conducted additional experiments with RoTTA [] in non-i.i.d. setting (concentration $\delta = 0.1$) of continual TTA on CIFAR10-C. We set the memory bank size of RoTTA to the test batch size for fairness in memory constraint. Following the previous work, we used WideResNet28 and ResNext29 on CIFAR10-C and CIFAR100-C, respectively. As shown in Table S4, our method outperforms RoTTA across almost all batch sizes, thanks not only to the use of average memory statistics but also to the domain shift indicator that promptly responds to domain shifts.

Table S5: Single domain adaptation on PACS. Error rate (↓) averaged over four domains (art painting, cartoon, photo, and sketch) using ResNet18 for each test batch size. We mark the best performance in **bold**.

Method	Test batch size					Avg. Error
	64	16	4	2	1	
Source	20.56	20.56	20.56	20.56	20.56	20.56
TENT	16.44	21.69	31.91	43.39	83.46	39.38
DomainAdaptor [8]	16.64	16.56	17.41	19.90	31.27	20.36
TENT + **MemBN (Ours)**	**15.99**	**15.13**	**15.70**	**15.93**	**18.45**	**16.24**

Table S6: Single domain adaptation on ImageNet-R. Error rate (↓) is measured using ResNet50 for each test batch size. We mark the best performance in **bold**.

Method	Test batch size					Avg. Error
	64	16	4	2	1	
Source	63.83	63.83	63.83	63.83	63.83	63.83
TENT	**58.14**	61.21	69.40	83.51	99.44	74.34
TENT + **MemBN (Ours)**	58.59	**59.24**	**60.27**	**60.52**	**63.44**	**60.41**

Evaluation on other datasets. To verify the effectiveness of our method on different datasets, we evaluated single domain TTA performance on PACS and ImageNet-R. For the experiments on PACS, we reported the average error rate of each TTA performance towards four domains (art painting, cartoon, photo, and sketch). As shown in Tables S5 and S6, our method consistently achieves high performance. Additionally, Table S5 compares our method with DomainAdaptor [8] that mixes test in-batch statistics with source statistics. The results suggest that the simple mixing strategy of DomainAdapter is vulnerable to small batch sizes, while our method remains effective.

Single TTA on ImageNet-C with other models. Table S7 shows the performance of Tent and ours using WideResNet101 or EfficientNet-B4 as backbone; the result suggests that our method well generalizes to diverse architectures.

D Additional analysis

Analysis on GPU memory usage. To identify the marginal increase in space complexity of our method's statistics memory, we conducted a GPU memory usage comparison among TBN, NOTE, and MemBN. As illustrated in Fig. S1, our method exhibited a mere 4 MB additional usage in batch size 64 and only 2 MB in batch size 1 when compared to TBN. Notably, our method demonstrated lower memory consumption compared to NOTE, a state-of-the-art method that utilizes a class-balanced memory bank.

Ablation study on our weighting policy. In order to show the superiority of adaptive weighting policy, a comparative experiment with NOTE was conducted. Unlike NOTE that employs a fixed mixing ratio between source and target statistics for out-of-distribution samples, our method uses adaptive weights based on

Table S7: Single domain adaptation on ImageNet-C. Error rate (↓) averaged over 15 corruption types with severity level 5 for each test batch size. We mark the best performance in **bold**.

Model	Method	Test batch size					Avg. Error
		64	16	4	2	1	
WideResNet-101	Source	74.52	74.52	74.52	74.52	74.52	74.52
	TENT	**53.68**	63.67	85.37	97.57	99.89	80.03
	TENT + **MemBN (Ours)**	55.14	**56.45**	**61.14**	**62.79**	**63.66**	**59.84**
EfficientNet-B4	Source	64.57	64.57	64.57	64.57	64.57	64.57
	TENT	65.15	68.75	80.98	92.95	99.56	81.48
	TENT + **MemBN (Ours)**	**57.34**	**56.60**	**57.13**	**56.36**	**56.43**	**56.79**

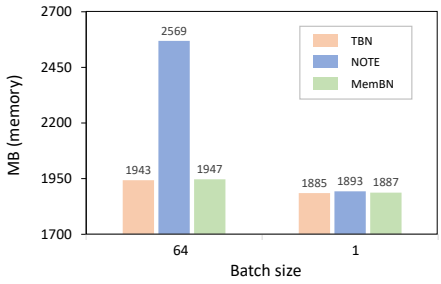

Fig. S1: GPU memory usage of each method. This space complexity is measured on CIFAR10-C using WideResNet-40-2 model.

Table S8: Comparing the weighting policies of ours and NOTE. Error rate (↓) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 in single domain TTA setting on CIFAR10-C.

Method	Test batch size						Avg.
	200	64	16	4	2	1	
Ours	**12.68**	**12.41**	**12.16**	**12.30**	**12.46**	**12.88**	**12.48**
Ours with NOTE weighting policy	13.90	13.86	13.88	13.90	13.92	13.96	13.90

the degree of domain shift. To validate the efficacy of our policy, we evaluated the performance of our framework using the policy of NOTE that combines source and average memory statistics, instead of MemBN. Table S8 demonstrates superiority of our policy.

Ablation study on interpolation method. In Table S9 and Table S10, we present a detailed comparative analysis of different interpolation techniques. In both tables, the first three rows in each table represent performance using only CBN, TBN, and average memory statistics, respectively. Row 4 in each table shows the performance when taking the average between CBN and TBN, while Row 5 reports the performance of TTN, which adopts a channel-wise optimized weights between CBN and TBN. For interpolation between CBN and average memory statistics, Row 6 presents the case of using their mean, and Row 7

Table S9: Ablation study on interpolation strategy on CIFAR10-C in single-domain TTA. Error rate (↓) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 for each test batch size. Grey color indicates our default setting.

#	Interpolation	Mixing b/w		Test batch size						Avg.
				200	64	16	4	2	1	
1	No interp.	CBN	-	18.27	18.27	18.27	18.27	18.27	18.27	18.27
2	No interp.	-	TBN	14.47	15.02	17.10	26.28	35.65	90.00	33.09
3	No interp.	-	Avg. of mem	14.29	14.29	14.35	14.66	14.88	15.95	14.74
4	Constant (Mean)	CBN	TBN	12.07	12.21	13.05	16.72	20.04	21.26	15.89
5	Optimized (Channel-wise)	CBN	TBN	**11.67**	**11.80**	**12.13**	13.93	15.83	17.99	13.89
6	Constant (Mean)	CBN	Avg. of mem	12.26	12.25	12.27	**12.32**	**12.38**	**12.73**	**12.37**
7	**Adaptive weight (Layer-wise)**	CBN	Avg. of mem	12.68	12.41	12.16	**12.30**	12.46	12.88	12.48

Table S10: Ablation study on interpolation strategy in DG benchmarks for semantic segmentation. mIoU(↑) on four unseen domains with test batch size of 2 using ResNet-50 based DeepLabV3+. We mark the best and second-best performance in **bold** and underline, respectively. Grey color indicates our default setting.

#	Interpolation	Mixing b/w		BDD-100K	Mapillary	GTA5	SYNTHIA	Cityscapes	Avg.
1	No interp.	CBN	-	43.50	54.37	43.71	22.78	**77.51**	48.37
2	No interp.	-	TBN	43.30	47.80	43.57	25.92	72.93	46.70
3	No interp.	-	Avg. of mem	49.93	51.83	44.70	26.89	77.09	50.09
4	Constant(Mean)	CBN	TBN	47.54	52.30	44.27	26.45	74.24	48.96
5	Optimized (Channel-wise)	CBN	TBN	47.89	57.84	46.18	27.29	75.04	50.85
6	Constant (Mean)	CBN	Avg. of mem	**50.62**	54.42	**47.08**	27.44	77.22	51.36
7	**Adaptive weight (Layer-wise)**	CBN	Avg. of mem	50.34	**57.87**	46.39	**27.90**	77.23	**51.95**

presents the performance of our proposed method utilizing layer-wise adaptive weighting. Across both tables, we observed that without interpolation, using average memory statistics (Row 3) enhanced performance compared to either CBN (Row 1) or TBN (Row 2). Additionally, comparisons between Rows 4 and 6, and Rows 5 and 7, clearly indicate that interpolating with average memory statistics is more effective than with TBN. Notably, our method outperformed TTN despite utilizing more coarse-grained weighting and not requiring the optimization of interpolation weights before testing. On common image corruption benchmarks, Rows 6 and 7 exhibited similar performance in Table S9. However, in Table S10, which covers various domain generalization benchmarks, adaptive weighting approach (Row 7) achieved the best or second best performance and surpassed the others in terms of average performance, highlighting its efficacy in diverse testing environments.

Comparison with online MLE for estimating feature statistics. To validate the effectiveness of our statistics estimation, we conducted a comparative experiment with the online Maximum Likelihood Estimation (MLE) for the parameters of a Gaussian distribution. Both methods aim to estimate global feature statistics. However, our method aggregates the most recent statistics of current domain, whereas online MLE updates global statistics with current samples under the assumption of a Gaussian distribution. Table S11 shows the superiority

Table S11: Comparing the statistics estimation method with ours and online MLE. Error rate (↓) averaged over 15 corruptions with severity level 5 in the continual TTA setting on CIFAR10-C.

Method	Test batch size						Avg.
	200	64	16	4	2	1	
Ours	**12.33**	**12.13**	**12.02**	**12.10**	**12.19**	**12.29**	**12.18**
Online MLE	15.66	15.71	15.71	15.67	15.67	15.67	15.68

Table S12: Ablation study on the memory size of our method in continual TTA on CIFAR10-C and CIFAR100-C. Error rate (↓) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 for each batch size. Grey color indicates our default setting.

Method	CIFAR10-C							CIFAR100-C						
	200	64	16	4	2	1	Avg.	200	64	16	4	2	1	Avg.
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	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
MemBN(size=8)	12.49	12.24	12.15	12.49	12.94	13.77	12.68	**36.87**	**36.65**	36.78	37.63	38.38	39.99	37.72
MemBN(size=32)	12.69	12.37	12.08	12.15	12.27	12.50	12.34	37.61	36.94	36.73	36.96	37.24	37.66	37.19
MemBN(size=128)	**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**
MemBN(size=512)	12.76	12.18	12.05	**12.10**	**12.10**	**12.18**	12.23	37.28	36.67	36.64	36.78	37.04	37.03	36.91

of our method over online MLE. Our method is more robust against domain shift in testing as it detects new domains and resets memory by referencing recent statistics. In contrast, online MLE struggles in new domains due to the adverse impact of prior global statistics, which leads to severe performance drop in the continual TTA setting.

Sensitivity analysis on memory size. For a sensitivity analysis on the size of statistics memory in MemBN, we conducted experiments in the continual TTA setting [7] with memory sizes of 8, 32, 128, and 512. Table S12 demonstrates that our method maintained stable performance across memory sizes ranging from 32 to 512, indicating that our method is not sensitive to memory size. Interestingly, even with a smaller memory size of 32, which is lower than our default setting of 128, the performance remained comparable to that of the default setting. This suggests the potential for further reducing additional memory overhead by using even smaller memory sizes.

Sensitivity analysis on γ. To analyze the sensitivity of the parameter γ for calculating adaptive weight α, we conducted experiments in continual TTA with γ set at 0.1, 0.3, 0.5, 0.7, and 0.9. Table S13 shows that our method achieved stable and high performance when γ is set within the range of 0.3 to 0.5.

E Derivation of Eq. (4) and (5)

Given that $X_1, X_2, ..., X_n$ are independent random variables with mean μ_i and variance σ_i^2 for each X_i, let's define the overall feature X as follows (for simplicity,

Table S13: Ablation study on the parameter γ of our method in continual TTA on CIFAR10-C and CIFAR100-C. Error rate (↓) averaged over 15 corruptions with severity level 5 using WideResNet-40-2 for each batch size. Grey color indicates our default setting.

Method	CIFAR10-C							CIFAR100-C						
	200	64	16	4	2	1	**Avg.**	200	64	16	4	2	1	**Avg.**
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	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
MemBN($\gamma = 0.1$)	12.62	12.56	13.00	13.56	13.77	13.92	13.24	38.01	37.99	38.84	39.74	40.20	40.55	39.22
MemBN($\gamma = 0.3$)	**12.33**	**12.12**	**12.02**	**12.10**	12.19	12.29	**12.18**	**36.91**	**36.66**	**36.59**	36.85	36.79	36.99	**36.80**
MemBN($\gamma = 0.5$)	13.00	12.68	12.38	12.18	**12.07**	**12.05**	12.39	38.12	36.88	**36.59**	**36.61**	**36.68**	**36.94**	36.97
MemBN($\gamma = 0.7$)	13.54	13.11	12.78	12.46	12.33	12.35	12.76	38.10	37.38	37.05	36.85	36.98	37.27	37.27
MemBN($\gamma = 0.9$)	13.77	13.43	13.16	12.83	12.67	12.73	13.10	39.01	37.95	37.42	37.24	37.36	37.77	37.79

we omit the channel index in the following expressions):

$$X = \begin{cases} X_1, & \text{with probability } \frac{1}{N} \\ X_2, & \text{with probability } \frac{1}{N} \\ \vdots & \\ X_N, & \text{with probability } \frac{1}{N} \end{cases}$$

Then, the mean and variance of X are computed as follows:

$$\begin{aligned} \mathbb{E}(X) &= \frac{1}{N}\sum_{i}^{N}\mathbb{E}(X_i) = \frac{1}{N}\sum_{i}^{N}\mu_i, \\ \mathrm{Var}(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[X_i^2] - \frac{1}{N^2}\left(\sum_{i=1}^{N}\mathbb{E}[X_i]\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(\sigma_i^2 + \mu_i^2) - \frac{1}{N^2}\left(\sum_{i=1}^{N}\mu_i\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\sigma_i^2 + \frac{1}{N}\sum_{i=1}^{N}\mu_i^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N}\mu_i\right)^2. \end{aligned}$$

F Discussion

In this work, we introduced a fully test-time adaptation method focusing on the architecture using batch normalization. While our method shows promising improvement without the need for any training before testing, we anticipate that focusing on learning more fine-grained weights for average memory statistics through methods such as meta learning will yield more significant improvements in TTA performance.

References

1. 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)
2. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: Proc. International Conference on Learning Representations (ICLR) (2019)
3. 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)
4. 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)
5. 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)
6. 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)
7. 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)
8. 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)
9. Zhao, B., Chen, C., Xia, S.T.: DELTA: DEGRADATION-FREE FULLY TEST-TIME ADAPTATION. In: Proc. International Conference on Learning Representations (ICLR) (2023)