A. Ablation

A.1. Ablation on Self-Ensembling

Since diffusion models themselves are not omnipotent, we propose a prediction fusion mechanism (Section 3.3) to automatically choose the better candidate between an image pair, the vanilla test image, and the diffusion model's generation. Entropy and confidence seem to be proper signals used for prediction fusion. According to Tent [56], they are amazing signals indicating the potential of a prediction. To some extent, the lower (higher) a prediction's entropy (confidence) is, the higher accuracy it would obtain. Based on the entropy (confidence) of a prediction, we study some possibilities to utilize the image pair better. Our exploration includes two parts, hard selection, and soft fusion, which simply selects an image from the image pair and fuses the image pair into one new image, respectively. The soft fusion can be operated on both pixel and logit levels.

Hard Selection Since entropy (confidence) can reflect the real accuracy of a prediction, a simple idea is to pick the image, logits of which have lower entropy (higher confidence) to make the final prediction.

Early Fusion Apart from the hard selection which selects an image from two, we can fuse two images on the pixel level into a new one according to entropy (confidence). We simply fuse two images, X_1 and X_2, using the weighted sum $F(a, b; f)$, where weights are from the entropy (confidence) of two images' logits, y_1 and y_2. We use a softmax operation to ensure the sum of two weights is one.

$$F(a, b; f) = [a * f(X_1) + b * f(X_2)]/[f(X_1) + f(X_2)] \quad (7)$$

It is worth noting that when using confidence, an image's weight is from its confidence, $conf(\cdot)$. The new image is $X_{new} = F(X_1, X_2; confidence)$. As for the entropy weight, it is from the other image's entropy, $ent(\cdot)$. The corresponding new image is $X_{new} = F(X_2, X_1; entropy)$.

Late Fusion Logits are also an excellent perspective for prediction fusion. We can take a similar strategy as the early fusion to fuse the logits, rather than image pixels. As for confidence fusion, the new image is $y_{new} = F(y_1, y_2; confidence)$. As for the entropy fusion, the new image is $y_{new} = F(y_2, y_1; entropy)$.

As can be seen in Table 6, late fusion shows a better performance in all six models. Experiments have shown that the prediction fusion can effectively combine information from both the test image and the diffusion model's generation, and make a more accurate precision.

Table 6. **Ablation on the design choices of selection module.** We report accuracy on corruption benchmark ImageNet-C at severity level 5 (most severe). Higher is better.

	ResNet-50	Swin-T	ConvNeXt-T	Swin-B	ConvNeXt-B
corruption	18.7	33.1	39.3	40.5	45.6
diffusion	28.4	34.6	39.3	38.6	42.8
entropy	29.7	39.7	43.9	43.9	49.2
confidence	29.6	39.8	44.0	44.1	49.2
entropy fuse	23.8	38.2	42.8	44.0	48.0
confidence fuse	23.8	38.2	42.8	44.0	48.0
entropy sum	29.7	40.0	44.2	44.5	49.4
confidence sum	29.7	39.9	44.2	44.4	49.4
sum (ours)	29.7	40.0	44.2	44.5	49.4
original test	76.6	81.2	82.1	83.4	83.9

The additional evaluation of Self-Ensembling Figure 6 evaluates the self-ensembling among DDA and state-of-the-art diffusion for adversarial defense (DiffPure). It is shown that the ensembling leads to better performance for most corruption types, and that the overall performance improvement is consistent for different image classification models. DDA is the best on average, even DDA without self-ensembling improves on DiffPure with self-ensembling, which shows the generality of our method, since both a strong domain shift and a weak one can be covered by DDA.

A.2. Ablation on Diffusion Models

We investigate different choices for scaling factor D and guidance scale $\boldsymbol{w}$ for latent guidance, related results of which are presented in Table 7 and Table 8. if $\boldsymbol{w}$ is too small, the guidance of the target image will not take effect. On the contrary, if $\boldsymbol{w}$ is too large, the shifting domain will bring a side effect. The refinement range D has the same effect as $\boldsymbol{w}$. If D is too small, the corruption on the target image will interfere the sampling process. If the scaling factor D is too large, the semantics information will be filtered by the low-pass filtering operation. The results in Table 7 and Table 8 confirm the effect of the hyper-parameters D and $\boldsymbol{w}$.

B. Tent, MEMO, BUFR, and DiffPure

B.1. Implementation

Model adaptation, including Tent, MEMO, and BUFR, is sensitive to optimization hyper-parameter, especially learning rate and optimizer type. We also compare with the input adaptation Methods DiffPure [30] We introduce the hyper-parameter in this part for the four methods.

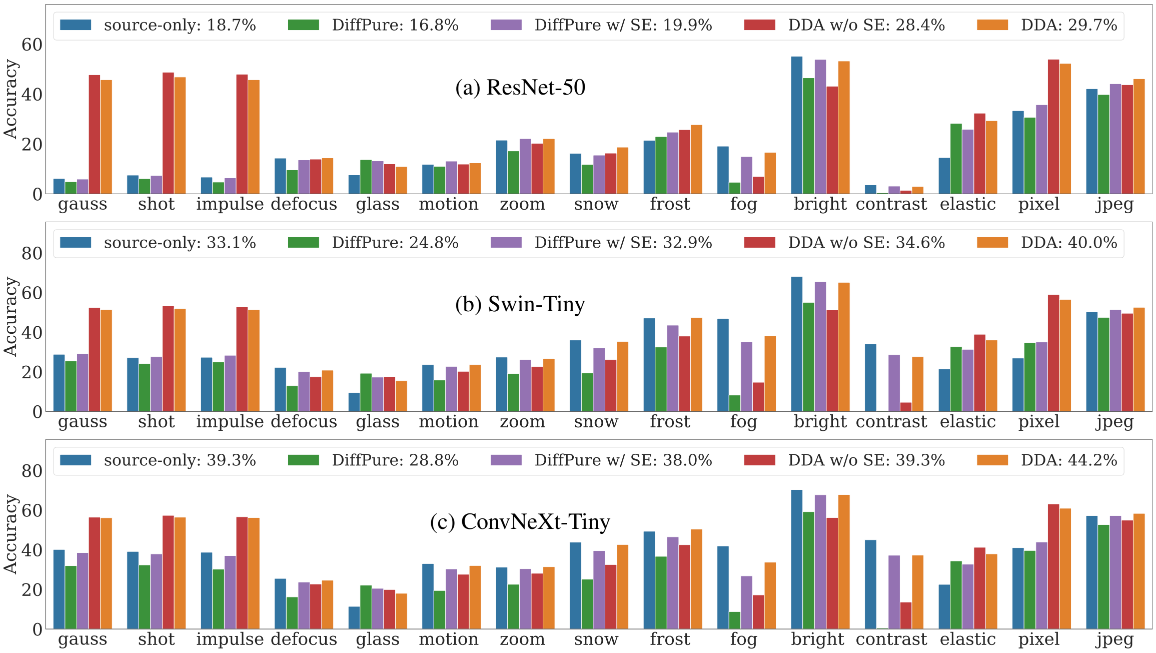

Figure 6. **Self-Ensembling reliably improves robustness across corruption types.** We evaluate the self-ensembling among DDA and state-of-the-art diffusion for adversarial defense (DiffPure). Self-Ensembling prevents catastrophic drops (on fog or contrast, for example) and improves the performance on DDA and DiffPure. DDA is the best on average, even DDA without self-ensembling improves on DiffPure with self-ensembling.

Table 7. **Ablation on choices of scaling factor D with fixed w.** Our default hyper-parameter is in the shadow row. The empirical results confirm the choice of hyper-parameters in DDA.

	$w = 6$	ResNet-50	Swin-T	ConvNeXt-T
	2	22.2	29.9	31.2
(a) elastic transformation	4	32.3	38.9	41.2
	8	41.6	45.2	48.0
	2	9.5	13.4	14.9
(b) glass blur	4	12.0	17.6	19.9
	8	19.7	26.0	29.8
	2	50.4	54.5	58.7
(c) shot noise	4	48.7	53.2	57.3
	8	40.4	44.6	48.7

Table 8. **Ablation on choices of refinement range w with fixed D.** Our default hyper-parameter is in the shadow row. The empirical results confirm the choice of hyper-parameters in DDA.

	$D = 4$	ResNet-50	Swin-T	ConvNeXt-T
	3	38.0	42.3	44.9
(a) elastic transformation	6	32.3	38.9	41.2
	9	27.8	35.2	37.9
	3	17.9	24.7	28.7
(b) glass blur	6	12.0	17.6	19.9
	9	9.2	13.4	15.2
	3	48.2	50.5	54.1
(c) shot noise	6	48.7	53.2	57.3
	9	40.7	49.1	54.4

Tent We augment the entropy loss $\mathcal{L}_{ent}$ from Tent [56] with the additional diversity loss $\mathcal{L}_{div}$, following the practice of SHOT [23]. The test-time training objective is the linear combination of these two losses. $\mathcal{L} = \mathcal{L}_{ent} + \mathcal{L}_{div}$:

$$\begin{aligned} \mathcal{L}_{ent} &= -\Sigma_c p(\hat{y}_c)\log(p(\hat{y}_c)) \\ \mathcal{L}_{div} &= D_{\mathrm{KL}}(\hat{y} \,||\, \frac{1}{C}\mathbf{1}_C) - \log(C) \end{aligned} \tag{8}$$

where C is the number of classes. $p(\hat{y}_c)$ denotes the c-th category probability in prediction $\hat{y}$. $\mathbf{1}_C$ is an all-one vector with C dimensions. Therefore, $\frac{1}{C}\mathbf{1}_C$ indicates that every class has the same evenly distributed $\frac{1}{C}$ probabilities. D_{KL} is the notation of Kullback-Leibler divergence.

Since most recent architectures, such as ViT [3], do not have BatchNorm layers anymore, we thus extend the training parameter to the whole parameter except the final classification layer. As for ResNet-like backbones, such as ResNet-50 [8], we choose SGD as an optimizer with a learning rate 0.001, momentum 0.9, and weight decay 0.0001. As for Transformer-like backbones, such as Swin-T [25] and ConvNeXt-T [26], we choose AdamW as an optimizer with a learning rate of 0.00001, weight decay 0.05. The reason behind the difference in optimizer is to follow the optimizer choice of the corresponding ImageNet training recipe.

Since the purpose of test-time adaptation is to equip the recognition model with a simple yet effective way to adapt itself, we do not change the hyper-parameter for either single or mixed domain settings. Also, there is no prior knowledge of what test domain is before the model's deployment. We cannot choose the best hyper-parameter to accompany the test-time sampling policy, batch size, *etc*. We observe that model adaptation is highly sensitive to the choice of optimization hyper-parameters. The ablation of learning rate and optimizer type could be found in Figure 7 and Figure 8.

MEMO Following the official repository of MEMO [58], we choose SGD as the optimizer with a learning rate of 0.00025. We have also explored the optimizer type and lr for different models and experiments show that the official setting is the best.

BUFR Since the official repository of BUFR [5] did not conduct the experiments on ImageNet-C with ResNet-50, we choose SGD as the optimizer with a learning rate of 0.001. The epochs per block is 3. Trained with ordered data, the average accuracy for BUFR is 2.8% / 17.9% accuracy when batch size is 1 / 64. When the class order is fixed, and the batch size is 64, BUFR achieved 4.2% / 3.4% with mixed/unmixed types. When the class order is shuffled, and the batch size is 64, BUFR achieved 4.3% / 8.3% with mixed/unmixed types. The results demonstrate that the limited, ordered, or mixed data does affect the training process and the classification accuracy.

DiffPure DiffPure [30] simply adds a given amount of noise and then reverses to $t = 0$. Following the official repository of DiffPure, we set the hyperparameter $t = 150$

B.2. Results

Benchmark Evaluation (Independent Adaptation): Tent and DDA Table 9 depicts the performance of Tent [56] and our DDA in 8 models. The first models, ResNet50, Swin-T, and ConvNeXt-T, are already mentioned in Sec 4.1. Here we additionally provide more experiments in much larger models, Swin transformer Base (Swin-B) and ConvNeXt Base (ConvNeXt-B). It is worth noting that we provide two versions of base models: 1) trained with ImageNet-1K only (denote as Swin-B and ConvNeXt-B), 2) pretrained with ImageNet-21K first and then finetuned with ImageNet-1K (denote as Swin-B* and ConvNeXt-B*). When the batch size equals one, DDA can beat Tent easily according to its advantage in tackling insufficient sampling. When the categories of test images are not shuffled, DDA still has much better performance, even with the larger state-of-the-art architectures.

Table 9. **DDA is reliably more robust on benchmark evaluation (independent adaptation) with fixed and shuffled class order.** Deployment may supply target data in various ways. To explore these regimes, we vary batch size and whether or not the data is ordered by class. We compare episodic adaptation by input updates with DDA (ours) and source-only test baseline against cumulative adaptation with Tent. DDA and source-only are invariant to these differences in the data. However, Tent is highly sensitive to batch size and order, and fails in the more natural data regimes.

Class Order	Batch Size	ResNet-50	Swin-T	ConvNeXt-T	Swin-B	ConvNeXt-B	Swin-B*	ConvNeXt-B*
✗	1	9.8	25.3	38.2	34.9	34.5	44.3	46.3
	64	10.4	7.6	30.5	7.3	34.5	18.8	44.3
✓	1	11.4	25.6	35.6	34.5	45.8	43.9	46.3
	64	37.8	50.6	54.3	57.5	59.1	64.3	67.7
Source-Only	N/A	18.7	33.1	39.3	40.5	45.6	51.6	51.7
DDA (ours)		29.7	40.0	44.2	44.5	49.4	54.5	55.4

Table 10. **DDA is reliably more robust when the target data is limited, ordered, or mixed.** Deployment may supply target data in various ways. To explore these regimes, we vary batch size and whether or not the data is ordered by class or mixed across corruption types. We compare episodic adaptation by input updates with DDA (ours) and by model updates with MEMO along with cumulative adaptation by Tent. DDA and MEMO are invariant to these differences. However, Tent is highly sensitive to batch size and order, and fails on ordered classes and mixed types.

Method	Mixed Classes	Mixed Types	Batch Size	ResNet-50	Swin-T	ConvNeXt-T
Source-Only	N/A		N/A	18.7	33.1	39.3
MEMO [58]				24.7	29.5	37.8
DiffPure [30]				16.8	24.8	28.8
DDA (ours)				**29.7**	**40.0**	**44.2**
Tent (Online)	✗	✗	1 / 64	0.1 / 0.4	2.8 / 2.3	10.5 / 9.6
	✗	✓	1 / 64	0.1 / 0.3	8.0 / 2.2	18.8 / 6.5
	✓	✗	1 / 64	0.1 / **22.6**	3.0 / **41.0**	11.0 / **50.1**
	✓	✓	1 / 64	0.1 / 6.5	8.5 / 36.9	18.9 / 47.4
Tent (Offline)	✗	✗	1 / 64	2.2 / 0.4	0.2 / 0.2	0.1 / 1.4
	✗	✓	1 / 64	1.6 / 0.5	0.2 / 0.5	0.3 / 0.5
	✓	✗	1 / 64	3.0 / **7.6**	0.1 / 43.3	0.2 / 48.8
	✓	✓	1 / 64	2.3 / 3.9	0.3 / **44.1**	0.3 / **51.9**

Challenge Exploration (Joint Adaptation): Offline Tent In Table 10, we provide a more detailed comparison between episodic, online, and offline model adaptation performance. In the first group of rows, we evaluate the episodic setting, in which adaptation and prediction are independent across inputs. In particular, the episodic source-only, MEMO, DiffPure, and DDA methods do not depend on any data except for each input in isolation. In the second group of rows we evaluate model adaptation by Tent in the on-

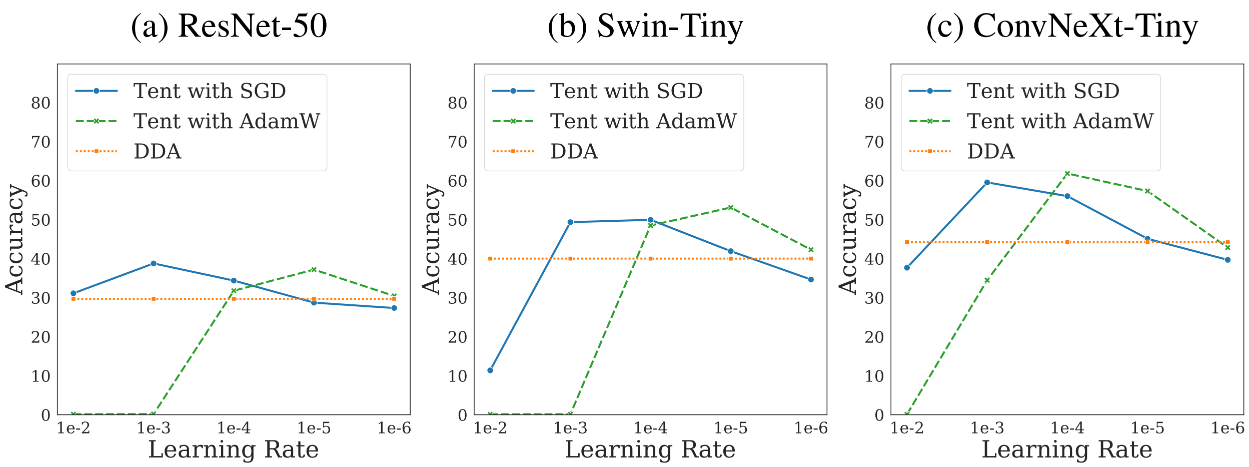

Figure 7. **DDA is invariant to learning rate while Tent is extremely sensitive on benchmark evaluation (independent adaptation).** To analyze sensitivity to the learning rate we measure the average robustness of independent adaptation across corruption types. DDA does not depend on these factors while Tent fails without the proper tuning on learning rate.

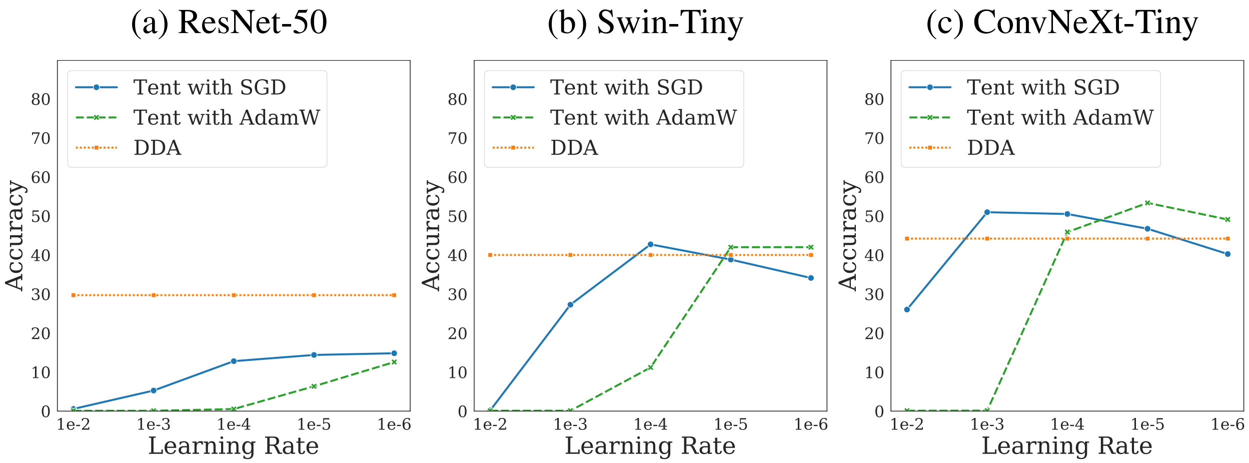

Figure 8. **DDA is invariant to learning rate while Tent is extremely sensitive on challenge exploration (joint adaptation).**

line setting, in which adaptation and prediction are done per batch, and adaptation updates persist across batches. (This is the setting reported in Table 3 of the main paper, as it is the recommended setting for Tent.) In the third group of rows, we evaluate model adaptation by Tent in the offline setting, in which the method first adapts to the entire test set, and then makes predictions for each input. In this setting, Tent first learns from all of the data but then makes predictions with a single model that cannot specifically adapt to each batch. Whether online or offline, Tent is sensitive to the order of the data, and fails when data arrive one by one (with a batch size of one) or when classes are not mixed by shuffling.

Challenge Exploration (Joint Adaptation): fixed and shuffled class order As shown in Figure 9 and Figure 10, the shuffled class order is essential to Tent, especially for models with low capacity. Even for big models, the potential of Tent as the batch size increases is suppressed largely.

C. More datasets

ImageNet-W ImageNet-W [22] is an evaluation set based on ImageNet for the watermark. The authors found that the watermark as a shortcut affects nearly every modern vision model. In our experimental settings, the watermark can be regarded as a common corruption by human activities, as a supplement dataset of the unexpected corruptions, ImageNet-C (IN-C) [10] and ImageNet-$\bar{\text{C}}$ (IN-$\bar{\text{C}}$) [27]. Figure 11 shows that DDA can effectively remove the water-

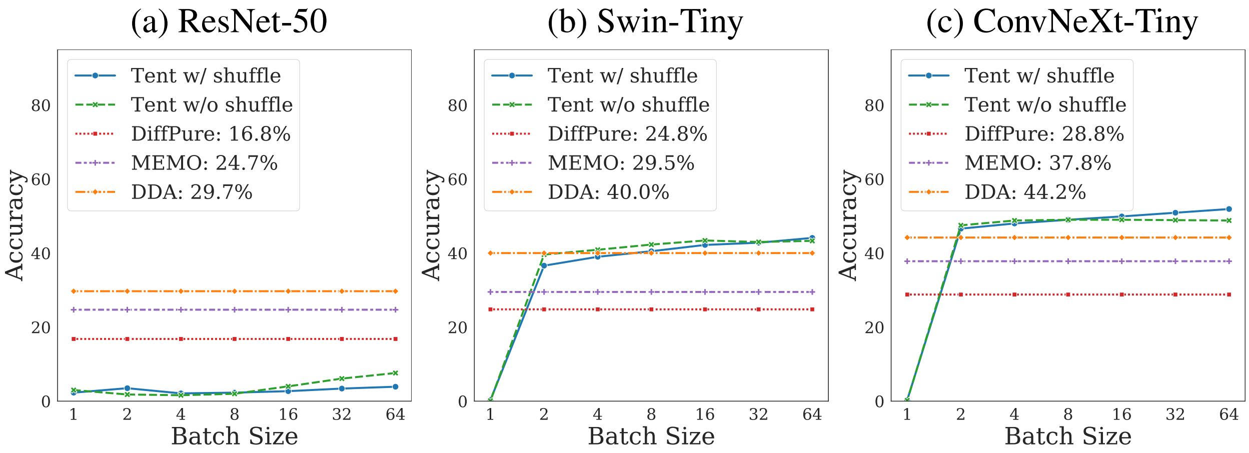

Figure 9. **DDA is reliably more robust on challenge exploration (joint adaptation) with shuffled class order.** Deployment may supply target data in various ways. To explore these regimes, we vary the batch size and whether or not the data is ordered by class or mixed across corruption types. We compare episodic adaptation by input updates with DDA (ours), DiffPure, and by model updates with MEMO against cumulative adaptation with offline and online Tent. DDA, DiffPure, and MEMO are invariant to these differences in the data. However, Tent is highly sensitive to batch size and order, and fails in the more natural data regimes.

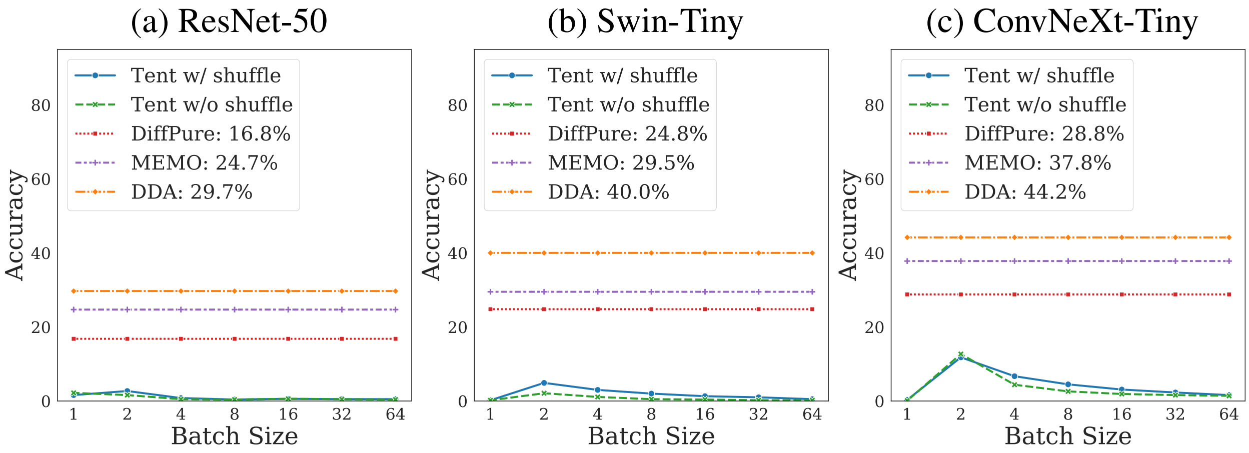

Figure 10. **DDA is reliably more robust on challenge exploration (joint adaptation) with fixed class order.**

mark to avoid shortcut reliance. Our experiments, conducted on ImagenNet-W using ResNet-50, showed that DDA achieved 58.3% accuracy, a significant improvement over the baseline accuracy of 47.4%. Both the visualizations and model performance demonstrate that DDA can enhance the robustness of the model against watermark corruption.

ImageNet-R ImageNet-R [9] is an evaluation set based on ImageNet for rendition, containing cartoons, embroidery, graphics, paintings, sketches, tattoos, toys, and so on. Our qualitative experiments in Figure 12 depict the performance of DDA on rendition to real images. Although the adapted images still reserve the original background, real-world characteristics have been added to the main part of the picture. DDA easily concentrates on the core domain shift in the renditions, especially for the gap between the 2D and 3D features.

D. Visualization

Progressive generation on ImageNet-C and ImageNet Figure 13 illustrates how diffusion models denoise and reconstruct the given corrupted images. Here we take elastic transformation, glass blur, and shot noise as the representative corruption for the digital artifact, natural blur, and common noise. We observe that diffusion models are able to clean up the "local", high-frequency noises, *i.e.* Gaus-

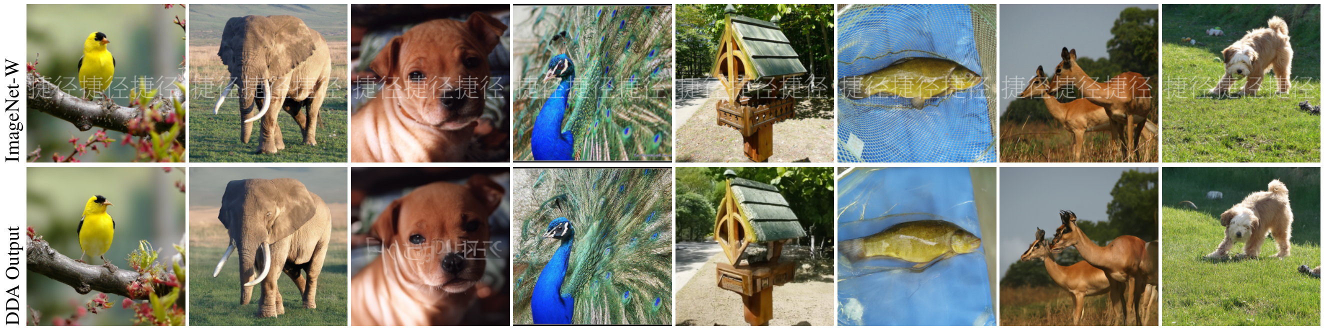

Figure 11. Visualization of generated images with DDA output on ImageNet-W.

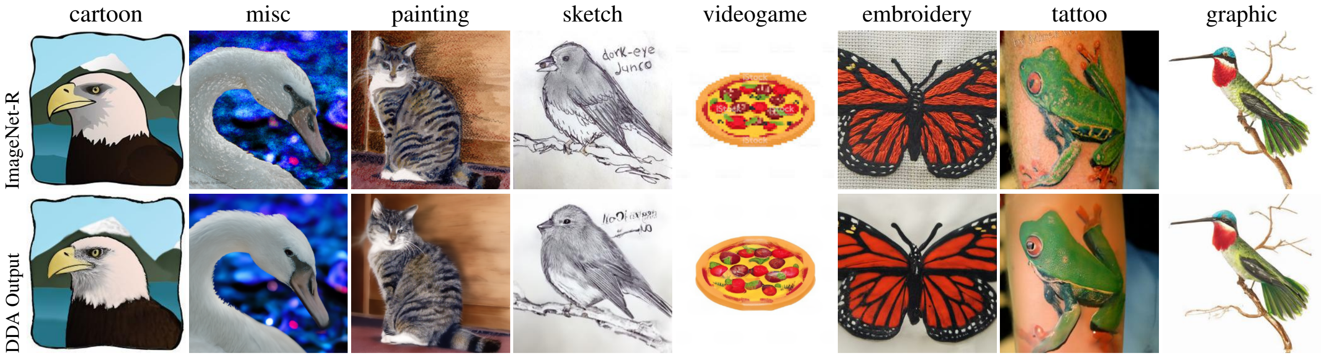

Figure 12. Visualization of generated images with DDA on ImageNet-R.

sian noise, pixelate, etc. As for "global", low-frequency corruption, *i.e*. fog, snow, *etc*., diffusion models failed to recover the original version. One reason behind this phenomenon could be these low-frequency corrupted images are treated as natural samples during ImageNet training. In other words, the diffusion model is trained on ImageNet, which may cover several augmentations including these low-frequency corruptions.

Figure 14 visualizes the procedure given the original ImageNet validation images. We observe that images after diffusion are almost the same as the original ones. In general, the reconstructions from diffusion models look similar to the original ImageNet validation images, which indicates the effectiveness of the leveraged generative models. It is worth noting the comparison between the output and the original in the second row, peacock. Diffusion models hallucinate more details on the left that do not exist in the original input image.

Qualitative results on success and failure Cases We visualize success and failure cases across corruptions on ImageNet-C as shown in Figure 15 and Figure 16. DDA forces the model to preserve the global structural information to avoid semantic drift, leading to a drawback that it may fail to adapt images from certain domains. While DDA performs well when projecting most high-frequency/local corruptions (*e.g*., Gaussian noise, impulse noise, jpeg encoding, . . .), it fails for a few low-frequency/global corruptions (*e.g*., frost, fog, brightness adjustment, . . .). However, our self-ensembling scheme effectively detects these cases to avoid significant drops in accuracy.

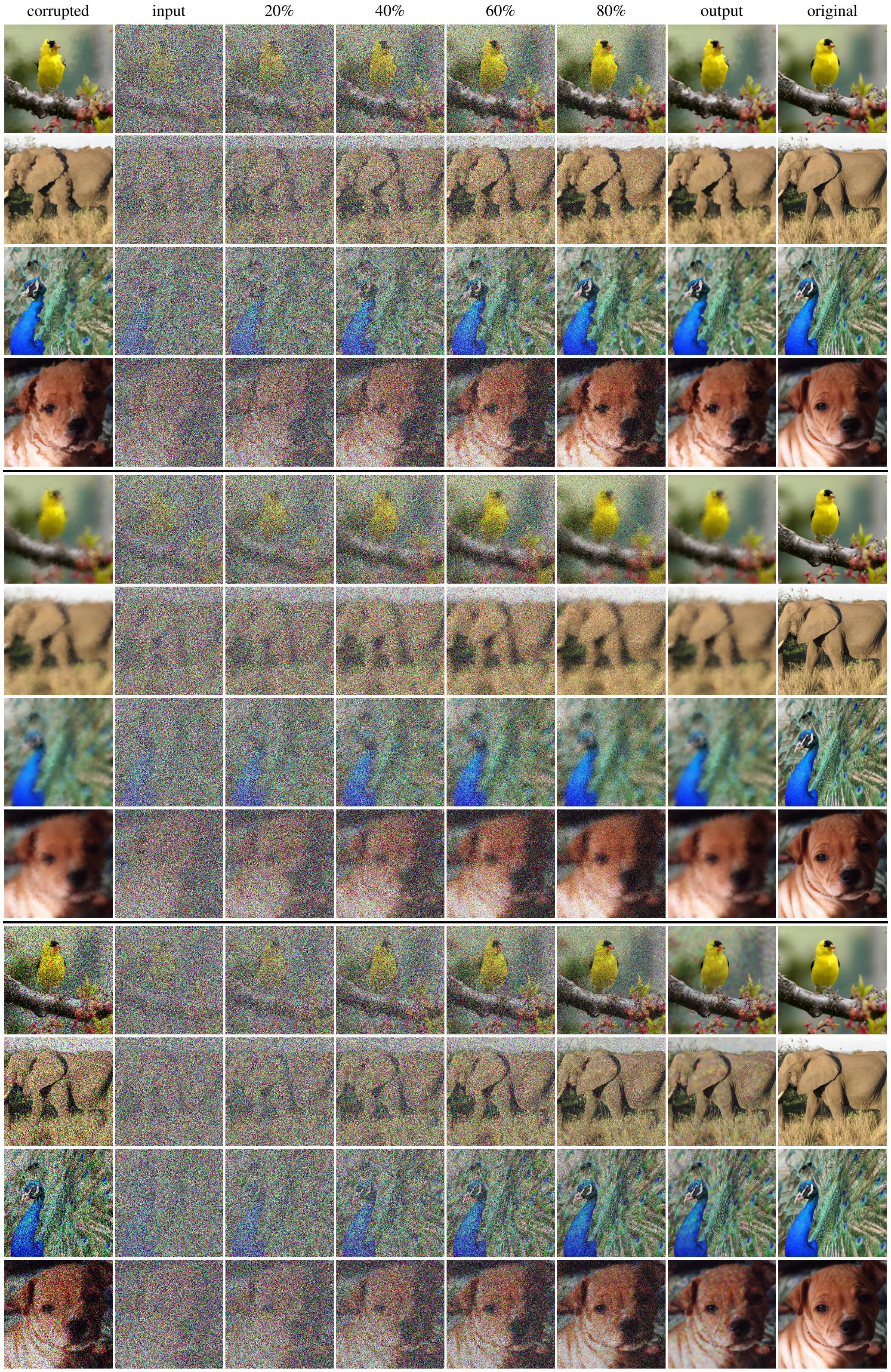

Figure 13. Visualization of generated images with diffusion models, given highest severity corrupted images during the test time.

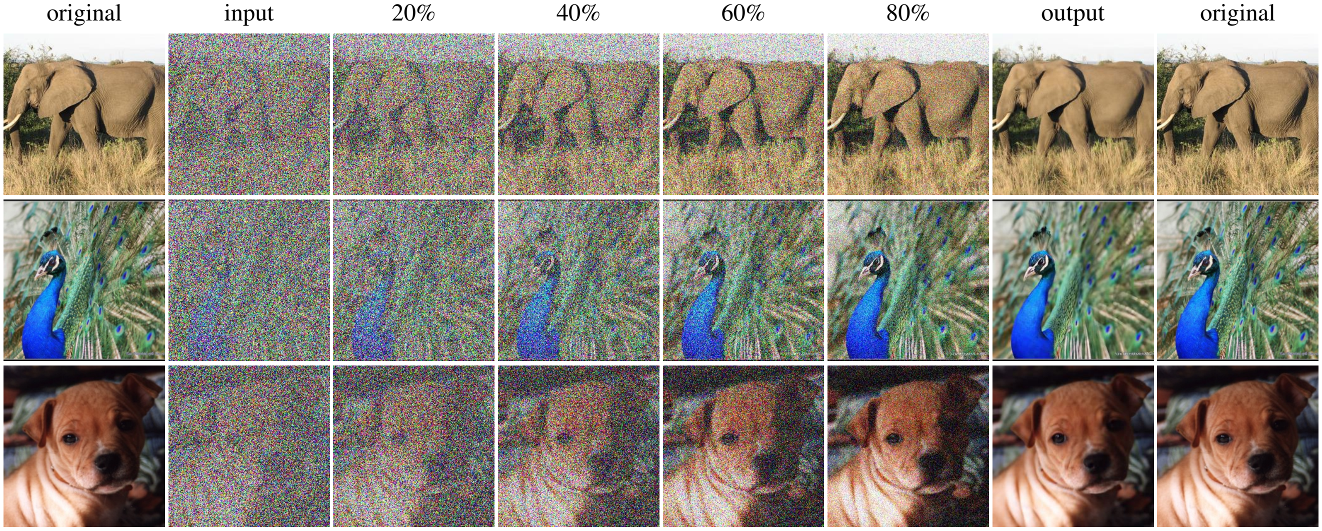

Figure 14. Visualization of the progress of image generation with diffusion models, given the original images during the test time.

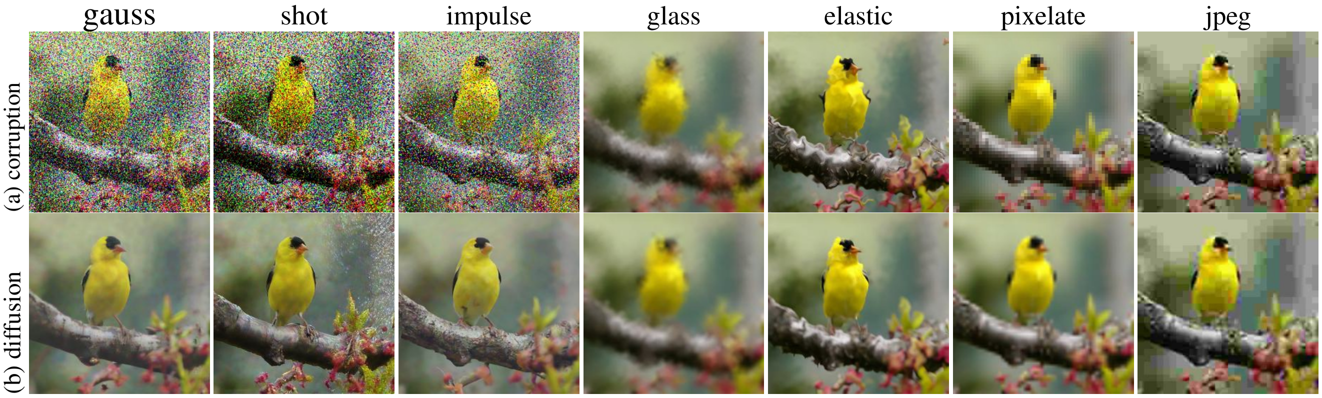

Figure 15. Visualization of *positive* generated images with diffusion models, given highest severity corrupted images during the test time.

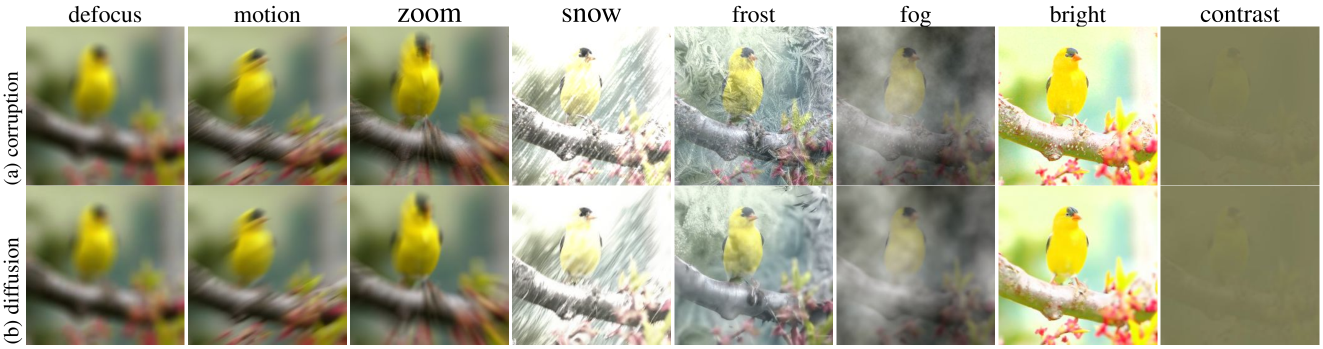

Figure 16. Visualization of *negative* generated images with diffusion models, given highest severity corrupted images during the test time.