AntiStyler: Defending Object Detection Models Against Adversarial Patch Attacks Using Style Removal

Supplementary Material

Abstract

This supplementary material provides additional information for the paper "AntiStyler: Defending Object Detection Models Against Adversarial Patches Using Style Removal". We start by expanding the related work section and providing more information on each patch attack used in our evaluation. Next, we provide implementation details on the target object detection models, the defenses compared, and the adversarial patch attacks used in our evaluation. Additionally, we provide implementation details regarding AntiStyler's random seed and hyperparameters for its filter and enhancement phases. Then, we expand our digital evaluation to include additional baselines and models, demonstrating AntiStyler's model-agnostic generalization. We also extend our ablation study to investigate: (1) the contribution of each of AntiStyler's enhancement phase filters; and (2) the impact of different attack characteristics (e.g., color, shape). Lastly, we extend our physical evaluation, present AntiStyler's performance on the Superstore dataset, explain the effect of style removal on adversarial regions, and elaborate on partial occlusion scenarios.

1. Related Work

In this section, we extend the related work section of the paper by reviewing the different attacks used in our evaluation. Examples for each attack (as well as the physical patches of Superstore and APRICOT and the adaptive patches) are provided in Figure 8.

1.1. Original Adversarial Patch (Google)

The original adversarial patch attack [3] was originally designed for the image classification task but later was adapted for the OD domain. This attack was defined as the following optimization problem:

$$\hat{P} = \arg\max_{P} \mathbb{E}_{x,t,l} \left[\log Pr(\hat{y}|h(A(P,x,l,t)))\right] \quad (9)$$

where $h(A(P,x,l,t))$ denote the model's prediction after a function applying the patch P to original image x at location l with transformation t and $Pr(\hat{y}|A)$ is the probability of classifying input A as class $\hat{y}$. By selecting the patch with the maximum expected $logPr(\hat{y}|A)$, the patch was optimized to be effective under multiple images, locations, and transformations, improving the patch's robustness to the real world.

1.2. Masked Projected Gradient Decent (M-PGD)

Unlike the Google attack, M-PGD (the masked variant of the standard PGD perturbation) [30] optimized the patch by calculating the derivative between the attacked image's prediction and the ground-truth annotation with respect to the patch area. This attack was defined as the following optimization problem:

$$P^{t+1} = clip(P^t + \alpha \nabla_{P^t} L(\hat{y}, h(A(P^t, x, l, t)), 0, 1) \quad (10)$$

where P^t is the patch from the previous optimization iteration, $\nabla_{P^t} L$ is the gradient of the loss between the ground truth annotation and the model's prediction on the attacked image, and clip is a function to clip the patch values to the range of (0,1). Using this formula, the attack can be targeted (minimizing the loss between the desired and current model predictions) and untargeted (maximizing the loss between the ground truth annotations and the current model predictions).

1.3. DPatch

DPatch attack [29] enhanced the Google patch for object detection models by optimizing the patch to harm the model's ability to predict the ground truth class and bounding box of the targeted object, as follows:

$$\hat{P}_{untargeted} = \arg\max_{P} \mathbb{E}_{x,s}[L(A(x,s,P); \hat{y}, \hat{B})] \quad (11)$$

$$\hat{P}_{targeted} = \arg\min_{P} \mathbb{E}_{x,s}[L(A(x,s,P); y_t, B_t)] \quad (12)$$

where x is a given image, $A(x,s,P)$ is a function that applies patch P on image x with shift s (uniformly sampled), $\hat{y}$ and $\hat{B}$ are targeted labels and bounding boxes, and y_t and B_t are ground truth labels and bounding boxes.

1.4. Printable Patches

To further optimize the adversarial patches for real-world use cases, Thys et al. used three loss components for the patch optimization process. The L_{tv} (Total Variation) loss was added to minimize the total variation in the patch's appearance to smooth color transitions and reduce noise. The L_{nps} (Non-printability Score) loss was added to ensure the patch maintains its colors after being printed by minimizing the distance between the patch colors and the achievable printer colors. The L_{obj} (Objectness) loss was added to reduce the maximum objectness score in the image, aiming to make the object detector miss objects in the image.

The total loss function for patch optimization was defined as follows:

$$L = \alpha L_{\text{nps}} + \beta L_{\text{tv}} + L_{\text{obj}} \tag{13}$$

where α and β are empirically determined scaling factors. By optimizing this loss function, the patches were optimized to be robust to real-world conditions where the position, angle, and visibility of the patch might vary.

1.5. Naturalistic Patches

Unlike previous attacks, Hu et al. focused on the appearance of the adversarial patch, stating that patches commonly possess conspicuous and attention-grabbing patterns that humans can easily identify. To address this issue, they used a pretrained GAN generator to restrict the space of generated adversarial patches, ensuring a naturalistic appearance while maintaining the adversarial effect.

1.6. TSEA

TSEA [16] was recently published as a transfer-based black-box attack on object detection. Unlike existing transfer-based attacks that rely on model ensembles, T-SEA leverages a single model with a self-ensemble technique, which utilizes data augmentation techniques and the ShakeDrop [48] model to help improve patch transferability.

2. Evaluation

In this section, we expand our evaluation by including additional implementation details, digital evaluation results on the RT-DETR2, SSD, YOLO3, YOLO4, and YOLO11 OD models, extended ablation study, and physical evaluation results on the Superstore dataset.

2.1. Extended Implementation Details

2.1.1. Target Object Detection models

For the OD models, we used the pre-trained Faster R-CNN and SSD models provided in the Torchvision library [1], the pre-trained DETR model provided in its GitHub [2], the pre-trained RT-DETR2 model provided in HuggingFace [3], the pretrained YOLO3 and YOLO4 models provided in the GitHub of the naturalistic patch attack, and the pretrained YOLO11 model provided in the Ultralytics library [4]. All models were pre-trained on COCO's training set.

2.1.2. Compared Defenses

For the compared defenses, we used the code and checkpoint weights provided in the GitHubs of LGS [5], GradDefense [6], SAC [7], ObjectSeeker [8], PAD [9], NAPGuard [10], DIFFender [11], NutNet [12], and KDAT [13].

2.1.3. Adversarial Patch Attacks

For the M-PGD, DPatch, and Google adversarial patch attacks against the COCO dataset, we generated adversarial patches using the corresponding attack classes in the Adversarial Robustness Toolbox (ART) library [33]. Each attack was configured according to the default settings in the ART library, creating both square- and circle-shaped patches. The original patch sizes (before placing them on the objects) were set to $\{100,120,150\}$. For each image, the patch was placed on the object with the highest confidence score, with an adjusted patch size of $\alpha * min(BB_{width}, BB_{height})$ where BB_{width}, BB_{height} are the object's bounding box width and height and α is a dynamic coefficient in the range of 30%-50%, resulting in patches of varying sizes and shapes (according to the object's bounding box size). After generating the patches, we excluded images in which the patch only caused partial occlusion without any adversarial effect by comparing the predictions on the attacked image with and without a black mask on the patched area. We generated approximately $\sim$ 300 patches for each attack, and we are working on separately publishing the entire dataset for the research community, which is available upon request.

For the adaptive patch attacks, we utilized the M-PGD as the base attack (using the same default settings in the ART library), modifying its optimization process to incorporate the relevant components at each adaptive level. For the black-box adaptive patch, we added a loss component to minimize the total variation of the patch as follows:

$$L_{smooth}(patch) = \alpha L_{MPGD}(patch) + \beta L_{TV}(patch) \tag{14}$$

where L_{MPGD} is the original loss for generating the M-PGD patch, and L_{TV} calculates the total amount of variation in value between neighboring pixels in the patch. α and β are positive hyperparameters that control the trade-off between the two losses. For the gray-box adaptive attack, we applied AntiStyler's AntiStyle model on the applied image (i.e., the image with the applied patch) before calculating the optimization's loss to include AntiStyle's gradients in the optimization. For the white-box adaptive attack, we applied the whole pipeline of AntiStyler on the applied image before calculating the optimization's loss to include AntiStyler's gradients in the optimization.

[1] https://docs.pytorch.org/vision/stable/index.html
[2] https://github.com/facebookresearch/detr
[3] https://huggingface.co/docs/transformers/model_doc/rt_detr_v2
[4] https://www.ultralytics.com/
[5] https://github.com/fabiobrau/local_gradients_smoothing
[6] https://github.com/UMBCvision/Contextual-Adversarial-Patches
[7] https://github.com/joellliu/SegmentAndComplete
[8] https://github.com/inspire-group/ObjectSeeker
[9] https://github.com/Lihua-Jing/PAD
[10] https://github.com/wsynuiag/NAPGaurd
[11] https://github.com/kkkcx/DIFFender
[12] https://github.com/0502tonylin/NutNet
[13] https://github.com/Yarinyl/KDAT

For the transferable [16], printable [42], and naturalistic [15] patches against the INRIA dataset, we used the original code and patches from the corresponding papers [14] [15] [16]. These patches were positioned on each person in the image, with a dynamically resized printable patch covering the center of each person's detection (based on the paper's original code). As a result, multiple patches were placed in each image in the cases where more than one person appeared.

2.1.4. Setting the Random Seed

To allow replicating AntiStyler's experimental results, we set the seed to 123 using the commands: `environ['PYTHONHASHSEED'] = 123` from the OS module [17], `backends.cudnn.deterministic = True` and `backends.cudnn.benchmark = False` from the Torch package [18] and `seed_everything(123)` from the Lightning package [19].

2.1.5. AntiStyler's Hyperparameters

For the COCO dataset, the top-change filtering percentage (τ) was set to 0.99, the kernel size for the dilation and erosion filters was set to 11, the kernel size for the smoothing filter was set to 51, and the final threshold was set to 0.5. Note that these parameters were obtained according to a validation set containing images that did not appear in the test set and were attacked only by the PGD attack.
For the INRIA dataset, the τ was set to 0.9575, the kernel size for the dilation and erosion filters was set to 11, the kernel size for the smoothing filter was set to 5, and the final threshold was set to 0.6. Note that these parameters were obtained according to a validation set containing images that did not appear in the test set and were attacked only by the OBJ, P1, and P2 patches.
For the Superstore dataset, τ was set to 0.94, the kernel size for the dilation and erosion filters was set to 11, the kernel size for the smoothing filter was set to 51, and the final threshold was set to 0.5.
For the APRICOT dataset, τ was set to 0.99, the kernel size for the dilation and erosion filters was set to 11, the kernel size for the smoothing filter was set to 5, and the final threshold was set to 0.1. Note that the parameters were obtained for both datasets according to a validation set containing images that did not appear in the test set.

2.2. Extended Digital Evaluation

Table 6 presents the extended results of Table 1 from the main paper (including results for LGS [32], Grad-

[14] https://github.com/VDIGPKU/T-SEA
[15] https://gitlab.com/EAVISE/adversarial-yolo
[16] https://github.com/aiiu-lab/Naturalistic-Adversarial-Patch
[17] https://docs.python.org/3/library/os.html
[18] https://pytorch.org/
[19] https://lightning.ai/docs/pytorch/stable/

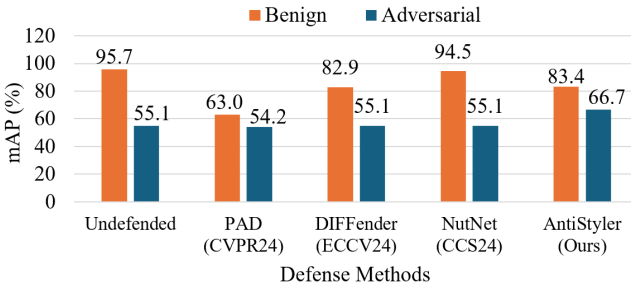

Figure 5. map% for physical patch attacks on the Superstore dataset and the Faster R-CNN OD model (higher is better).

Defense [38], SAC [27], and NAPGuard [44]). As can be seen, AntiStyler's performance remains superior in most cases (5/6). In addition, to further demonstrate AntiStyler's model-agnostic capabilities, we evaluate AntiStyler on seven OD models: (1) Faster R-CNN, (2) DETR, (3) RT-DETR2, (4) SSD, (5) YOLO3 (6) YOLO4, and (7) YOLO11. Table 7 presents the results of our evaluation against the transferable, printable, and naturalistic attacks on the INRIA dataset. As can be seen, across all models and most attack types, AntiStyler improved robustness significantly while preserving benign performance. These results confirm the generalizability and effectiveness of AntiStyler as a post-training defense and validate its model-agnostic capabilities.

2.3. Extended Physical Evaluation

Unlike the APRICOT dataset (in which evaluation focused only on adversarial robustness), the Superstore dataset also contains benign images, allowing us to evaluate a defense's performance on both benign and adversarial examples on real-world physical images. Superstore's target model is Faster R-CNN OD, and we use the test set for evaluation. Figure 5 presents the results of AntiStyler compared to the other defenses on the Superstore dataset. As can be seen, all the defenses reduce the benign performance, with AntiStyler achieving second-best benign performance. On the other hand, compared to the other defenses, which did not effect the adversarial performance, AntiStyler improves the adversarial performance, thus obtaining the best overall performance.

2.4. Extended Ablation Study

2.4.1. One-Step Ablation Study

In addition to the study presented in the main paper, we also conducted a one-step ablation study to evaluate the contribution of each layer in the enhancement phase. Table 8 presents the results of one-step ablated variants of AntiStyler (**D**ilation, **E**rosion, **S**moothing, **T**hresholding) against the M-PGD, Google, and DPatch attacks on the Faster R-CNN model and the COCO dataset. As can be

OD Model	Defense Method	Processing Time [ms]	Google			M-PGD			DPatch		
			Benign	Adv	Mean	Benign	Adv	Mean	Benign	Adv	Mean
Faster RCNN	Undefended	43.2	51.6	16.6	34.1	45.8	22.3	34.1	49.4	23.0	36.2
	LGS (WACV19)	713.8	42.5	26.8	34.7	38.0	30.8	34.4	43.2	27.7	35.5
	Grad-Defense (CVPR20)	**43.2**	46.7	22.9	34.8	47.1	28.2	37.6	47.8	28.6	38.2
	SAC (CVPR22)	78.0	**51.6**	22.7	37.2	45.8	35.5	40.7	49.4	31.5	40.5
	ObjectSeeker (SP23)	4191.9	**51.6**	31.0	41.3	44.5	28.4	36.5	48.0	28.4	38.2
	PAD (CVPR24)	55098.8	39.8	27.2	33.5	39.0	36.0	37.5	43.0	35.9	39.5
	NAPGuard (CVPR24)	384.6	47.8	27.0	37.4	43.9	35.4	39.7	46.2	32.8	39.5
	DIFFender (ECCV24)	7606.0	34.1	19.4	26.8	32.1	26.3	29.2	35.8	25.4	30.6
	NutNet (CCS24)	45.2	42.4	21.4	31.9	36.7	31.5	34.1	41.5	31.1	36.3
	KDAT (AAAI25)	**43.2**	50.1	31.5	40.8	**47.6**	33.3	40.5	**49.9**	34.3	42.1
	AntiStyler (Ours)	92.7	**51.6**	**32.5**	**42.1**	45.8	**38.0**	**41.9**	48.9	**38.3**	**43.6**
DETR	Undefended	39.3	52.8	30.8	41.8	53.6	29.0	41.3	56.8	35.5	46.2
	LGS (WACV19)	705.8	45.3	36.0	40.7	48.2	36.3	42.3	47.3	43.8	45.6
	Grad-Defense (CVPR20)	**39.3**	49.9	37.4	43.7	49.8	41.2	45.5	53.0	40.5	46.8
	SAC (CVPR22)	71.7	51.9	34.6	43.3	53.5	32.9	43.2	56.7	43.1	49.9
	ObjectSeeker (SP23)	4072.4	52.8	35.0	43.9	50.5	36.2	43.4	56.8	40.8	48.8
	PAD (CVPR24)	54721.5	50.6	33.7	42.2	46.1	40.2	43.2	55.2	**46.3**	50.3
	NAPGuard (CVPR24)	378.9	50.8	39.9	45.4	50.6	**41.8**	46.2	56.5	46.0	**51.3**
	DIFFender (ECCV24)	7603.4	39.0	23.6	31.3	38.3	26.9	32.6	41.1	32.2	36.7
	NutNet (CCS24)	42.8	50.7	36.6	43.7	49.2	40.2	44.7	51.9	45.5	48.7
	KDAT (AAAI25)	**39.3**	**53.0**	37.8	45.4	52.7	37.9	45.3	55.9	45.5	50.7
	AntiStyler (Ours)	85.6	**53.0**	**41.7**	**47.4**	**53.6**	39.6	**46.6**	**57.8**	44.4	51.1

Table 6. Extended evaluation of adversarial defenses for OD models under different patch attacks on the COCO dataset.

Model	Defense	Benign	TSEA		Printable Patches			Natural Patches			
			B-Patch	C-Patch	OBJ	UPPER	CLS-DET	P1	P2	P3	P4
Faster R-CNN	Undefended	94.9	17.6	17.3	45.6	40.2	59.7	54.6	66.2	53.1	67.4
	AntiStyler	**95.8**	**94.9**	**87.8**	**82.0**	**83.0**	**80.2**	**74.4**	**81.9**	**57.3**	**81.7**
DETR	Undefended	**91.8**	23.3	30.6	47.1	39.5	64.7	**65.5**	67.3	**46.5**	69.8
	AntiStyler	90.0	**90.0**	**87.5**	**74.7**	**74.1**	**74.7**	61.0	**74.6**	41.6	**73.1**
RT-DETR2	Undefended	**94.8**	38.9	45.7	55.2	59.1	56.2	51.8	60.4	49.9	62.8
	AntiStyler	94.5	**89.8**	**88.4**	**73.3**	**74.7**	**70.7**	**67.1**	**67.0**	**52.6**	**70.1**
SSD	Undefended	**57.7**	23.7	14.4	15.3	18.5	22.6	16.8	28.5	18.8	**40.1**
	AntiStyler	56.7	**79.4**	**71.1**	**41.0**	**44.5**	**40.6**	**24.8**	**33.6**	**19.9**	37.9
YOLO3	Undefended	75.6	1.7	2.2	9.9	20.2	38.9	24.6	20.5	44.3	43.5
	AntiStyler	**77.3**	**54.0**	**47.0**	**60.8**	**62.1**	**56.4**	**42.9**	**36.6**	**48.8**	**54.7**
YOLO4	Undefended	**77.9**	0.6	1.5	2.6	11.7	20.6	15.4	22.1	31.0	4.6
	AntiStyler	74.7	**30.1**	**31.7**	**35.6**	**40.1**	**34.2**	**31.9**	**32.2**	**36.6**	**29.2**
YOLO11	Undefended	**85.4**	54.0	**57.8**	44.2	30.1	47.1	**40.3**	**38.1**	**36.0**	43.0
	AntiStyler	82.4	**64.3**	56.1	**52.0**	**53.2**	**48.3**	39.1	35.2	33.3	**43.2**

Table 7. Extended evaluation of AntiStyler for OD models under patch attacks on the INRIA dataset.

Defense Method	Google			M-PGD			DPatch		
	Benign	Adv	Mean	Benign	Adv	Mean	Benign	Adv	Mean
Undefended	51.6	16.6	34.1	45.8	22.3	34.1	49.4	23.0	36.2
RM-AntiStyler	48.1	30.4	39.3	41.5	35.3	38.4	45.5	35.2	40.4
RM-AntiStyler + D	32.8	18.5	26.7	26.6	23.5	25.1	32.9	23.2	28.1
RM-AntiStyler + D + E	49.1	26.0	37.6	40.8	35.8	38.3	46.1	36.2	41.2
RM-AntiStyler + D + E + S	47.7	32.4	40.1	43.6	34.8	39.2	46.9	36.2	41.6
RM-AntiStyler + D + E + S + T	50.7	30.1	40.4	45.7	37.1	41.4	48.8	37.8	43.3
AntiStyler	**51.6**	**32.5**	**42.1**	**45.8**	**38.0**	**41.9**	**48.9**	**38.3**	**43.6**

Table 8. mAP% for one-step ablated variants of AntiStyler on the COCO dataset and the Faster R-CNN OD model.

seen, while AntiStyler's whole pipeline achieved the best performance, the erosion step (RM-AntiStyler + D + E) was the most impactful, boosting both benign and adversarial performance by 14 mAP% compared to the AntiStyler's variant with only dilation (RM-AntiStyler + D). This can be explained since this step removes small unconnected masked regions, reducing unnecessary masking and focusing on relevant areas (Sec. 4.4). In addition, the results of RM-AntiStyler (AntiStyler's variant which does not include the enhancement phase) show that it is possible to apply AntiStyler without needing to improve the raw mask, which further highlights the effectiveness of detecting and masking the attacks only based on identifying pixels associated with the random style.

2.4.2. Patch Appearance Ablation Study

We further evaluated AntiStyler 's robustness against patches of varying shapes and appearances, including square patches, circular patches, and background-blended patches (i.e., patches that attempt to blend into the background appearance). Results show that AntiStyler improves performance across all cases, increasing robustness from 23.9% to 39.5% for square patches, from 22.6% to 30.5% for circular patches, and from 30.3% to 36.5% for background-blended patches. These gains indicate that AntiStyler remains effective even when the patch geometry varies or blending techniques are applied, highlighting its robustness to diverse patch designs.

3. Discussion

In this section, we expand on the Discussion section of the paper by explaining the effect of style removal on adversarial regions relative to benign ones, and further elaborating on AntiStyler's limitation of partial occlusion.

3.1. Style Removal Effect

We conducted an experiment (Figure 6) demonstrating that adversarial Regions of Interest (ROIs) induce Gram matrices (GMs) with higher magnitudes than those induced by benign ROIs, supported by prior studies [19, 37]. Since style transfer minimizes the distance between the feature GMs of a target style and an input image [11], we reversed this process and sought a style whose GMs are similar to those of adversarial ROIs, while avoiding constraining AntiStyler to a specific attack style. Consequently, we found that random-noise style, defined by sampling a random-noise image, extracting its features from VGG19's first four convolutional layers, and computing their GMs, yields similarly high magnitude GMs. As a result, by attempting to maximize the distance between the input image's style and the random style, AntiStyler effectively reduces the magnitude of the input ROI GMs, with a more pronounced reduction on adversarial ROI GMs than benign ones.

3.2. Partial Occlusion

As presented in the paper, AntiStyler maintained the high benign performance of the undefended Faster R-CNN OD model. However, an interesting observation was made when AntiStyler was applied to DETR, as it not only maintained but also slightly improved its benign performance. In this section, we explore this observation by comparing images in which the DETR-only prediction was improved after applying AntiStyler. As shown in Figure 7, while in most images AntiStyler does not affect the benign image, in those where it does, it primarily masks noisy textures, such as vegetation and vibrant fabrics. Interestingly, due to these "mistaken" partial occlusion masks, AntiStyler slightly improved DETR's benign performance, with effects ranging from correcting misclassified objects to detecting previously unnoticed objects and improving existing object bounding boxes. Note that in all these examples, the "mistaken" mask is often far from the affected (and improved) object detections, thus not only AntiStyler does not negatively affect by partial occlusion, in some cases it even improves the base model's detections.

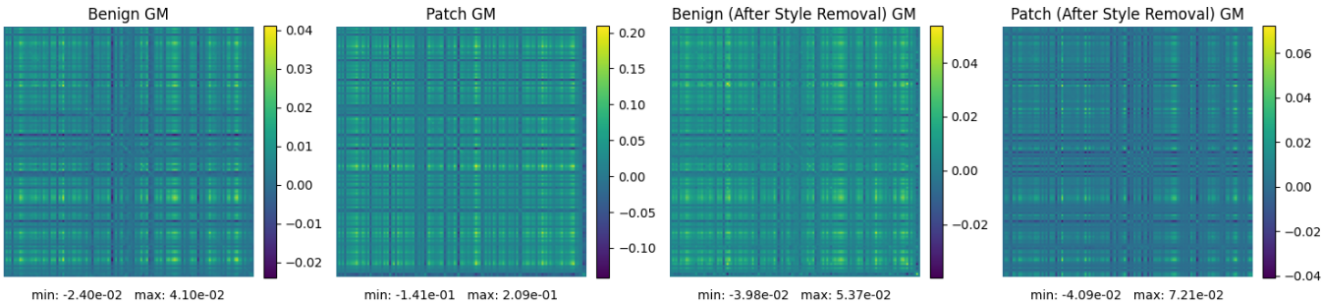

Figure 6. Gram matrix comparison between benign and adversarial regions of interest.

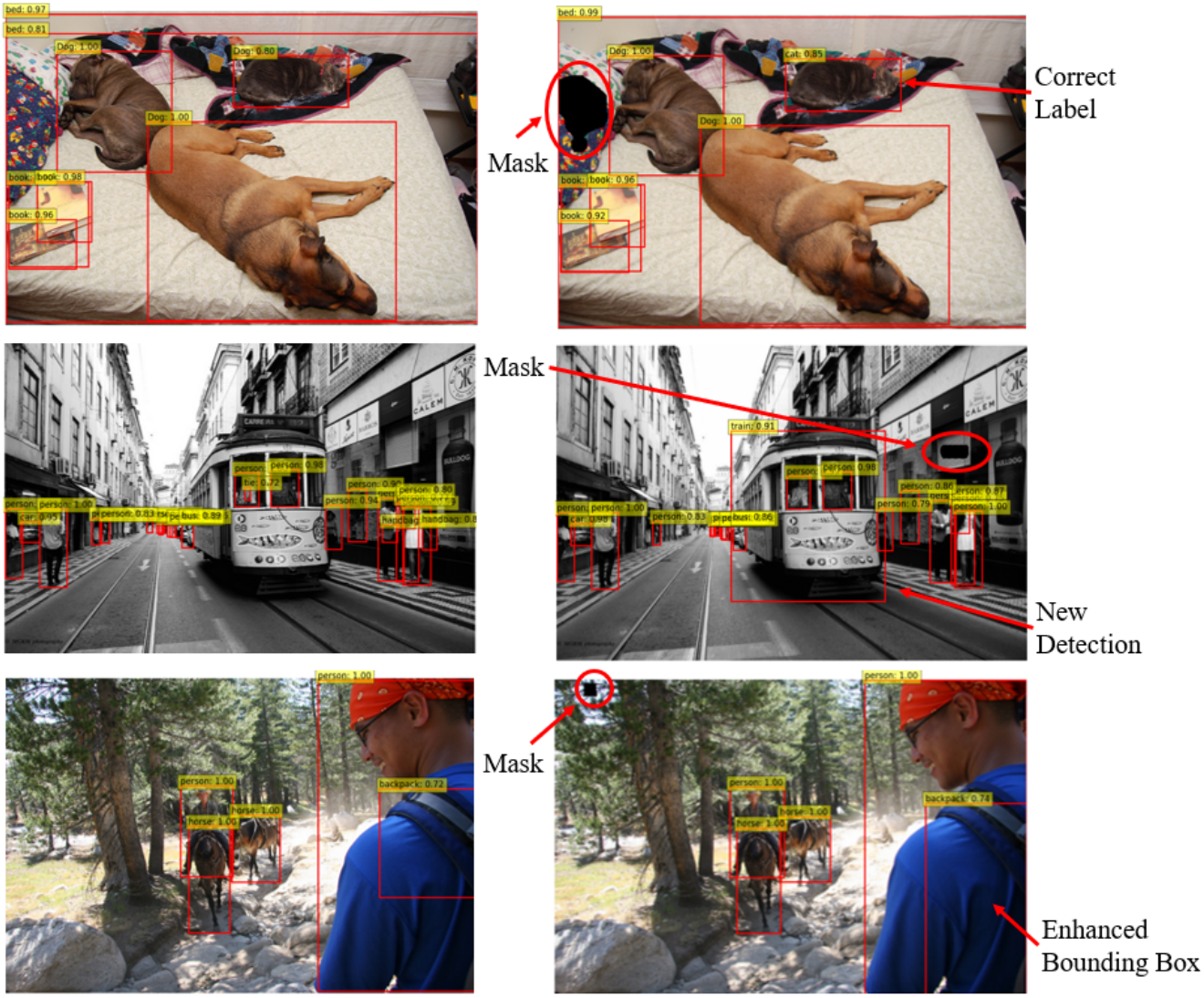

Figure 7. Examples for AntiStyler's partial occlusion scenarios.

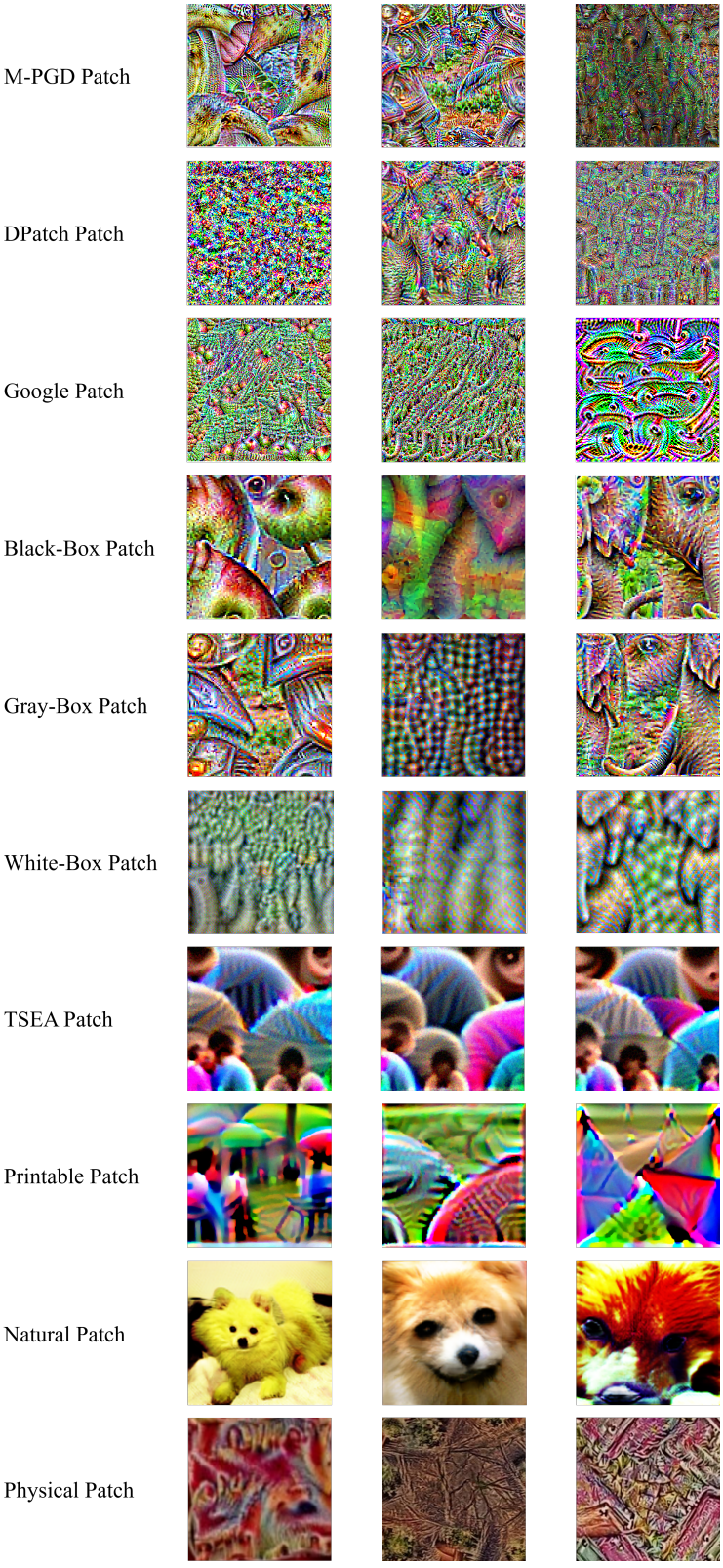

Figure 8. Visualization examples for each adversarial patch attack used in our evaluation.