Supplementary for Towards Open-Set Test-Time Adaptation Utilizing the Wisdom of Crowds in Entropy Minimization

Jungsoo Lee[1,2*] Debasmit Das[1] Jaegul Choo[2] Sungha Choi[1†]

[1]Qualcomm AI Research[‡] [2]KAIST

[1]{jungsool, debadas, sunghac}@qti.qualcomm.com [2]{bebeto, jchoo}@kaist.ac.kr

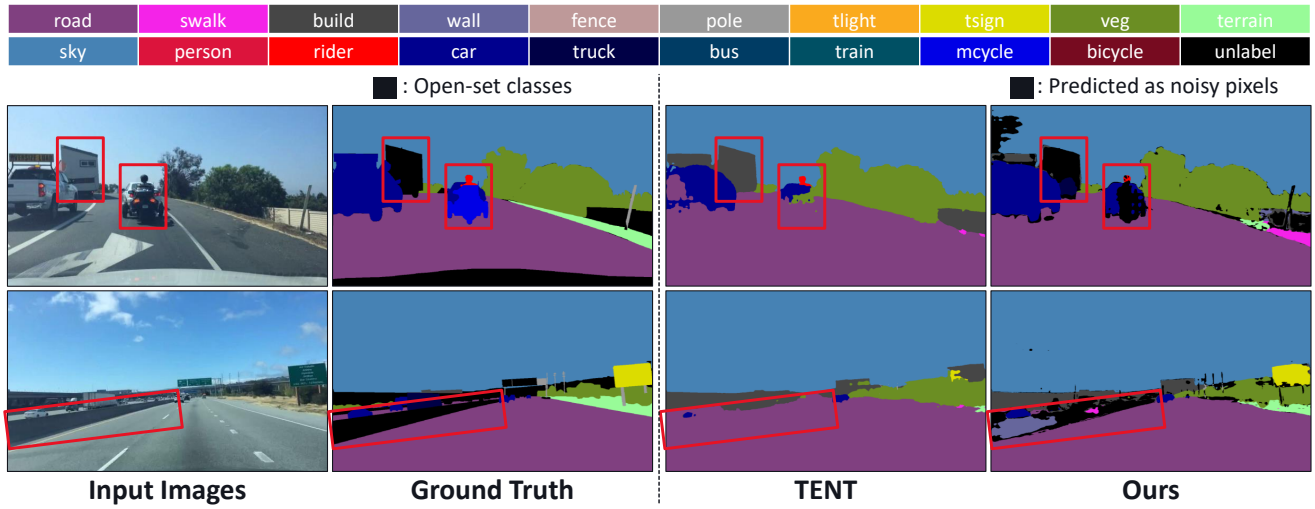

Figure 1: Identifying wrong predictions and open-set samples in test-time adaptation (TTA). During the long-term adaptation, previous models not only show a large performance degradation but also predict the open-set samples as one of the pre-defined classes learned during the training phase. By filtering out noisy ones, both wrong and open-set samples, we can (a) prevent performance degradation and (b) identify unexpected obstacles to prevent accidents immediately. Red boxes indicate the regions of pixels that include misclassified predictions or open-set classes. In the fourth column, on top of the prediction of the model trained through our method, we color pixels that are filtered out by our method as black.

A. Further Analysis on Semantic Segmentation

Fig. 1 shows how filtering out noisy samples is important in semantic segmentation. As mentioned in the main paper, discarding noisy samples is crucial in two aspects: we can 1) prevent significant performance degradation caused by noisy samples and 2) immediately identify unknown objects that could be highly dangerous if not detected. For example, TENT [16] predicts the motorcycle (wrong prediction) or the guardrails (open-set predictions) as roads in the first and the second rows, respectively. When applying TTA in real-world applications (*e.g.,* autonomous driving), such an issue could lead to a serious accident. However, our method effectively identifies them immediately (black pixels in the fourth column), which can prevent such accidents.

Table 1 shows that open-set samples degrade the performance of TTA models in semantic segmentation. For the analysis, we compare the performance of TENT [16] and that of TENT trained without the backpropagation of the open-set pixels. We use the ground truth labels and filter out the open-set pixels. As shown, TENT achieves better performance without the backpropagation of the open-set pixels compared to the original performance. Such a result

Time	$t \longrightarrow$			Average
Method	Cityscapes	BDD-100K	Mapillary	
TENT [16]	46.73	29.59	35.69	37.34
TENT w/o open-set [16]	**47.04**	**31.12**	38.66	38.94
+ Ours	46.76 (+0.03)	30.55 (+0.96)	**43.42 (+7.73)**	**40.24 (+2.90)**

Table 1: Effect of removing open-set samples in semantic segmentation. We filtered out open-set pixels using ground-truth labels for TENT. We observe performance gain compared to the original performance of TENT.

*Work done during an internship at Qualcomm AI Research.

† Corresponding author. ‡ Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Method	ImageNet-C		Average
	Closed	Open	
Source [18]	81.99	81.99	81.99
BN Adapt [12]	68.49	69.65	69.07
TENT [16]	99.71	99.72	99.72
+ Ours	**65.62 (-34.09)**	**67.78 (-31.94)**	**66.70 (-33.02)**
SWR [3]	65.20	68.40	66.80
+ Ours	**64.35 (-0.85)**	**66.33 (-2.07)**	**65.34 (-1.46)**

(a) Error rates after 10 rounds of adaptation.

Method	ImageNet-C		Average
	Closed	Open	
Source [18]	81.99	81.99	81.99
BN Adapt [12]	68.49	69.65	69.07
TENT [16]	95.79	97.53	96.66
+ Ours	**60.82 (-34.97)**	**64.33 (-33.20)**	**62.58 (-34.08)**
SWR [3]	66.59	69.02	67.81
+ Ours	**65.29 (-1.30)**	**66.86 (-2.16)**	**66.08 (-1.73)**

(b) Error rates after 1 round of adaptation.

Table 2: Comparisons on ImageNet-C. We note the performance gain by reduced error rates.

Method	CIFAR-10-C			CIFAR-100-C			Memory	Time
	(a)	(b)	(c)	(a)	(b)	(c)	**(MB)**	**(ms)**
TENT [16]	56.00	45.84	45.84	45.20	42.34	42.34	556	18.38
CoTTA [17]	31.28	75.97	83.19	41.40	94.52	97.43	36442	379.49
TENT + Ours	**20.16**	**14.10**	**14.10**	**33.39**	**38.62**	**38.62**	565	26.62

(a) Image classification

Time	t ⟶			Average
Method	Cityscapes	BDD-100K	Mapillary	
TENT [16]	46.73	29.59	35.69	37.34
CoTTA [17]	41.03	26.42	40.03	33.23
TENT+ Ours	**46.76 (+0.03)**	**30.55 (+0.96)**	**43.42 (+7.73)**	**40.24 (+2.90)**

(b) Semantic Segmentation

Table 3: Comparison between our method and CoTTA [17]. We show the results of our method applied to TENT. We perform better than CoTTA even with a substantially smaller amount of memory usage and time consumption.

again demonstrates that addressing open-set samples is crucial for practical TTA. Note that our approach still outperforms TENT adapted with open-set samples filtered out after a long-term adaptation (*e.g.,* Mapillary). This is mainly due to the fact that our method discards the wrong predictions well in addition to the open-set samples.

B. Comparisons on ImageNet-C

In Table 2, we also verify the effectiveness of our method on a large-scale dataset, ImageNet-C [7]. Due to the fact that experimentation on ImageNet-C is time consuming, we simulate the long-term adaptation with 10 rounds instead of the 50 rounds used in the main paper. We evaluate under continuously changing target domains without resetting the model between each domain. We use the batch size of 64 and the learning rate of 0.00025 with the SGD optimizer [14], following the previous studies [16, 11, 3]. We observe that our method again consistently improves the TTA performance on existing baseline models in closed-set and open-set settings with short-term and long-term adaptation. Regarding SWR [3], we observe a significant performance drop of SWR when utilizing the adapted model of the previous iteration for the regularization. Therefore, we use the source pretrained original model, θ_o, for the regularization. Other hyper-parameters are set as the default values.

C. Comparisons with CoTTA

We also compare our method with CoTTA [17], another seminal work in the continual test-time adaptation. Table 3 compares the performances of image classification and semantic segmentation and the resource costs between CoTTA and our method applied to TENT [16]. As shown, although our method utilizes a significantly smaller amount of memory usage and time consumption, we achieve better performance in both image classification and semantic segmentation. We describe the results in detail.

C.1. Image Classification

We observe that CoTTA [17] shows performance variations depending on the hyper-parameter p_{th}, which is a threshold to decide whether to use ensembled predictions or a single prediction in CoTTA. However, we found it challenging to find adequate p_{th} for CoTTA with the model architecture used in our work (*i.e.,* WideResNet40 [18] for both CIFAR-10-C and CIFAR-100-C). Although the supplementary of CoTTA illustrates how to find p_{th}, we could not obtain identical values by using the architecture used in CoTTA even with the description. Therefore, we report the comparisons between CoTTA and our method with the following experimental setups: a) architectures used in the CoTTA paper (*i.e.,* WideResNet28 [18] for CIFAR-10-C and ResNeXt-29 for CIFAR-100-C) with their default p_{th} values, b) architectures used in our main paper with their

Method	Memory (MB)	Time (ms)
TENT [16]	2714	529
SWR [3]	5969	625
CoTTA [17]	20276	4499
TENT [16] + Ours	3036	685

Table 4: Comparisons on memory usage (MB), and time consumption (ms) on semantic segmentation. We evaluate with DeepLabV3Plus-ResNet-50 [2]. For memory usage, we use the batch size of 2. For the time, we report the average time after 5000 trials with the image resolution of $3\times800\times1455$ on NVIDIA RTX A5000.

default p_{th} values, c) architectures used in our main paper with p_{th} values we found by following the description of the supplementary of CoTTA. Table 3a shows that our method outperforms CoTTA in all three cases even with a substantially smaller amount of memory usage and time consumption. For the experiments, we use the official repository of CoTTA[1].

C.2. Semantic Segmentation

Regarding semantic segmentation, we evaluate CoTTA with continuously changing target domains with a model pretrained on GTAV, as done in the main paper. While TENT [16] and our method show performance gains by using TTN [11], CoTTA achieves better performance by utilizing batch normalization with the test statistics (*i.e.,* TBN) than by using TTN. Therefore, we report the performance of CoTTA using the TBN and the results of TENT and ours using TTN. In Table 3b, we again observe that our method outperforms CoTTA with real-domain shifts in semantic segmentation.

Additionally, we compare the memory usage and time consumption of our method applied to TENT and other baseline models on semantic segmentation in Table 4. As shown, our method accompanies a negligible amount of resource cost. For example, while our method outperforms CoTTA, we accompany a substantially smaller amount of resource cost compared to CoTTA.

D. Further Details on Experimental Setup

D.1. Datasets

Image classification For constructing SVHN-C and Imagenet-O-C, we apply corruption types used for CIFAR-10/100-C and TinyImagnet-C by using the official code[2] of Hendrycks [7]. Since the image sizes of Imagenet-O [9] and TinyImageNet [10] are different, we resize the resolution of Imagenet-O images to 64×64. Among the 5 severity levels, we use corruption level 5, the most corrupted version. Fig. 2 shows the example images of the datasets used in our work.

Semantic segmentation For the experiments with continuously changing domains, we use the train sets of each target domain in order to conduct experiments with a long-term adaptation without using multiple rounds. Note that each target domain includes a different number of images. For example, Cityscapes, BDD-100K, and Mapillary include 2975, 7000, and 18000 images, respectively. Due to this fact, for showing the mIoU changes in Table 3b of the main paper, we evaluate models an equal number of times (*i.e.,* 20 times) for each target domain, not after certain steps. For the experiment with a fixed target domain over multiple rounds, we use the validation sets of each target domain.

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

[2] https://github.com/hendrycks/robustness

(a) Closed-set (b) Open-set

Figure 2: Examples of datasets used in our work. We use CIFAR-10-C and SVHN-C for the images. In the closet-set TTA, all images in the mini-batch only include the covariate shift (*i.e.,* domain shift). On the other hand, in the open-set TTA, half of the images in the mini-batch only include covariate shift while the other half includes both covariate shift and semantic shift (*i.e.,* open-set samples).

D.2. Baselines

Conjugate [5] Conjugate pseudo labeling was recently proposed on the observation that conjugate functions are approximate to the optimal loss function. We use the official codes[3] of Conjugate [5].

GCE [19] Generalized Cross Entropy (GCE) loss was first introduced to address the noisy labels in image classification. It emphasizes the learning of correct samples by imposing high weights on the gradients of the samples achieving low loss values, which are highly likely to be correctly annotated. Following Conjugate [5], we use GCE as the baseline model to show that simply applying existing noisy-labeling studies does not guarantee preventing the error accumulation in TTA. Since the official repository of Conjugate includes GCE codes, noted as RPL, we use the same codes in our work.

EATA [13] EATA[4] filters out samples that achieve loss values higher than a pre-defined static threshold and utilizes the fisher regularization to prevent catastrophic forgetting. For the fisher regularization, the original paper utilizes the *test set* of the source distribution to obtain the weight importance $w(\theta)$. However, we believe that such an assumption is not valid since the currently widely used corrupted test sets apply the corruptions to the test samples of the source distribution. In other words, such an approach necessitates the test samples to obtain the weight importance before encountering the test samples. Therefore, we use the *train set* of the source distribution to obtain the weight importance. For the fisher coefficient, we use 1 for CIFAR-10/100-C and 2000 for TinyImageNet-C, which are the default values reported in the main paper. For applying our method to EATA, we only replace the filtering method and utilize the fisher regularization.

SWR [3] SWR proposes 1) updating domain-sensitive weight parameters more than the insensitive ones and 2) aligning the prototype vectors of the source and the tar-

[3] https://github.com/locuslab/tta_conjugate

[4] https://github.com/mr-eggplant/EATA

get distributions [3]. Since SWR does not have an official repository, we re-implemented the codes and report the results.

D.3. Implementation Details

Image classification For the image classification on CIFAR-10/100-C, we mainly use WideResNet40 [18] which applied the AugMix [8] during the pre-training stage, following the previous recent TTA studies [11, 3, 15]. The pretrained model is available from RobustBench [4]. For the TinyImageNet-C, we use ResNet50 [6]. We pretrained ResNet50 for 50 epochs with a batch size of 256 and a learning rate of 0.01 with cosine annealing applied using the SGD optimizer [14]. We set $\lambda_{max} = 0.5$ for experiments on SWR and $\lambda_{max} = 0.25$ for the rest of the experiments.

Semantic segmentation For all semantic segmentation experiments which utilize the backpropagation, we use TTN [11] since it brings further performance gain compared to using TBN. For applying our method on semantic segmentation, we use a relaxed version: we select pixels achieving $\hat{y}^{c_o} - \tilde{y}^{c_o} \geq -0.2$. For applying our method on SWR, we reduce the coefficient of the mean entropy maximization loss (λ_{max}) from 0.5 to 0.2. The main reason is that the mean entropy maximization works as regularization and reduces the effect of entropy minimization loss. However, since our work improves the quality of entropy minimization, the mean entropy maximization rather hampers further performance gain from our method. By reducing the coefficient of mean entropy maximization, our method improves the semantic segmentation performance. Such an observation again demonstrates that our method improves the quality of the entropy minimization loss. We set other hyper-parameters as the default values.

E. Further Details on Resource Costs

We illustrate how we compute the resource costs including memory usage and time consumption. For memory usage, as mentioned in the main paper, we use the official code provided by TinyTL [1]. Note that the activation size occupies memory usage more than the parameter size [1, 15]. For ENT, which updates all parameters, we add the parameter size and activation size of all parameters. For TENT [16], which updates the affine parameters in the batch normalization layers, we only add the parameter size and activation size of the affine parameters. For SWR [3], which updates all parameters and utilizes an additional model for the regularization, we add the parameter size of the whole model parameters in addition to the memory usage of ENT. For EATA [13], which also utilizes an additional model for the fisher regularization, we only add the parameter size of the affine parameters in addition to the memory usage of TENT. For our method applied to TENT, in addition to the memory usage of TENT, we add 1) the parameter size of all parameters and 2) the parameter size of the output tensors. We add the parameter size of all parameters since we need the whole model parameters in order to compute $\tilde{y}$. Also, since we utilize $\tilde{y}$, we add the memory of the output tensors that is negligible compared to the parameter size of the whole model.

Method	CIFAR-10-C	CIFAR-100-C	TinyImageNet-C
Source [18]	18.27	46.75	76.71
BN Adapt [12]	14.49	39.26	61.90
TENT [16]	45.84	42.34	98.10
+ Ours (logit)	33.46 (-12.38)	72.08 (+29.74)	92.24 (-5.86)
+ Ours (softmax)	**14.10 (-31.74)**	**38.62 (-3.72)**	**60.87 (-37.23)**

Table 5: Variant of our method. We observe that utilizing the softmax outputs outperforms utilizing the logit values.

F. Variant of Proposed Method

To compare the prediction values between θ_a and θ_o, our method utilizes the probability values of the softmax outputs. In Table 5, we also analyze our method by using the logit values instead of the softmax values. We observe that utilizing logit values fails to bring large performance gains compared to using the softmax values. The main reason is that the logit values generally increase regardless of the correct or wrong samples. However, such an issue is not found in the softmax outputs since the values are normalized to sum-to-one vectors.

References

[1] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. *NeurIPS*, 2020. 4

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2

[3] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. *ECCV*, 2022. 2, 3, 4

[4] 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. 4

[5] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and Zico Kolter. Test-time adaptation via conjugate pseudo-labels, 2022. 3

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

[7] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019. 2, 3

[8] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020. 4

[9] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021. 3

[10] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 3

[11] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *ICLR*, 2023. 2, 3, 4

[12] 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. 2, 4

[13] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. *ICML*, 2022. 3, 4

[14] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 2, 4

[15] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. *CVPR*, 2023. 4

[16] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *ICLR*, 2021. 1, 2, 3, 4

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

[18] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016. 2, 4

[19] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018. 3