7 Appendix

7.1 Algorithm

Algorithm 1: Convolutional Visual Prompt

Input: Pretrained classifier $\mathcal{F}(\cdot)$, OOD images x, Self-supervised objective function $\mathcal{L}_s(\cdot)$, Convolutional operator $Conv(\cdot)$, Convolutional kernel k, Learning rate η, Number of iteration $\mathcal{T}$

Output: Class prediction $\hat{y}$ for adapted sample of x

Inference
 # Initialize the kernel parameters
$k^0 \sim \mathcal{U}\{(\alpha, \beta)$
 # Calculate initial SSL loss
$loss^0 = \mathcal{L}_s(x)$
for $t \in \{1, ..., T\}$ **do**
  # Generate adapted samples
  $x^t = x + \lambda * Conv(x, k^t)$
  # Calculate SSL loss with adapted samples
  $loss^t = \mathcal{L}_s(x^t)$
  # Update kernel parameters
  $k^{t+1} = k^t + \eta \frac{\partial loss^t}{\partial k^t}$
 # Get optimal kernel parameters
$k^\star \leftarrow k^T$
if $loss^T > loss^0$ **then**
  # Use the initial kernel parameters
  $k^\star \leftarrow k^0$;
 # Get final adapted samples
$x^\star = x + \lambda * Conv(x, k^\star)$
return $\hat{y} \leftarrow \mathcal{F}(x^\star)$

7.2 Baseline Details

Here, we show the detail of the baselines that we compare with.

• **Standard**: The baseline uses pre-trained model without adaptation. For CIFAR-10-C, the standard is trained with 50000 clean CIFAR-10 train dataset on WideResNet18 and ResNet. For ImageNet1K-C, the standard is trained with ∼1.2M clean ImageNet train dataset on ResNet50.

• **SVP**: The self-supervised visual prompts attempt to reverse the adversarial attacks by modifying the input pixels with ℓ_p-norm perturbations, where the perturbations are optimized via contrastive loss [40]. For the patch setting, we setup the shape of VP as 32*32*3 for CIFAR-C and 224*224*3 for all ImageNet OOD datasets. For the padding setting, we set the padding size as 1 for CIFAR-10-C and 15 for ImageNet OOD dataset. Take CIFAR data as example, we first initialize a mask with all zeros value with the shape 30*30*3 and set the pad value as 1 with padding size 1 so that the mask after padding is as the same shape of CIFAR data (32*32*3). Then, we multiply the mask with the VP to preserve only the VP located at the position we just pad with 1 value. We can further optimize the VP with mask by adding it with the corrupted samples.

• **BN[53]**: 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.

• **TTT [57]**: The test-time training trains the model with an auxiliary SSL rotation task and leverages the rotation loss for model adaptation during the testing time. In TTT method, instead of adapting the whole model, they only adapt the last few layers of the model and freeze the parameters in the front layers.

• **MEMO**: The model adaptation method proposed in [69] 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.

• **TENT [61]**: 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.

7.3 Implementation details

For the training part of SSL model, we set the training parameters with batch size as 64, training epoch as 200, and the learning rate (*lr*) as 0.001. The *lr* is decayed with a cosine annealing for each batch [37]. The transformations for contrastive learning are predefined. We augment the inputs with random resize crop, random flip, and random rotation in degree [-90, 90] for positive/negative pairs generation in every batch. The number of transformations for one sample is set as 3.

For the test-time adaptation part, we set the range of parameter δ for VP. For the ℓ_2-norm perturbations, the ϵ is [-8/255,8/255] and the step size is 2/255. We set the iteration number i either as 1 or 5, which means each component has 1 or 5 steps during PGD.

For the convolutional visual prompts (CVP), Table 8 shows the kernel setting for different datasets. For the fixed initialization setting, a sharpen kernel is used at the beginning. For example if the kernel size is 3, we set up the sharpened kernel as [[0,-1, 0], [-1, 5, -1], [0, -1, 0]].

	CIFAR-10-C	ImageNet-C,R,S,A
Kernel Size	3*3	3*3 / 5*5
λ	[0.5, 1]	[0.5, 3]
Update iters.	1, 5 (default), and 10	
Initialization	fixed / random	

Table 8: parameter setting

• **Number of Trainable Parameters**: We compare the trainable parameters v.s. accuracy for different prompting methods. As Figure 6 shows, CVP contains less than 0.2% number of trainable parameters, compared to VP(patch).

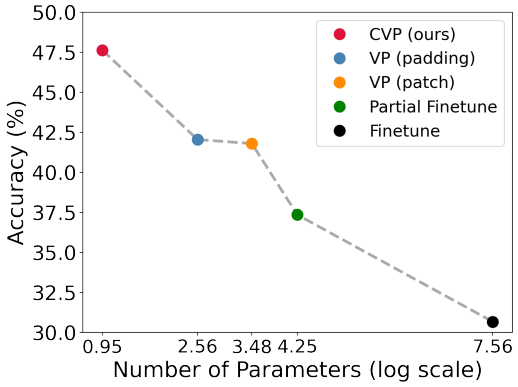


Figure 6

7.4 More Evaluation

• We show more detailed results for CIFAR-10-C in Table 9 and 10. In Table 11, we further compare TTT [57], which is a test-time method with CVP.

	Standard	BN	Finetune	VP		CVP			
				patch	append	fixed (3*3) w/o update	fixed (3*3) w/ update	rand (3*3) w/o update	rand (3*3) w/ update
Gaussian Noise	19.90	24.51	19.26	20.07	19.92	23.11	23.50	24.59	26.27
Shot Noise	20.37	24.25	19.43	20.56	20.40	22.95	23.32	24.29	**25.26**
Impulse Noise	27.44	26.14	19.19	27.54	27.43	30.76	30.98	31.13	**31.08**
Defocus Blur	12.90	14.54	13.39	13.59	13.83	16.81	17.41	18.85	**20.03**
Motion Blur	23.26	19.95	27.67	30.50	26.02	29.21	30.03	30.37	**31.89**
Glass Blur	25.97	33.30	18.61	19.70	39.57	36.14	37.54	37.42	**40.51**
Zoom Blur	71.08	50.46	42.42	71.07	54.08	86.76	87.65	85.76	**88.19**
Brightness	89.38	71.47	61.03	89.39	77.25	89.37	**89.39**	89.25	89.31
Snow	71.21	49.73	39.95	71.52	**72.29**	71.23	71.44	71.17	71.52
Frost	74.83	58.40	46.35	**74.93**	48.66	74.79	74.81	74.69	74.90
Fog	45.69	42.45	32.96	46.52	**82.95**	50.08	51.42	49.64	51.65
Contrast	58.36	49.87	39.22	57.95	58.45	67.74	68.99	69.05	**70.21**
Elastic Transform	17.54	24.01	19.95	17.72	18.95	17.39	17.62	18.07	**19.66**
Pixelate	23.45	39.80	34.26	24.10	**35.86**	25.91	26.47	28.39	30.58
Jpeg Compression	45.06	31.37	26.42	**45.65**	31.37	43.99	44.44	40.74	43.43
Avg. Acc.	41.76	37.35	30.67	42.05	41.80	45.75	46.33	46.23	**47.63**
Avg. Error	58.24	62.65	69.33	57.95	58.20	54.25	53.67	53.77	**52.37**

Table 9: Comparison of the different prompting methods with CVP for every CIFAR-10-C corruption type. The Standard model is WideResNet18. Number in bold shows the best performance.

Severity / Method	Standard	BN	Finetune	VP		CVP			
				Patch	Append	fixed (3*3) w/o update	fixed (3*3) w/ update	rand. (3*3) w/o update	rand. (3*3) w/ update
S1	59.68	52.48	43.28	59.94	59.76	65.17	66.00	65.98	**68.07**
S2	47.88	43.18	34.72	48.26	47.94	52.73	53.42	53.41	**54.73**
S3	40.31	36.44	29.08	40.67	40.32	44.22	44.87	44.51	**45.96**
S4	32.75	29.49	24.66	32.94	32.75	36.20	36.74	36.56	**37.79**
S5	28.20	25.16	21.63	28.46	28.25	30.42	30.64	30.68	**31.61**
Avg. Acc.	41.76	37.35	30.67	42.05	41.80	45.75	46.33	46.23	**47.63**
Avg. Error	58.24	62.65	69.33	57.95	58.20	54.25	53.67	53.77	**52.37**
Avg Diff.	-	4.41	11.09	-0.29	-0.04	-3.99	-4.57	-4.46	**-5.87**

Table 10: Comparison of the different adaptation baslines with CVP for every severity on CIFAR-10-C. The Standard model is WideResNet18. Number in bold shows the best performance.

	WideResNet18 Avg. Error (%)
Standard	58.24
VP (patch)	57.94 (-0.3)
CVP (rand. w/ update)	**52.37 (-5.87)**
TTT [61]	52.92 (-5.32)
TTT + CVP	53.07 (-5.17)

Table 11: TTT [57] result for CIFAR-10-C

• We show the detailed results for each corruption on ImageNet-C dataset.

	Standard	Finetune	BN Adapt	VP		CVP			
				patch	append	fixed 3*3	rand 3*3	fixed 5*5	rand 5*5
Gaussian Noise	80.00	78.85	79.43	79.44	79.99	78.49	**78.16**	78.47	78.75
Shot Noise	82.00	80.80	81.57	81.56	81.97	80.45	**80.00**	80.10	80.81
Impulse Noise	83.00	81.80	82.72	82.72	83.00	80.80	**79.82**	80.88	81.40
Defocus Blur	73.58	75.49	75.32	77.27	73.56	74.13	**73.73**	74.29	75.10
Motion Blur	90.95	79.85	92.38	80.18	77.96	89.99	89.31	**89.14**	88.60
Glass Blur	76.32	87.16	76.86	90.41	88.98	75.98	75.47	75.99	**75.45**
Zoom Blur	80.00	79.84	80.52	82.07	79.96	79.87	79.67	79.72	**79.45**
Snow	43.86	45.10	45.82	47.88	88.07	**44.24**	44.27	44.60	44.91
Frost	79.88	81.04	82.22	83.78	**74.99**	80.12	80.19	79.69	80.05
Fog	74.38	75.74	76.80	78.06	**64.38**	74.53	74.91	74.36	74.85
Brightness	78.25	79.90	81.23	84.99	**47.95**	78.78	78.49	78.83	78.91
Contrast	71.00	73.27	74.14	76.59	**70.98**	71.46	70.83	71.31	71.79
Elastic Transform	87.58	87.96	88.89	96.50	87.62	87.93	87.85	**87.42**	88.15
Pixelate	74.72	74.22	75.75	78.30	74.68	67.05	**63.98**	67.29	64.15
Jpeg Compression	77.00	75.47	74.85	81.29	76.99	74.41	**73.46**	74.05	74.26
mCE	76.83	77.10	77.90	80.07	76.74	75.88	**75.34**	75.74	75.77
Diff.		0.27	1.07	3.24	-0.09	-0.95	**-1.49**	-1.09	-1.06

Table 12: ImageNet-C results. Number in bold shows the best performance.

• **Generalize to Cutout-and-Paste samples** To justify that our method can be generalized to non-structured OOD, we do more experiments on other types of OOD samples, such as the Cutout-and-Paste samples. Here, we launch the experiment on the **Waterbirds** dataset, which is constructed by cropping out birds from images with "water" backgrounds in the Caltech-UCSD Birds-200-2011 (CUB) dataset [59] and transferring them onto backgrounds from the Places dataset [70]. We follow the GitHub repo[2] and choose the "Forest" as our new background to generate the samples. The training of SSL is based on the pre-trained ResNet34 backbone model for the original CUB dataset. The original CUB (200 classes) accuracy for the backbone ResNet34 is 75.34%. We compare our CVP with self-supervised VP and demonstrate that CVP is more effective on the Cutouted-CUB dataset. The following table shows our results. Our CVP improves the result upon Standard by 1.61 points and VP by 1.3 points.

Cutouted-CUB (200)	**Before Adapt**	**VP (patch)**	**CVP**
Accuracy (%)	62.03	62.32	**63.64**
contrastive loss (Avg.)	2.78	2.71	**2.52**

Table 13: Performance on the Cutouted-CUB

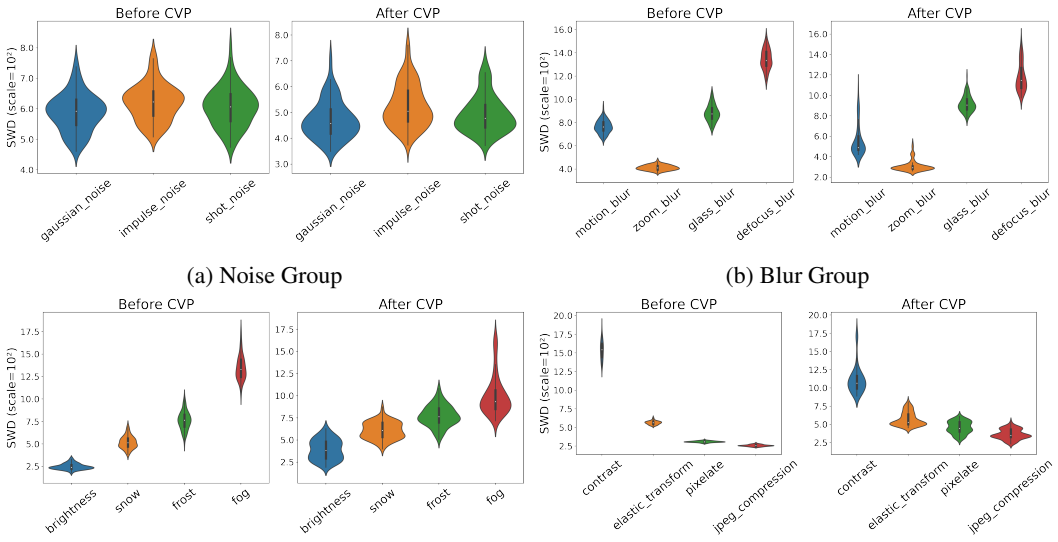
[2]WaterBirds Datasethttps://github.com/kohpangwei/group_DRO

7.5 Distance measurement with SWD and SSIM

We do the quantitative measurement on CVP by using the Sliced Wasserstein Distance (SWD) and structural similarity index measure (SSIM). To calculate the distance between two input distributions via the Sliced Wasserstein Distance, we first obtain a group of marginal distributions from a high dimensional probability distribution via the linear projection, then calculate the p-Wasserstein Distance for those marginal distributions. Here, we aim to measure the two input distributions: source domain distribution and target domain distribution (before/after adaptation). Table 14 and Figure 7 shows the result of SWD on CIFAR-10-C with severity 1. On average, CVP achieves lower SWD after adaptation, which means the target distribution is closer to the source one after adaptation. The average SWD reduce by 0.7% after prompting.

	SWD (scale: 10^2) ↓		SSIM ↑	
	before	after	before	after
Gaussian Noise	5.90	**4.71**	0.7242	**0.7849**
Shot Noise	6.08	**4.93**	0.7124	**0.7676**
Impulse Noise	6.23	**5.26**	0.7463	**0.7764**
Glass Blur	8.85	9.19	0.5873	0.5865
Defocus Blur	13.52	**11.82**	0.6031	0.6013
Zoom Blur	4.13	**3.09**	0.8726	0.8703
Motion Blur	7.68	**5.57**	0.6491	0.6459
Brightness	2.48	3.94	0.9702	0.9692
Snow	5.18	6.07	0.8258	**0.8275**
Frost	7.61	7.72	0.8025	0.8012
Fog	13.49	**9.99**	0.5840	0.5785
Contrast	15.39	**11.09**	0.7049	0.6997
Pixelate	3.09	4.56	0.8603	**0.8669**
Jpeg Compression	2.58	3.65	0.8681	**0.8710**
Elastic Transform	5.62	5.75	0.5272	**0.5789**
Avg. Mean	7.19	**6.49**	0.7539	**0.7884**
Avg. Std	4.05	**2.79**	0.1294	**0.7260**

Table 14: Results of Sliced Wasserstein Distance and Structural Similarity Index Measure on CIFAR-10-C (Severity 1).



(a) Noise Group

(b) Blur Group

(c) Weather Group

(d) Digital Group

Figure 7: Violin Plot of SWD for different corruption groups on CIFAR-10-C. The left figure of each subplot shows the SWD before adapting, and the right shows the SWD after adaptation

• **Distribution changes after applying the proposed CVP**

In the main paper, Figure 2, we show the distribution changes in different corruption types and severity. Here, in Table 15, we show the distribution shifts after applying our CVP by calculating the average loss. In general, the distribution moves back to the original distribution. We show the SSL average loss (before adapt/after CVP adapt) for four corruption types on severity 1,3,5 for CIFAR10-C. The average SSL loss for the original CIFAR10 is 1.26. For every corruption we show here, the average SSL loss after adaptation is lower than the loss before adaptation.

severity	**s1**	**s3**	**s5**
	Before /After	Before /After	Before /After
Gaussian noise	1.9 / 1.6	2.5 / 2.1	3.3 / 2.6
Defocus blur	3.2 / 2.9	3.4 / 2.8	3.7 / 3.1
Snow	3.1 / 3.0	3.8 / 3.3	3.9 / 3.5
Contrast	2.7 / 2.3	2.9 / 2.4	3.6 / 3.3

Table 15: Distribution changes on different corruption types.

7.6 The Effect of Different Prompt Designs

We do analysis on different prompting methods, including original visual prompts with different norm-bound (ℓ_2, ℓ_∞), convolutional prompts, and their combinations ($\ell_2 + conv.$, $\ell_\infty + conv.$). We show the error rate on different numbers of adapt iters for every prompting method from 0, 1, 5, to 10. To compare the results, we set up other parameters such as the epsilon ϵ as $8/255$ for ℓ_∞, 1 for ℓ_2. As Figure 8 shows the error rate for different prompting methods, the convolutional prompt $conv.$ and its combination with ℓ_2 reduce the error rate, and the former one reduces more from 40.32% to 36.08% when increasing the adapt iters. However, other prompting methods increase the error rate after prompting. To understand the risk of over-fitting for different prompting methods, Figure 4b shows the SSL loss curve v.s. performance on different prompting methods.

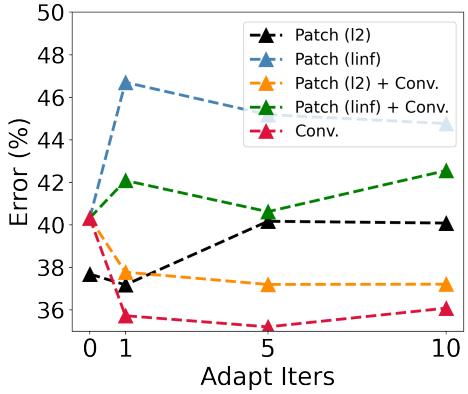


Figure 8

• **Training Cost v.s. Different Kernel Size**: In Table 16, we evaluate different kernel sizes for CVP and empirically discover that increasing the kernel to a proper size can improve the performance slightly. We choose one corruption-type impulse noise under severity 4 and show the results. When increasing the kernel size, the optimization cost increases. For impulse noise, kernel size 7*7 achieves the best robust accuracy, yet the optimization cost is much higher.

Kernel Size	**Accuracy (%)**	**# of Trainable Params.**	**Training Cost/Batch**
3*3	16.22	9	0.67s
5*5	16.3	25	0.68s
7*7	**16.62**	**49**	**1.24**s
13*13	16.61	169	1.28s
21*21	16.52	441	1.29s
25*25	15.4	625	1.32s

Table 16

• **Training time v.s. Number of Adapt Iteration** In the main paper, Figure 4(b), we have shown the CVP trained under different adapt iters v.s. their performance. When increasing the number of adapt iters, the training time increases. The following shows the result of CIFAR10-C on gaussian noise type with severity 1. We compare the accuracy and per batch training time on several numbers of adapt iters (from 0 to 20). While adapting with a few epochs (epoch number 1), we empirically found that CVP has a larger performance gain than VP.

# of Adapt Iters	**0**	**1**	**5**	**10**	**15**	**20**
Cost/Batch	0.00s	0.17s	0.67s	1.29s	1,.92s	2.57s
CVP Acc.(%)	39.51	51.09	56.1	58.76	59.30	59.58

Table 17

• **Low-rank Prompt Analysis**

In Table 18, we show the detailed results of low-rank prompt (LVP) on different severity (from 1 to 5) for CIFAR-10-C. We set up the same rank as 3 for LVP and CVP. Our results show that the CVP is more effective than LVP when reversing natural corruption. In Figure 9, we further plot the averaged contrastive loss on different rank sizes for both LVP and CVP. On every corruption type, while increasing the rank size from 3 to 31, the loss curves of LVP consistently drop, which demonstrates the LVP is much more easier to overfit the contrastive loss.

Severity / Method	Standard	LVP	CVP
s1	40.32	37.05	31.93
s2	52.12	48.83	45.27
s3	59.69	56.05	54.04
s4	67.25	63.42	62.21
s5	71.80	68.89	68.39
Avg. Error	58.24	54.85	**52.37**
Diff.		-3.39	**-5.87**

Table 18

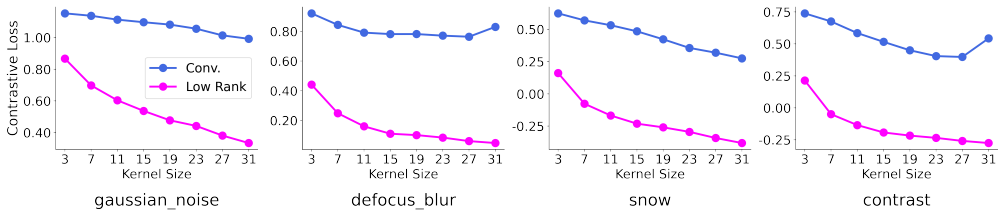
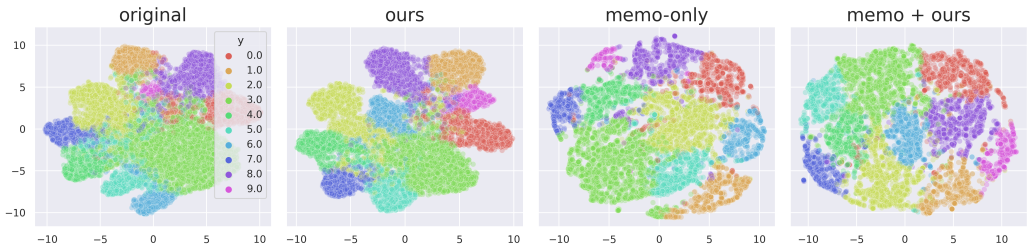


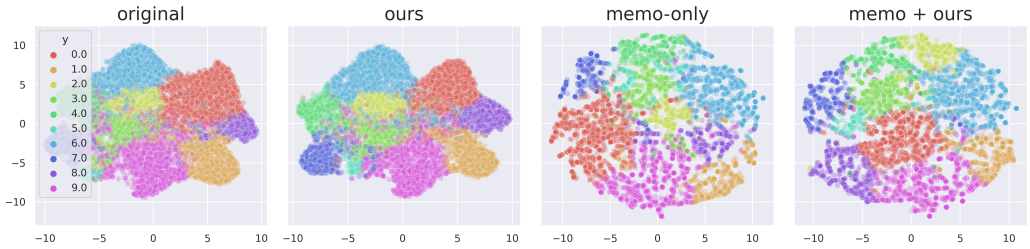
Figure 9

7.7 t-SNE analysis on different adaptation methods.

In addition to performing analysis on single sample, we further conduct the t-SNE visualization for whole sample distribution on different baseline. For each type of corruption data, we extract the 1-dimensional logit features in the last layer of model and calculate the distance between them with respect to the predicted class labels. We compare our method with standard, MEMO, and MEMO + Ours. As Figure 10 shows, the original feature embedding shows low separability between different classes. On the other hand, our approach clearly discriminates the embedding feature, which demonstrates its robustness against distribution shifts.



(a) Motion Blur

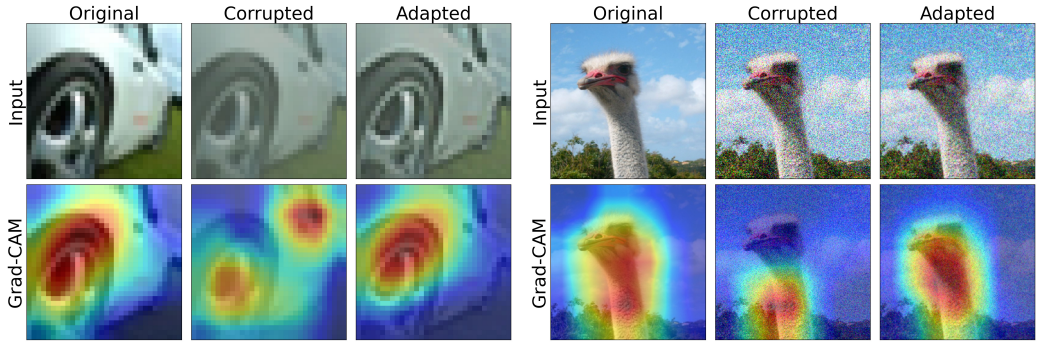


(b) Impulse Noise

Figure 10

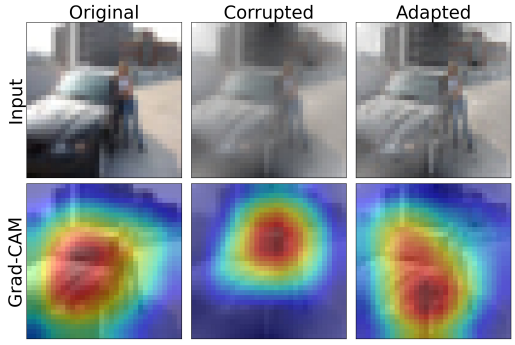
7.8 Saliency map analysis on different corruption types

To better understand how self-supervised visual prompts adapt to the corrupted inputs, we visualize the saliency map of different types of corruption. As Figure 11 shows, from left to right, 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 prompting, the red region of the adapted sample's heap map is re-target on the similar region as original image, which demonstrates that the self-supervised visual prompts indeed improve the input adaptation and make the model refocus back on the correct regions.

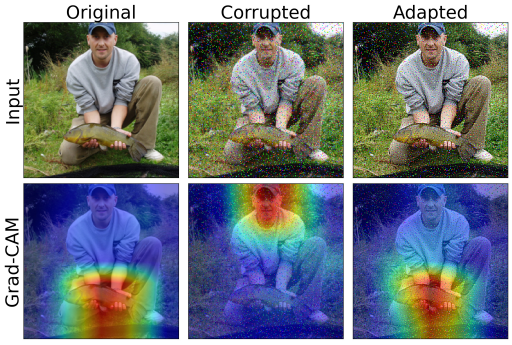


(a) Contrast

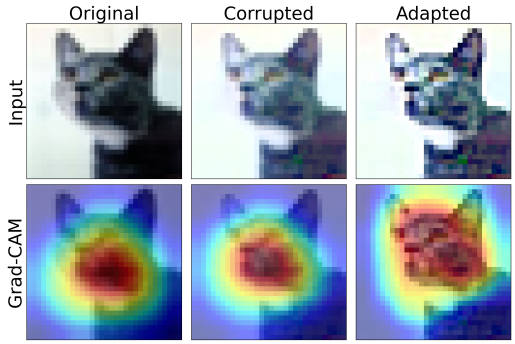
(b) Gaussian Noise



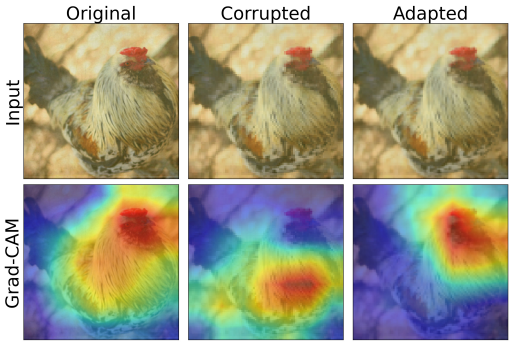
(c) Fog



(d) Impulse noise



(e) Brightness



(f) Pixelate

Figure 11: Grad-CAM analysis on different types of corruption.