Appendix

We provide transferability results of our self-supervised adversarial perturbations computed at lower perceptual budget ($\ell_\infty \leq 0.08$) for in-domain (Tables 8 & 9) and cross-domain (Table 10) settings in Appendix A. Our *Adversarial Pixel Restoration* approach remains effective as compared to the baseline [26] in fooling the Convolutional Networks, Vision Transformers as well as state-of-the-art input processing defense [32] (Fig. 8 & 9 in Appendix A.1). In appendix B and C, we visualize adversarial examples and analyze the attention shift cause by our attack, respectively. Finally, in appendix D, we provide psuedocode for self-supervised adversarial training of surrogate models using our approach.

A Adversarial Transferability under $\ell_\infty \leq 0.08$

Transformation	Method	VGG-19	Inc-V3	Res152	Dense121	SeNet	WRN	MNet-V2	Average
Jigsaw	[26]	40.00	58.20	55.66	50.30	66.62	59.52	34.60	52.13
	Ours	30.88	37.82	46.14	38.04	52.18	42.62	26.32	**39.14**(-12.99)
Rotation	[26]	38.88	56.16	57.06	49.56	65.30	58.14	34.64	51.39
	Ours	33.48	37.78	47.16	38.98	52.96	43.74	28.52	**40.37**(-11.02)
Prototypical	[26]	30.08	45.74	47.28	37.66	54.42	44.82	27.32	41.05
	Ours	30.44	31.96	42.76	34.50	49.20	38.80	23.94	**35.94**(-5.11)

Table 8: Comparative analysis of adversarial transferability. Results (top-1 (%), *lower is better*) are reported on 5k images from ImageNet validation set under the perceptual budget of $\varepsilon \leq 0.08$. Our attack provides favorable results as compared to [26].

Transformation	Method	Deit-T	Deit-S	ViT-T	ViT-S	Average
Jigsaw	[26]	51.32	68.26	46.68	68.16	58.61
	Ours	53.48	67.50	32.78	62.60	**54.09**(-4.52)
Rotation	[26]	53.22	68.36	47.94	66.72	59.06
	Ours	51.16	66.10	32.68	60.20	**52.54**(-6.52)
Prototypical	[26]	47.44	64.22	32.94	60.06	51.17
	Ours	45.70	62.26	29.50	57.98	**48.86**(-2.31)

Table 9: Comparative analysis of adversarial transferability for Vision Transformers on 5k images [26] from ImageNet validation set under the perceptual budget of $\varepsilon \leq 0.08$. ViTs are relatively more robust as compared to CNNs.

Transformation	Method (→)	Convolutional Networks		Vision Transformers	
		[26]	Ours	[26]	Ours
Rotation	CoCo	38.47	**35.72**	48.24	**46.44**
	Paintings	36.81	**30.48**	45.92	**42.04**
	Comics	62.46	**39.16**	70.39	**53.03**
Jigsaw	CoCo	51.75	**43.6**	61.95	**55.36**
	Paintings	51.56	**44.42**	61.59	**56.64**
	Comics	69.89	**52.65**	75.66	**62.15**

Table 10: Adversarial perturbations ($\varepsilon = 0.08$) are transferred from a single surrogate auto-encoder trained on CoCo, Paintings or Comics datasets to different Convolutional and Transformer based models. Results (top-1 (%) accuracy, *lower is better*) are averaged across all models. Details on these models are provide in Sec. 4

A.1 Adversarial Transferability against Neural Purification Defense

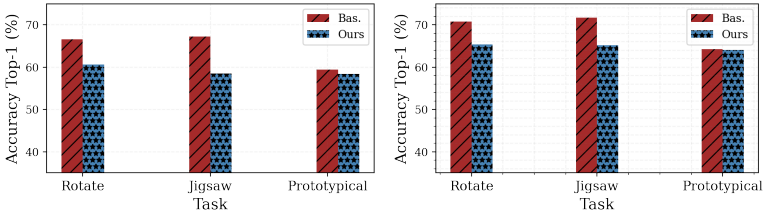

Figure 8: We test the vulnerability of the state-of-the-art input processing defense NRP [52] against our self-supervised attack. Adversarial perturbations are transferred from auto-encoders under the constraint of limited in-domain samples [76] to Convolutional Networks (*left plot*), and Vision Transformers (*right plot*) protected by NRP defense. We report Top-1 (%) (*lower is better*) averaged across the models. Our method consistently improves the attack success rate.

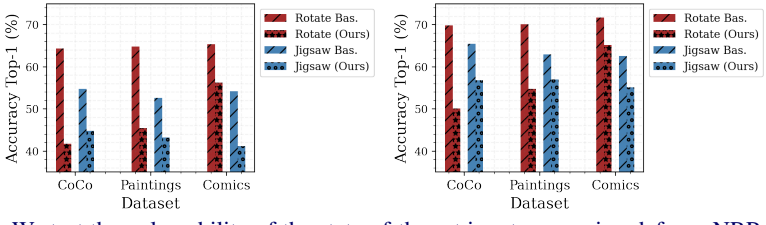

Figure 9: We test the vulnerability of the state-of-the-art input processing defense NRP [52] against our self-supervised attack under cross-domain setting (Sec. 4). Adversarial perturbations are transferred from single auto-encoder trained on CoCo, Paintings or Comics to Convolutional Networks (*left plot*), and Vision Transformers (*right plot*) protected by NRP defense. We report Top-1 (%) (*lower is better*) averaged across the models. Our method consistently improves the attack success rate.

B Adversarial Examples

Figure 10: Adversarial examples crafted on surrogate models trained on limited data, with perturbation bound $\varepsilon \leq 0.1$. The top and bottom row show adversarial examples crafted on surrogate model trained using [76] and ours method, respectively.

C Attention Dispersion

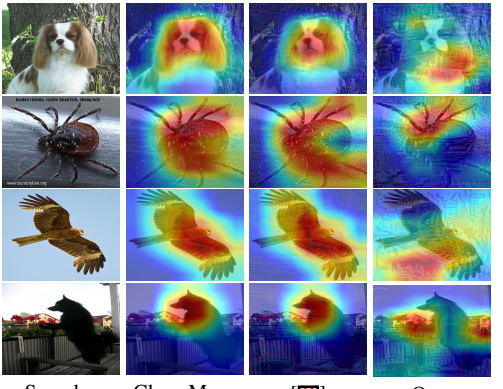

Figure 11: GradCAM [14] explanation of adversarial examples. The activation maps were generated on ImageNet pretrained ResNet50 model. Adversarial examples are transferred from surrogate trained via our adversarial rotation restoration task at the perceptual budget of $\varepsilon \leq 0.1$ [26]. Our approach significantly shifts the attention of the model, boosting the misclassifcation rates on the adversarial examples.

D Training Algorithm

Algorithm 2 Adversarial Pixel Restoration: Pytorch style Pseudocode

```
def train(model, optimizer, images, iterations, fgsm_step):
    """
    images:     images shaped (B, C, H, W)
    model:      autoencoder
    """
    attack = FGSM(model, eps=fgsm_step)
    target_images = images.clone()

    for i in range(iterations):

        # Generate adversarial images
        transformed_images = transform(images)
        adv_images = attack(transformed_images, target_images)

        # Get the encoder features and output
        clean_output, clean_enc_output = model(transformed_images)
        adv_output, adv_enc_output = model(adv_images)

        # Compute losses
        clean_loss = nn.MSELoss()(clean_output, target_images)
        adv_loss = nn.MSELoss()(adv_output, target_images)
        feat_loss = nn.MSELoss()(adv_enc_output, clean_enc_output)
        loss = clean_loss + adv_loss + feat_loss

        # Update model parameters
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```