Supplementary Material

A Further Explanations

A.1 TTA with Data Augmentation Approaches

Data augmentation is being used across various fields to enhance robustness against distribution shifts, including supervised [19,20], semi-supervised [15], and self-supervised learning [1]. Similarly, in TTA, MEMO [21] applies multiple data augmentations to a single input. Notably, it utilizes 64 data augmentations on test-time input to minimize marginal entropy and enables more stable adaptation than TENT [17]. Here, MEMO applies data augmentation to the input and tunes the classifier by minimizing the averaged prediction over augmentations. In contrast, Decorruptor uses augmentation when training the diffusion model for robustness against distribution shifts before adaptation.

A.2 Detailed Contributions of Decorruptor

We clarify our Decorruptor is a classifier-agnostic generator that modifies corrupted images into clean images. Decorruptor allows for obtaining stable performance through the ensemble of multiple decorrupted images. Moreover, it can also remove corruption from images with out-of-distribution (OOD) classes, making them usable for downstream tasks. This is supported by VideoTTA results (Section 5.4) in the main text. These are key differences from other EM-based TTA approaches (*e.g.*, EATA, SAR, and DeYO) and a single robust model (*e.g.*, PIXMIX). Following Eq. (10), the final prediction is obtained by ensembling the predictions of the generated clean images with the original prediction. This indicates that Decorruptor can be applied *orthogonally* with other TTA methods that modify the original prediction. Note that the single Decorruptor checkpoint was utilized *across all datasets*, methods (*e.g.*, EATA and PIXMIX), and tasks (*e.g.*, image and video classification), demonstrating its versatility. Furthermore, Decorruptor achieves a threefold increase in speed and superior performance compared to the data augmentation-based model updating baseline.

A.3 Justification/Implication of Universal Prompt:

We chose the general text prompt (`Clean the Image`) to handle *any* unknown distribution shifts at test time (*i.e.*, corruption levels and types), Other valid prompts (*e.g.*, `Decorrupt the image`) will also work while they are fixed during training and inference time. To clarify it, since this prompt is fixed, our Decorruptor can be considered an image-to-image translation model that reverts corrupted images to their clean image counterparts.

A.4 Pseudo-codes

We provide a pseudo-code in Algorithm 1 for training Decorruptor-CM with consistency distillation [16]. Note, following LCM [11], two distinct timesteps t_n and t_{n+k}, which are k steps apart, are randomly selected and the same Gaussian noise ϵ are applied. The generated noisy latents z_{t_n} and $z_{t_{n+k}}$ are represented as follows:

$$z_{t_{n+k}} = \alpha(t_{n+k})z + \sigma(t_{n+k})\epsilon, \quad z_{t_n} = \alpha(t_n)z + \sigma(t_n)\epsilon.$$

Note, following our multi-modal guidance scheme described in the main text, the self-consistency property can be held during distillation and the skipping-step technique can also be used. In the following, we append the inference pseudo-code of both Decorruptor-DPM and CM as described in Algorithm 2.

Algorithm 1 Decorruptor-CM Training

Input: Given dataset $\mathcal{D}^{(p)}$, distance metric $d(\cdot,\cdot)$, pre-trained model parameter θ,learning rate η, EMA coeff μ, noise schedule $\alpha(t)$, $\sigma(t)$, multi-modal guidance scale: $[w_{I,\min}, w_{I,\max}]$ and $[w_{T,\min}, w_{T,\max}]$, skipping interval k, and encoder $E(\cdot)$
Encode paired clean/corrupt data into the latent space: $\mathcal{D}_z^{(p)} = \{(z_{c1}, z_{co}, c) \mid z = E(x), (x_{cl}, x_{co}, c) \in \mathcal{D}^{(p)}\}$
$\theta^- \leftarrow \theta$ ▷ Initialization
repeat
  Sample $(z_{cl}, z_{co}, c) \sim \mathcal{D}_z^{(p)}$, $n \sim \mathcal{U}[1, N-k]$
  Sample $\epsilon \sim \mathcal{N}(0, I)$, w_I and w_T
  $z_{t_{n+k}} \leftarrow \alpha(t_{n+k})z + \sigma(t_{n+k})\epsilon$
  $z_{t_n} \leftarrow \alpha(t_n)z + \sigma(t_n)\epsilon$
  Minimize Eq. (4)
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(\theta, \theta^-)$
  $\theta^- \leftarrow \text{stopgrad}(\mu\theta^- + (1-\mu)\theta)$
until convergence

B Additional Results

B.1 Detailed Results of Image Corruption Editing

Tables B.1 and B.2 present detailed performance results of Decorruptor on ImageNet-C [4] and ImageNet-$\bar{\text{C}}$ [12], respectively, using ResNet-50 [3] as the architecture. As shown in Table B.1, Decorruptor demonstrates significant performance improvement for Noise and Weather corruptions. Moreover, in Fig. B.1, 4×Decorruptor-CM shows performance improvements over the source-only case in all corruptions except for pixelate and jpeg, and it outperforms DPM in most corruptions.

DDA [2] also shares the limitation of not being able to properly edit some corruptions. Addressing this issue is crucial for the effectiveness of the input

Algorithm 2 Decorruptor-DPM/CM Inference

Input: Text and image guidance scales ω_T and ω_I, Given text c_T and corrupted image c_I, Noise schedule $\alpha(t)$, $\sigma(t)$, Decoder $D(\cdot)$
ts: Diffusion timesteps (20 for DPM, 4 for CM), T: maximum timesteps (1000)
ϵ_θ: Pre-trained DPM or CM
Sample $\hat{z}_T \sim \mathcal{N}(0; I)$, $z \leftarrow \epsilon_\theta(\hat{z}_T, c_T, c_I, T)$
for $t \leftarrow ts \dots 1$ **do** ▷ sequence of timesteps
    $\hat{z}_t \sim \mathcal{N}(\alpha(t)z; \sigma^2(t)I)$
    Eq (7), (9) for DPM, CM ▷ multi-modal guidance
    $z \leftarrow \epsilon_\theta(\hat{z}_t, c_T, c_I, t)$
end for
$\hat{x}_0 \leftarrow D(z)$ ▷ decoding latent to decorrupted image
return $\hat{x}_0$

updating TTA [17] method. As illustrated in Table B.2, Our Decorruptor shows consistent improvement for all corruptions in ImageNet-$\bar{\text{C}}$. Commonly, ensembling more edited images always presents performance improvement for all corruptions in ImageNet-C and ImageNet-$\bar{\text{C}}$.

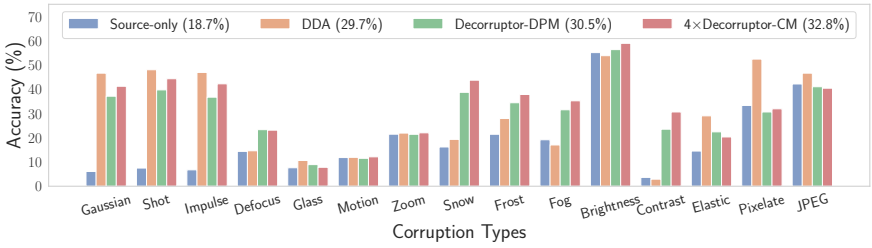

Fig. B.1: Bar graph of comparisons for performances between DDA and our Decorruptors on ImageNet-C using ResNet50.

B.2 Comparisons with EM-Based TTA Methods

In Table B.3, we present additional comparisons of our computational costs in terms of accuracy and memory compared to existing EM-based TTA methods. As a result, adding Decorruptor-4×CM resulted in significant improvements across all approaches and datasets. In terms of efficiency, DDA incurs an additional runtime of 19.5s, whereas 4×CM adds an additional runtime of 0.14s, which is only about three times the runtime of EM-based methods (0.05s). Decorruptor shows significant improvements compared to the state-of-the-art EM-based method DeYO (48.6% → 51.6%) and can be applied to other downstream tasks (*e.g.*, VideoTTA), demonstrating its strong superiority.

Table B.1: Detailed results on ImageNet-C at severity level 5 regarding accuracy (%). The **bold** value signifies the top-performing result.

	Noise			Blur				Weather				Digital				
ImageNet-C	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	Avg.
ResNet-50 (Source-only)	6.1	7.5	6.7	14.4	7.6	11.8	21.4	16.2	21.4	19.1	55.1	3.6	14.5	**33.3**	**42.1**	18.7
• Decorruptor-DPM	37.1	39.7	36.7	23.3	**8.9**	11.5	21.4	38.7	34.4	31.5	56.3	23.5	**22.4**	30.6	41.0	30.5
• Decorruptor-CM	30.3	33.4	30.7	19.5	7.4	11.6	20.8	33.2	29.8	28.5	56.0	22.1	17.9	30.6	40.2	27.5
• 4×Decorruptor-CM	41.2	44.3	42.2	23.1	7.8	12.1	22.0	43.7	37.8	35.2	58.9	30.6	20.3	31.9	40.4	32.8
• 8×Decorruptor-CM	**44.1**	**46.7**	**44.9**	**24.0**	8.0	**12.4**	**22.5**	**46.2**	**40.3**	**36.4**	**59.7**	**32.6**	20.6	33.2	41.1	**34.2**

Table B.2: Detailed results on ImageNet-$\bar{\text{C}}$ at severity level 5 regarding accuracy (%). The **bold** value signifies the top-performing result.

ImageNet-$\bar{\text{C}}$	Blue.	Brown.	Caustic.	Checker.	Cocentric.	Inverse.	Perlin.	Plasma.	Single.	Sparkles	Avg.
ResNet-50 (Source-only)	23.7	41.3	37.7	32.7	4.2	9.3	46.3	9.9	4.6	48.1	25.8
• Decorruptor-DPM	38.6	53.5	45.3	45.4	31.5	26.6	54.8	34.0	30.6	58.1	41.8
• Decorruptor-CM	37.4	51.5	43.5	44.2	27.1	21.3	53.1	29.3	26.8	56.7	39.1
• 4×Decorruptor-CM	45.6	57.4	48.2	51.5	42.2	28.2	57.5	39.8	39.6	61.2	47.1
• 8×Decorruptor-CM	**47.2**	**58.2**	**49.2**	**53.4**	**43.2**	**29.4**	**58.6**	**41.2**	**40.4**	**62.2**	**48.3**

B.3 Quantitative Video Corruption Editing Results

In this section, we elaborate on the results obtained by applying Decorruptor-CM for video corruption editing. As shown in Fig. B.2, Decorruptor-CM outperformed DDA in corruption editing. For a 3-second input video, Decorruptor takes about 10 seconds, while DDA takes nearly 20 minutes. This demonstrates that Decorruptor is both highly effective and efficient for video corruption editing. For our experiments, we referred to the performance chart of ViTTA (see Table 2 in Lin *et al.* [8]). The UCF-101C [8] dataset includes 3,783 corrupted videos for each type of corruption, covering a total of 12 different corruptions. The entire process of video decorruption was conducted using eight A40 GPUs and took about three days. The network used in the experiments was TANet [9]. Instead of using an ensemble, we assessed the performance solely using the generated dataset when combining the model update method with our approach. The results are described in Table B.4. Ensembling with the source resulted in an average performance improvement of approximately 13% compared to source-only inference. These findings suggest that our Decorruptor-CM can be effectively applied to video domains. Additionally, by applying our method with the TTA methodology, we observed an average performance improvement of about 3% compared to ViTTA, particularly showing robust decorruption results against noise.

B.4 Corruption Granularity

Our proposed Decorruptor-DPM and CM methodologies also exhibit superior decorruption capabilities across all levels of severity when compared with DDA [2].

Table B.3: Comparisons with TTA methods.

	Source	+ DPM	+ 4×CM	EATA	+ DPM	+ 4×CM	SAR	+ DPM	+ 4×CM
Runtime (s/sample)	0.004	+ 0.42	+ 0.14	0.047	+ 0.42	+ 0.14	0.054	+ 0.42	+ 0.14
Memory (MB)	2,340	+ 4,602	+ 4,958	2,704	+ 4,602	+ 4,958	2,702	+ 4,602	+ 4,958
IN-C acc (%)	18.0	31.2	**33.8**	47.8	47.5	**51.6**	44.0	47.4	**49.6**
IN-$\bar{C}$ acc (%)	25.0	41.6	**47.7**	54.0	57.6	**59.7**	49.9	55.9	**58.8**

Table B.4: Quantitative results for the UCF101-C dataset. Here, 'Ours-Only' refers to results obtained from inference using only input updates. We further provide the results of combining our methodology with the baseline TTA method.

Update	Methods	Gauss	Pepper	Salt	Shot	Zoom	Impulse	Defocus	Motion	JPEG	Contrast	Rain	H265.abr	Avg
	Source-Only	17.92	23.66	7.85	72.48	76.04	17.16	37.51	54.51	83.40	62.68	81.44	81.58	51.35
Data	Ours-Only	42.43	54.24	33.01	85.83	75.83	56.25	37.82	58.33	85.77	74.83	85.85	81.97	**64.34**
	NORM [14]	45.23	42.43	27.91	86.25	84.43	46.31	54.32	64.19	89.19	75.26	90.43	83.27	65.77
	DUA [13]	36.61	33.97	22.39	80.25	77.13	36.72	44.89	55.67	85.12	30.58	82.66	78.14	55.34
	TENT [17]	58.34	53.34	35.77	89.61	87.68	59.08	64.92	75.59	90.99	82.53	92.12	**85.09**	72.92
Model	SHOT [7]	46.10	43.33	29.50	85.51	82.95	47.53	53.77	63.37	88.69	73.30	89.82	82.66	65.54
	T3A [5]	19.35	26.57	8.83	77.19	79.38	18.64	40.68	58.61	86.12	67.22	84.00	83.45	54.17
	ViTTA [8]	71.37	64.55	45.84	91.44	**87.68**	71.90	**70.76**	**80.32**	91.70	**86.78**	**93.07**	84.56	78.33
All	ViTTA + Ours	**77.05**	**79.03**	**64.18**	**93.25**	86.54	**78.32**	65.72	78.30	**91.76**	86.41	92.25	83.58	**81.37**

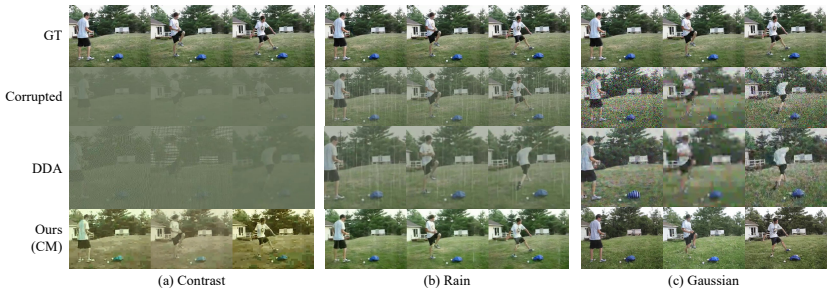

Fig. B.2: Results of corruption editing for corrupted videos in UCF101-C.

Notably, as depicted in Fig. B.3 (b), CM shows comparable results with DPM only with 4 NFEs while effectively preserving the object-centric regions of a given image. Note that background colors sometimes change due to the stochastic nature of the diffusion model.

B.5 Further Use-Cases

Furthermore, as depicted in Fig. B.4, although such image degradations were not specifically learned in our U-Net fine-tuning stage, our Decorruptor-DPM shows the editing capabilities of corruptions like haze and low-light conditions. The datasets used for these examples are the Reside SOTS [6] and LOL [18] datasets, respectively.

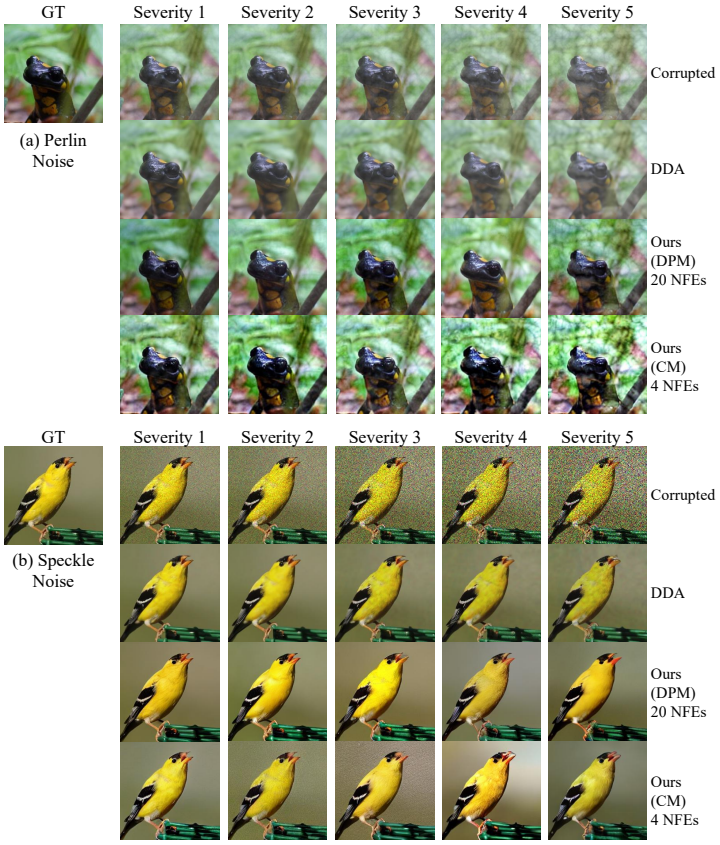

Fig. B.3: Visualization of corruption editing results based on the granularity of severity for various corruptions.

B.6 Additional Results of the Ensemble

As shown in Table 3 of Section 5.2 in the main text, the addition of an ensemble in Decorruptor-CM led to performance improvement. However, without careful consideration, increasing the number of edited images required for an ensemble results in drawbacks in terms of runtime and memory consumption. Therefore, the number of edited images also becomes a crucial hyperparameter. We illustrated the performance variations with the change in the number of images used for the ensemble in ImageNet-C and ImageNet-$\bar{\text{C}}$ in Figures B.5 and B.6, respectively. The results indicate a consistent performance increase regardless of the architecture as the number of edited images increases, and the performance tends to converge around 4 ensembles.

Fig. B.4: Further applications of our Decorruptor model in image restoration tasks.

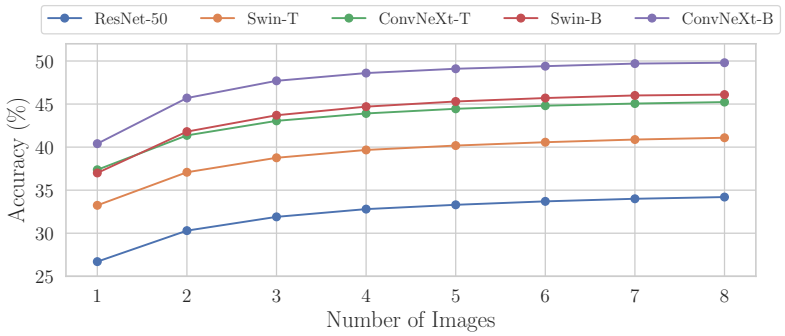

Fig. B.5: The accuracy (%) according to the number of edited images for ensembling in ImageNet-C.

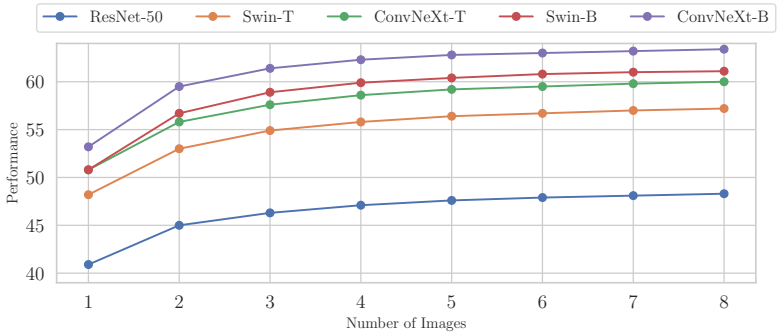

Fig. B.6: The accuracy (%) according to the number of edited images for ensembling in ImageNet-$\bar{\text{C}}$.

B.7 Diverse Corruption Scenarios, Image Sizes, and Domains

In this section, we consider a realistic scenario where an image with various mixed corruptions is encountered at test time. We evaluate the editing performance for this situation using Decorruptor-CM with 4 NFEs. As shown in Fig. B.7, we confirm that corruption editing is feasible in mixed corruption scenarios for both (a) in-domain images and (b) out-of-domain images (*e.g.*, panorama images). We use a mixed corruption severity level of 5 for each type of corruption. In each figure, the left side presents the corrupted images, while the right side displays the edited counterparts.

Fig. B.7: Visualization of experimental results on (a) in-domain and (b) out-of-domain corruption editing performance. We confirmed that our proposed method robustly performs corruption editing even in scenarios with mixed corruption at test time.

C Ablation Studies

C.1 Image Guidance Scaling on Consistency Model

For the ablation study, we trained Decorruptor-CM under three conditions: (a) using our multi-modal guidance, (b) using a fixed image guidance, and (c) without using image guidance. Each experiment involved training the model for 12K iterations, consuming 24 GPU hours on an A40 GPU. Fig. C.1 highlights the importance of our multi-modal guidance scale. We demonstrate corruption editing performance for checkerboard, Brownian noise, and Gaussian noise. As seen

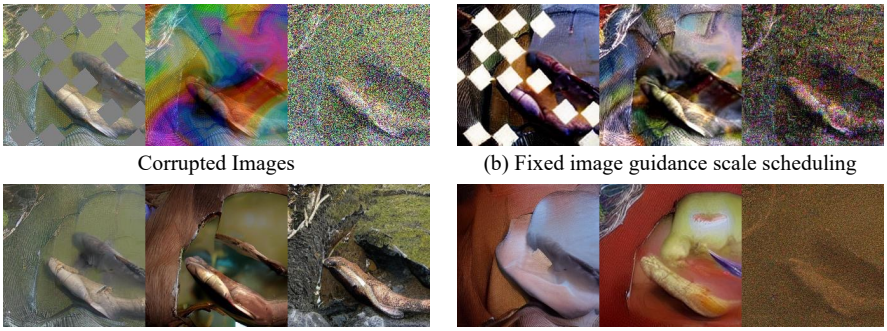

Fig. C.1: Ablation studies on (a) using the image guidance scale as conditioning, (b) fixed image guidance scale as 1.3, and (c) not using image guidance scale conditioning during distillation.

in (a), applying the proposed method by combining it with a text-guidance scale demonstrated the highest performance in corruption editing. In (b), we observe that abnormal images are generated when image guidance scale scheduling is not used for distillation. In (c), editing is minimal when the guidance scale is fixed during distillation, with the images remaining close to the original semantics. Thus, we confirm the importance of a learnable image guidance scale during the distillation stage for effective corruption editing. It is worth noting that guidance scheduling is considered not in CM inference, but only in DPM inference.

C.2 Using Other Fast Diffusion Schedulers

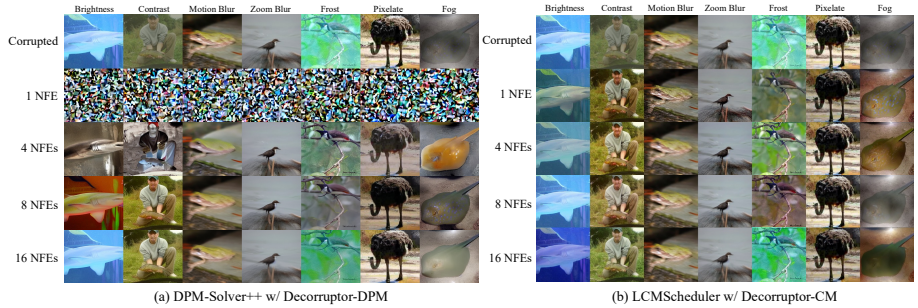

Fig. C.2: For several corruptions, (a) a combination of DPM and fast scheduler, (b) results of corruption editing according to the number of NFEs through CM. Note, we used the proposed multi-modal guidance scale conditioning method for the distillation of CM.

We conducted experiments based on the type of scheduler using the DPM-Solver++ sampler [10], traditionally utilized for fast sampling. The results, as shown in Fig.C.2 (a), indicate that the sample quality of edited images dramatically decreases with smaller NFEs, with catastrophic failure occurring at 1 NFE. Conversely, as shown in Fig.C.2 (b), our Decorruptor-CM demonstrated comparable corruption editing performance at 1 NFE to that at 4 NFEs and showed better editing quality as the NFE increased. This suggests that our proposed Decorruptor-CM enables fast, high-performance corruption editing. Each experiment was conducted with a fixed random seed.

D Failure Cases

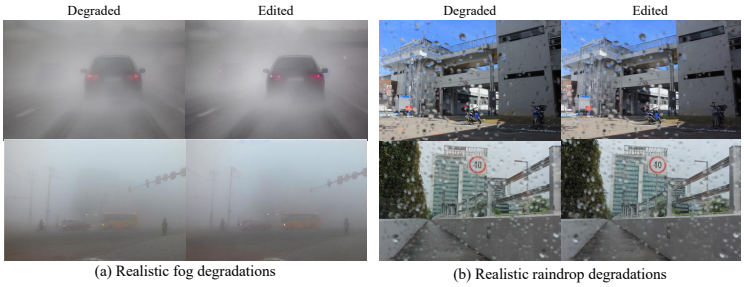

Fig. D.1: Failure cases of our model in scenarios involving realistic image degradations.

Although our method consistently outperforms other baselines, noticeable improvements were not observed in editing blur and pixelation corruptions. Furthermore, as illustrated in Fig. D.1, our model exhibits limitations in corruption editing when faced with more realistic degradations, such as fog or raindrops, which could not be accurately modeled in our corruption modeling scheme. While including paired datasets in the pre-training stage can address such realistic degradations, finding methods to edit such images at test time without having realistic paired images (*i.e.*, clean and corrupted) during training remains a challenging problem for TTA researchers.

References

1. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 15750–15758 (2021)
2. Gao, J., Zhang, J., Liu, X., Darrell, T., Shelhamer, E., Wang, D.: Back to the source: Diffusion-driven test-time adaptation. arXiv preprint arXiv:2207.03442 (2022)

3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
4. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. Proceedings of the International Conference on Learning Representations (2019)
5. Iwasawa, Y., Matsuo, Y.: Test-time classifier adjustment module for model-agnostic domain generalization. Advances in Neural Information Processing Systems **34**, 2427–2440 (2021)
6. Li, B., Ren, W., Fu, D., Tao, D., Feng, D., Zeng, W., Wang, Z.: Reside: A benchmark for single image dehazing. arXiv preprint arXiv:1712.04143 **1**, 5 (2017)
7. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In: International conference on machine learning. pp. 6028–6039. PMLR (2020)
8. Lin, W., Mirza, M.J., Kozinski, M., Possegger, H., Kuehne, H., Bischof, H.: Video test-time adaptation for action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22952–22961 (2023)
9. Liu, Z., Wang, L., Wu, W., Qian, C., Lu, T.: Tam: Temporal adaptive module for video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 13708–13718 (2021)
10. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. arXiv preprint arXiv:2211.01095 (2022)
11. Luo, S., Tan, Y., Huang, L., Li, J., Zhao, H.: Latent consistency models: Synthesizing high-resolution images with few-step inference. arXiv preprint arXiv:2310.04378 (2023)
12. Mintun, E., Kirillov, A., Xie, S.: On interaction between augmentations and corruptions in natural corruption robustness. Advances in Neural Information Processing Systems **34**, 3571–3583 (2021)
13. Mirza, M.J., Micorek, J., Possegger, H., Bischof, H.: The norm must go on: Dynamic unsupervised domain adaptation by normalization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14765–14775 (2022)
14. Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., Bethge, M.: Improving robustness against common corruptions by covariate shift adaptation. Advances in neural information processing systems **33**, 11539–11551 (2020)
15. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Advances in neural information processing systems **33**, 596–608 (2020)
16. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models (2023)
17. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. arXiv preprint arXiv:2006.10726 (2020)
18. Wei, C., Wang, W., Yang, W., Liu, J.: Deep retinex decomposition for low-light enhancement. arXiv preprint arXiv:1808.04560 (2018)
19. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6023–6032 (2019)
20. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)

21. Zhang, M., Levine, S., Finn, C.: Memo: Test time robustness via adaptation and augmentation. Advances in Neural Information Processing Systems **35**, 38629–38642 (2022)