Supplementary Material for Layer-wise Auto-Weighting for Non-Stationary Test-Time Adaptation

Junyoung Park[1] Jin Kim[1] Hyeongjun Kwon[1] Ilhoon Yoon[1] Kwanghoon Sohn[1,2*]
[1]Yonsei University [2]Korea Institute of Science and Technology (KIST)
{jun_yonsei, kimjin928, kwonjunn01, ilhoon231, khsohn}@yonsei.ac.kr

In this document, we first complement the quantitative comparisons with encoder type variations (Sec. A). Moreover, we provide ablation studies (Sec. B) including visualization of diagonals of Fisher Information Matrix (FIM) of our proposed method. Our code is available at https://github.com/junia3/LayerwiseTTA.

A. Additional Results

In this section, we conducted a comparative analysis between our proposed method and previous Continual Test-Time Adaptation (CTTA) approaches, including TENT [9], BN-1 [7], AdaContrast [1], and CoTTA [10].

A.1. Comparison with model variation

To validate the adaptive applicability of our proposed method across various encoder types, we conducted additional experiments. We used several distinct encoder configurations for evaluation. In the CIFAR-C benchmark, we used ResNext-29, WideResNet-28, WideResNet-40 from [2], and the ResNet-50 model from [6]. For simplicity, we refer to these models as RNXT29, WRN28, WRN40, and RN50, respectively. However, the ImageNet-C [3] pre-trained model parameters for RNXT29 and WRN are not available in the previous benchmark [2]. Therefore, we conducted experiments using pre-trained parameters from other benchmarks for ResNext-50 [11] and WideResNet-50 [12] models. To simplify, we refer to these models as RNXT50 and WRN50, respectively. Although the pre-trained ResNet-50 [4] for ImageNet-C differs from the one in [6], we will use the term RN50 for convenience.

Comparison on CIFAR-10C. Table 1 presents the average classification errors on CIFAR-10C [5] with the CTTA setting using four distinct encoder configurations. Our method demonstrates an average mean error of 9.8%, 15.7%, 11.0%, and 12.8% in each model framework, surpassing all previous source-free TTA methods across all encoder architectures. Previously proposed methods, such as TENT [9], CoTTA [10] and AdaContrast [1], exhibited favorable performance in specific network structures but struggled to achieve stable adaptation performance across all network architectures. In contrast, our method consistently delivers strong performance, regardless of the network structure, confirming its effectiveness.

Table 1. Classification mean error (%) for the CIFAR-10C online CTTA task on the highest corruption severity level 5. We report the performance of each method averaged over 5 runs.

Method	RNXT29 [2]	WRN28 [2]	WRN40 [2]	RN50 [6]
Source	18.0	43.5	18.3	48.8
BN-1 [7]	13.3	20.4	14.6	16.1
TENT-cont. [9]	14.8	20.7	12.5	14.8
CoTTA [10]	11.0	16.2	12.7	13.1
AdaContrast [1]	11.0	18.5	11.9	14.5
Ours	**9.8**	**15.7**	**11.0**	**12.8**

Comparison on CIFAR-100C. Additionally, we conducted evaluations on CIFAR-100C [5] using the CTTA setting with our proposed method. In Table 2, we can observe similar issues as with previous approaches in terms of model variations. TENT [9] shows improved performance in WRN40 and RN50 except for RNXT29 than BN-1, which simply updates batch normalization statistics. Regarding CoTTA [10] and AdaContrast [1], they show improvement in RNX29 and RN50 but exhibit performance degradation in WRN40. Our method demonstrates outstanding performance across all variations, confirming

Table 2. Classification mean error (%) for the CIFAR-100C online CTTA task on the highest corruption severity level 5. We report the performance of each method averaged over 5 runs.

Method	RNXT29 [2]	WRN40 [2]	RN50 [6]
Source	46.4	46.8	73.8
BN-1 [7]	35.4	39.3	43.7
TENT-cont. [9]	60.9	36.9	44.2
CoTTA [10]	32.5	38.2	37.6
AdaContrast [1]	33.4	37.1	41.3
Ours	**30.9**	**35.0**	**36.2**

its model-agnostic usability. It achieves 30.9%, 35.0%, and 36.2% with each encoder.

Comparison on ImageNet-C. We also conducted evaluations on ImageNet-C [3] with CTTA setting using our proposed method. Although BN-1 [7], which does not require optimization, improved performance compared to the source approach, its performance significantly declined compared to optimization-based approaches. In this experiment, TENT [9] outperforms CoTTA [10] and AdaContrast [1], both of which update the entire set of parameters. TENT reports 62.6%, 58.7%, and 57.7% accuracy in each architecture. Nevertheless, we demonstrate that our method is superior to others in all model frameworks. Our proposed method achieves accuracy rates of 60.1%, 57.5%, and 56.4%, improving by 2.6%, 2.3%, and 1.3% over TENT, respectively.

Table 3. Classification mean error (%) for the ImageNet-C online CTTA task on the highest corruption severity level 5. We report the performance of each method averaged over 5 runs.

Method	RN50 [4]	RNXT50 [11]	WRN50 [12]
Source	82.0	78.9	78.9
BN-1 [7]	68.6	67.1	66.2
TENT-cont. [9]	62.6	58.7	57.7
CoTTA [10]	62.7	59.8	57.9
AdaContrast [1]	65.5	63.1	63.3
Ours	**60.1**	**57.5**	**56.4**

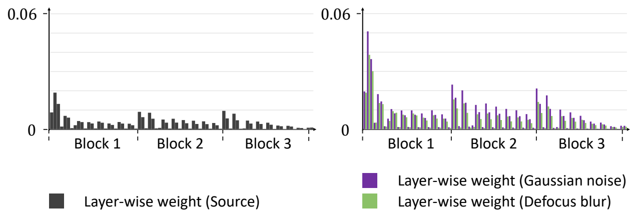

Figure 1. Layer-wise weights comparison between the source and target domains.

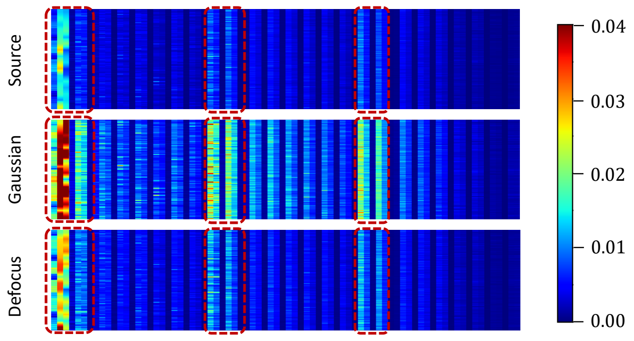

Figure 2. **Diagonal of the FIM per layer** with the source domain, gaussian noise domain and defocus blur domain in the CIFAR-10C dataset. The x- and y-axes represent the layer index and the diagonal values of the layer-wise FIM, respectively. This demonstrates that our method is robust to any kind of domain distribution.

B. Ablation study

B.1. Ablations with domain-level FIM

In all experimental settings, we used domain-level Fisher Information Matrix (FIM), which accumulates information from continuous domain samples, rather than solely relying on temporal FIM. To verify the effectiveness of our method, we further introduced a hyperparameter γ for the domain-level FIM in Eq. (4) of our main paper, inspired by [8]:

$$\tilde{I}^l_t = \gamma \tilde{I}^l_{t-1} + I^l_t. \tag{1}$$

In Table 4, we compared the performance of CTTA by varying the γ of the regulated version of FIM from 0 to 1. Since γ is set to be less than 1, we can adjust our domain-level FIM from the batch level to the domain level. For CIFAR-10C [5], our method achieved an accuracy of 15.74%, representing a 0.62% improvement over the performance when $\lambda = 0$. Our method exhibits an improvement of 0.86% for CIFAR-100C [5] and 0.87% for ImageNet-C [3]. This demonstrates that our domain-level FIM outperforms the FIM which focus on short intervals.

Table 4. Mean classification error (%) with varing γ.

γ	0	0.3	0.6	0.9	**Ours**
CIFAR-10C	16.36	16.24	16.01	15.95	**15.74**
CIFAR-100C	31.77	31.74	31.70	31.58	**30.91**
ImageNet-C	60.94	60.84	60.79	60.67	**60.07**

B.2. Ablations with exponential min-max scaler

In addition to our main paper, we show additional ablations on the hyperparameter τ of the exponential min-max scaler in Eq. (7) of our main paper. Table 5 provides additional results for τ values ranging from 0 to 1.2 for each dataset in the CTTA benchmark. When $\tau = 1$, it represents the result obtained by applying simple min-max normalization to the weight importance from domain-level FIM. Conversely, for $\tau = 0$, it assumes that all weight importance values have equal weights of 1. As τ approaches zero, layer-wise weights tend to have relatively uniform values, and the overall performances diverge during the adaptation process.

Table 5. Mean classification error (%) with varying τ.

τ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
CIFAR-10C	89.30	89.35	88.85	86.03	78.28	65.92	39.55	19.96	17.74	16.72	**15.74**	15.97	16.20
CIFAR-100C	37.16	33.98	32.77	32.12	31.83	31.75	**30.91**	31.42	32.16	32.42	32.76	33.03	33.30
ImageNet-C	99.57	99.39	98.72	97.10	94.79	90.95	77.29	69.89	63.44	61.08	**60.07**	61.21	61.65

B.3. Layer-wise learning weights in each domain

In the main paper, we attempted to apply layer-wise learning rates according to the domain shift problem. From this perspective, it should also be noted that if there is no domain shift, the learning weights will assume a minimal value. To demonstrate this, we compare the learning weights from the source domain with those from the target domain. Specifically, for the target domain, we choose gaussian noise and defocus blur for comparison. In Figure 1, we compare the learning weights measured by the pre-trained model on the CIFAR10-to-10C dataset without applying the exponential min-max normalization in Eq. (7) of our main paper. The learning weights for gaussian noise/defocus blur vary by layer depending on the type of corruption, demonstrating that our model can adapt its learning rate to different types of domain shifts. On the other hand, the learning weights in the source domain have a smaller value than gaussian noise/defocus blur domains. Note that, the pre-trained model exhibits a flatter log-likelihood surface for the source data than the target data since the pre-trained parameters are already optimized on the source domain. From this, we conjecture that our method is capable of identifying the importance of layers by referring to the surface information of the log-likelihood.

B.4. Layer-wise diagonal of FIM in each domain

In our proposed method, layer-wise learning weights are calculated as the trace of the FIM. To analyze this, in Figure 2, we visualize the diagonal of the FIM in gaussian noise domain and defocus blur domain. In comparison to the source domain, we demonstrate that the lower diagonal values of the FIM within layer θ^l signify the convergence of that layer, offering advantages in representing a particular domain distribution. Since these diagonal elements reflect the curvature of each layer's distribution with respect to the log-likelihood of model outputs, we can establish that our method efficiently leverages layer sharpness in layer-wise learning. Therefore our method, which selects layers to optimize using Hessian approximation with FIM, can effectively employ auto-weighting to identify layers for preservation or concentrated adaptation.

References

[1] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 1, 2

[2] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS*, 2021. 1

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[5] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1, 2

[6] 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? In *NeurIPS*, 2021. 1

[7] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 1, 2

[8] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018. 2

[9] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 1, 2

[10] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022. 1, 2

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 2

[12] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 1, 2