GDA: Generalized Diffusion for Robust Test-time Adaptation

Supplementary Material

7. Implementation Details

Style loss We apply the CLIP model with model architecture *ViT-Base/16* for calculating the style loss. By leveraging the rich semantic information of CLIP, we are able to shift the OOD sample to the source domain. It has been used in [C2] for style transfer. The input images are presented to the model as a sequence of fixed-size patches, where the patch size is 16*16). We get the corresponding image embedding for all image patches from the output of the visual encoder of CLIP model. We then calculate the similarity between the image embeddings and the text token embedding extracted from language encoder of CLIP model. The text prompts we use for style loss are the words related to *photo-realistic* or *real photo*. We assume partially knowing the source domain information is allowable in domain generalization.

Content preservation loss We provide a more detailed of the contrastive loss for content preservation. The input of content loss is a batch of features extracted from generated sample itself x_t^g and the corresponding source sample x_0. For example, v is the i^{th} patch in sample x_t^g, the i^{th} patch p in sample x_0 is its positive pair $p+$, and all the other patches except the i^{th} patch in sample x_0 will be the negative pair $p-$. The purpose of the contrastive loss is to force the feature distance between a patch p and its corresponding positive patch $p+$ to become closer to each other under the latent space. Meanwhile, the loss forces p and $p-$ apart from each other.

Marginal entropy loss We adopt AugMix [12], a data augmentation tool from Pytorch, which randomly select several augmentation functions (e.g., posterize, rotate, equalize) to augment the data. The augmention set $\mathcal{A} = A_1, A_2, ..., A_k$ excludes operations that overlap with corruption types in ImageNet-C. For generating one augmented sample x_{aug}, we set the mixing weight $w_1, w_2, ...w_3$ for every augmentation in $\mathcal{A}$. The mixing weight, which is a k-dimensional vector of convex coefficients, is randomly sampled from a Dirichlet distribution. The augmented sample x_{aug} equals to $w_n * A_n(...(W_2 * A_2(w1 * A_1(x_{orig}))$.

Analysis of Hyperparameters in the Loss Term We conduct the sensitivity analysis on hyperparameters for every loss. We follow the range of hyperparameters used in [49]. In Table 5, we whoe the results of ImageNet-R under different combination of loss terms.

Style loss					
Param.	1000	5000	15000	20000	30000
Acc.	38.6	**44.5**	44.0	44.2	40.1
Content loss					
Param.	100	500	700	1000	1500
Acc.	38.8	39.4	42.6	**44.5**	39.8
Marginal loss					
Param.	50	100	150	200	250
Acc.	38.7	38.9	41.6	**44.5**	42.4

Table 5. Hyperparameter analysis for ImageNet-R

The Impact of Different Loss Term We show the impact of different loss term by removing content preservation loss or style loss in Table 6. The result of using only style loss is better than content loss on ImageNet-Rendition and Sketch.

	w/o style	w/o content	w/o marg.	GDA (Ours)
Rendition	37.7	37.9	39.4	44.5
Sketch	23.3	23.5	23.9	25.5

Table 6. The Impact of Different Loss Term

The Choices of Hyperparameters In GDA, the weights for each loss function are hyperparameters that need to be chosen by users. We combine the three loss terms as a joint optimization, with their Lagrange multipliers as hyperparameters. The hyperparameter values for each benchmark are shown in Table 7.

	Marg. Entropy	Style	Content
ImageNet-C	100	5000	1500
Rendition	200	5000	1000
Sketch	200	1000	700
Stylized	200	1000	700

Table 7. Hyperparmeter setting for marginal entropy loss, style loss, and content preservation loss. The number will be multiplied on every loss function during the optimization.

8. More Experimental Results

In this section, we show more experimental results on GDA, including the detailed results of ImageNet-C on different severity, comparison with input-based adaptation baselines, and model-based adaptation baselines.

8.1. ImageNet-C Detailed Results

In main paper Table 1, we show the average accuracy on 15 types of corruption for ImageNet-C. Here, in Table 8, we show the detailed comparison of GDA with Standard and three diffusion-based baselines. The four main groups of corruption, Noise, Blur, Weather, and Digital, are composed of 15 types of corruptions. We show the detailed corruption types in every group in Table 9. Our GDA improves the robust accuracy by 4.4%∼5.64% on three standard models and outperforms every baselines.

		Standard	DiffPure [30]	DDA-10 [5]	w/o marg.	GDA (Ours)
ResNet50 [8]	Noise	23.6	17.03	33.4	29.2	**37.0**
	Blur	30.5	9.28	26.8	32.4	**36.2**
	Weather	45.1	11.42	39.7	46.4	**46.5**
	Digital	50.1	25.62	47.9	50.9	**52.0**
	Avg. Acc.	37.3	15.83	36.9	40.9	**41.7**
ConvNext-T [23]	Noise	64.2	56.30	66.17	63.96	**78.99**
	Blur	44.83	31.4	**50.68**	44.32	47.78
	Weather	64.67	45.46	65.92	63.75	**67.83**
	Digital	67.15	55.8	70.3	66.77	**70.08**
	Avg. Acc.	59.60	47.23	63.26	59.70	**65.24**
Swin-T [22]	Noise	57.56	44.93	50.4	59.7	**64.3**
	Blur	38.05	19.27	38.85	39.3	**45.2**
	Weather	59.68	35.63	50.05	61.1	**62.2**
	Digital	62.03	42.93	59.3	63.33	**65.7**
	Avg. Acc.	54.33	35.69	49.65	55.86	**59.35**

Table 8. Performance on the ImageNet-C for three model architectures under four groups of corruptions. Numbers in bold show the best accuracy.

	Corruption Types
Noise	Gaussian Noise, Impulse noise, Shot noise
Blur	Motion blur, Zoom blur, Defocus blur, Glass blur
Weather	Snow, Frost, Fog, Brightness
Digital	Contrast, Jpeg compression, Pixelate, Elastic transform

Table 9. Detail of four corruption groups with 15 corruption types

Results of Severity 5 In Table 10, we show more experimental results on ImageNet-C under severity 5. We compare the results between GDA and the four baselines, including Standard, Diffpure [30], DDA [5], and w/o marginal. GDA consistently achieves the highest accuracy and surpasses all baselines.

	ResNet50	ConvNext-T	Swin-T
Standard	18.7	39.3	33.1
Diffpure [30]	16.8	28.8	24.8
DDA [5]	29.7	44.2	40.0
w/o marg.	30.2	44.4	41.6
GDA (ours)	**31.8**	**44.8**	**42.2**

Table 10. The average classification accuracy on the ImageNet-C under severity level 5 for three model architectures.

8.2. Compare with Input-based Adaptation

Similar to our GDA, prior works studied input-based adaptation [1, 25, 43], updating the *input* during the inference time. However, most of them typically focus on adding extra vectors or visual prompts (VP) to the input and optimizing with pre-defined objectives, which is different from our diffusion-based method. To better understand the efficacy of traditional VP and diffusion-based approaches, we compare the performance of GDA with several input-based adaptation baselines in Table 11. As Table 11 shows, compared to BN and Memo, GDA outperforms all four input-based adaptation baselines by 2.42% to 4.46% in avgerage accuracy, which demonstrates that our proposed diffusion-based method is better than the baselines which add vector directly to the input pixel. We explain each input-based adaptation baselines as follows.

Baseline details for input-based adaptation

- **Self-supervised Visual Prompt (SVP) [25]**: The prompting method to reverse the adversarial attacks by modifying adversarial samples with ℓ_p-norm perturbations, where the perturbations are optimized via the self-supervised contrastive loss. We extend this method with two different prompt settings: ***patch*** and ***padding***. For the patch setup, we directly add a full-size patch of perturbation into the input. For the padding setup, we embed a frame of the perturbation outside the input.
- **Convolutional Visual Prompt (CVP) [43]**: The prompting method that adapts the input samples by constructing the convolutional kernels. Given a corrupted sample x and a convolutional kernel k. The convolutional kernels can be initialized with random initialization and optimized with a small kernel size (e.g., 3*3 or 5*5) by projected gradient descent using self-supervised loss. We convolve the input x with the convolutional kernel k and update them iteratively by $x' = x_0 + \lambda * Conv(x_0, k)$, where the λ parameter controls the magnitude of convolved output when combined with the residual input. We set the range to be [0.5, 3] and run test-time optimization to automatically find the optimal solution. We chose the contrastive loss as our self-supervision task.

	Standard	SVP (patch)	SVP (padding)	CVP (3*3)	CVP (5*5)	GDA
Noise	28.85	29.37	29.38	31.59	30.53	**37.03**
Blur	30.45	29.59	29.58	30.80	31.0	**32.4**
Weather	42.99	41.18	41.22	42.27	42.45	**46.5**
Digital	50.45	48.96	48.96	**52.58**	51.45	50.98
Avg.	38.19	37.27	37.28	39.31	38.85	**41.73**

Table 11. Compare GDA with input-based adaptation baselines.

8.3. Compare with Model-based Adaptation

In Section 2, we introduce prior existing works on *model-based* adaptation, such as TENT [45], BN [38], and MEMO [50]. While they all focus on updating the model weights during the inference time, such as changing batch normalization statistics or the scaling parameters in the batch-norm layer, GDA updates the input directly using the diffusion model. We compare our GDA with three model-based adaptation baselines in Table 12, including TENT, BN, and Memo. For TENT and BN, they adapt the models by input batches, which is different from GDA's setting, as we do the single-sample adaptation. Therefore, we set up the batch size for TENT and BN as 16. For Memo, the same as our single-sample adaptation setting, we set the batch size as 1. We evaluate the accuracy on ResNet50 backbone for every corruption group for GDA and three baselines. As Table 12 shows, compared to BN and Memo, GDA has a 0.3 to 2.7 points gain in robust accuracy. However, GDA is slightly worse than TENT by 2.16 points.

Baseline details for model-based adaptation

- **BN[38]**: The model adaptation method aims to adjust the BN statistics for every input batch during the test-time. It requires to adapt with single corruption type in every batch.
- **TENT [46]**: The method adapts the model by minimizing the conditional entropy on batches. In our experiment, we evaluate TENT in $episodic$ mode, which means the model parameter is reset to the initial state after every batch adaptation.
- **MEMO [50]**: The model adaptation method proposed in [50] alters a single data point with different augmentations (ie., rotation, cropping, and color jitter,...etc), and the model parameters are adapted by minimizing the entropy of the model's marginal output distribution across those augmented samples.

	Standard	**BN [38]**	**TENT [45]**	**Memo [50]**	**GDA (Ours)**
Noise	28.85	31.14	35.75	32.61	**37.03**
Blur	30.45	28.79	33.63	**34.31**	32.4
Weather	42.99	44.81	**49.65**	44.93	46.5
Digital	50.45	51.39	**56.53**	53.76	50.98
Avg.	38.19	39.03	**43.89**	41.40	41.73

Table 12. Compare GDA with model-based adaptation baselines

9. Visualization

We visualize more saliency maps on different types of OOD. As Figure 7 and 8 shows, from left to right for every subfigure, the first row is the original / corrupted, and adapted samples; the second row shows their corresponding Grad-CAM with respect to the predicted labels. The red region in Grad-CAM shows where the model focuses on for target input. We empirically discover the heap map defocus on the target object for corrupted samples. However, after adapting by GDA, the red region of the adapted sample's heap map is re-target on the similar region as original image, which demonstrates that the diffusion indeed improves the input adaptation and makes the model refocus back on the correct regions.

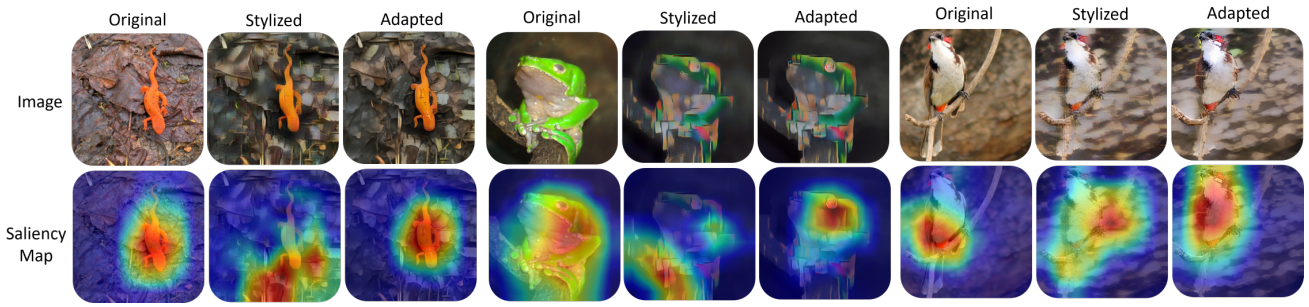


Figure 7. GradCam Visualization on ImageNet-Stylized

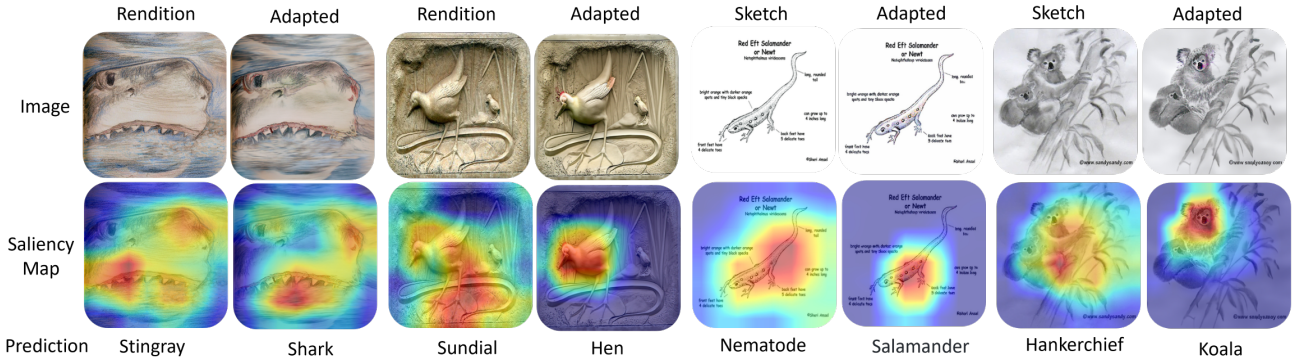
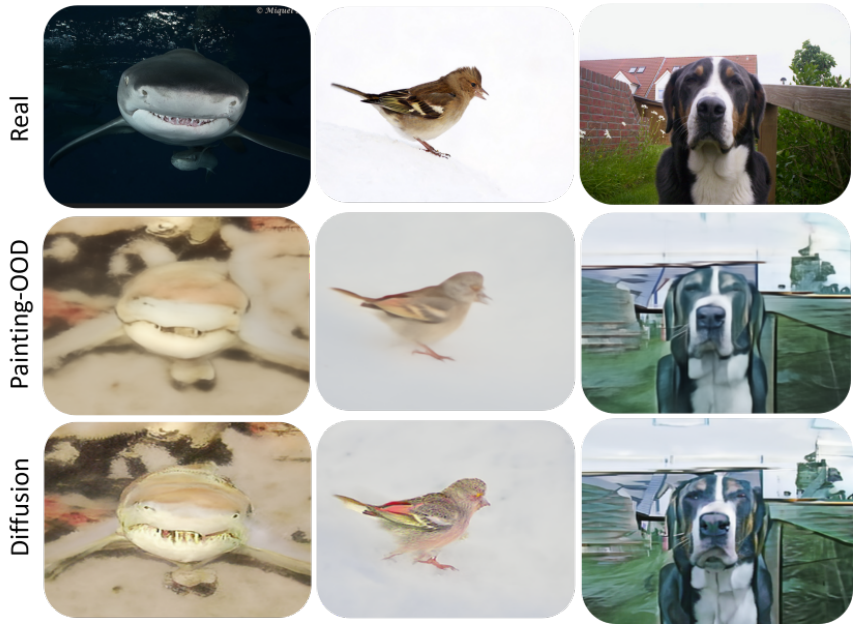


Figure 8. GradCam Visualization on ImageNet Rendition and Sketch

(a) Imagenet-Sketch

(b) Imagenet-Rendition



(c) Imagenet-Stylized

Figure 9. More GDA visualization for different OOD benchmarks, including Sketch, Rendition, and Stylized-ImageNet. We show that GDA not only can effectively guide the samples back to the source domain but also can visually change the sample with visual effects, such as colorizing the sketch images, background removing for painting-style samples, and object highlighting for stylized samples.