

# PBCAT: Patch-Based Composite Adversarial Training against Physically Realizable Attacks on Object Detection

## Supplementary Material

### A. Theoretical Cost of Different Patch-Based AT Methods

Patch-based AT methods originally designed for classifiers [34, 36, 50] incur significant computational overhead when applied to object detectors due to the presence of multiple target objects per image. To quantitatively compare their time complexity, we analyze the number of forward/backward passes required per optimization iteration in Tab. A1 and provide detailed explanations below. Notably, the theoretical computational overhead is proportional to the number of passes.

1. AT-LO [36]: This method selects a single patch per image and performs inference over  $k$  steps. In each step, it evaluates  $t$  candidate positions and selects the one with the highest loss value. Once the optimal position is determined, a forward/backward pass is conducted to update the patch. When extended to object detection, the computational cost scales linearly with the number of target objects  $n$ .
2. DOA [50]: This method also selects a single patch per image. It first performs  $t$  steps to determine the patch location, followed by  $k$  PGD steps to generate the adversarial patch. The computational cost scales by  $n$  when adapted to object detection.
3. MAT [34]: This method selects  $p$  candidate patches, applies them to the image, and computes the loss. It then updates the patch using I-FGSM [23], requiring a total of  $p + 1$  steps per iteration. When applied to object detectors, the cost increases proportionally with  $n$ .
4. PBCAT (Ours): Unlike the above methods, PBCAT is designed for object detection. It requires only a single forward/backward pass with FreeAT [41], irrespective of the number of targets, and thus it significantly reduces computational overhead.

Method	Forward/Backward Times
AT-LO [36]	$n \times k \times (1 + t)$
DOA [50]	$n \times (t + k)$
MAT [34]	$n \times (p + 1)$
PBCAT (Ours)	1

Table A1. The number of forward/backward pass required for one optimization step between different patch-based AT methods.

### B. Pseudo-code of PBCAT

The pseudo-code of “Free” PBCAT for object detection is shown in Algorithm 1.

---

#### Algorithm 1 “Free” PBCAT on object detection

---

**Require:** Dataset  $\mathcal{D}$ ,  $l_\infty$ -bounded global perturbation intensity  $\epsilon$ , patch perturbation step size  $\alpha$ , patch perturbation intensity  $\beta$ , replay parameter  $r$ , model parameters  $\theta$ , epoch  $N_{\text{ep}}$

```

1: Initialize  $\theta$ 
2:  $\delta_g \leftarrow \mathbf{0}$ ;  $\delta_p \leftarrow \mathbf{0}$ ;  $\delta \leftarrow \mathbf{0}$ 
3: for epoch = 1, ...,  $N_{\text{ep}}/r$  do
4:   for minibatch  $B \sim \mathcal{D}$  do
5:     for  $i = 1, \dots, m$  do
6:       Compute gradient of loss with respect to  $\mathbf{x}$ 
7:        $\mathbf{g}_{\text{adv}} \leftarrow \mathbb{E}_{\mathbf{x} \in B} [\nabla_{\mathbf{x}} \mathcal{L}_d(f_\theta(\mathbf{x} + \delta), y)]$ 
8:       Update the model parameter
9:        $\mathbf{g}_\theta \leftarrow \mathbb{E}_{\mathbf{x} \in B} [\nabla_{\theta} \mathcal{L}_d(f_\theta(\mathbf{x} + \delta), y)]$ 
10:      update  $\theta$  with  $\mathbf{g}_\theta$  and the optimizer
11:      Update the patch and global perturbation
12:       $\delta_g \leftarrow \delta_g + \epsilon \cdot \text{sign}(\mathbf{g}_{\text{adv}})$ 
13:       $\delta_p \leftarrow \delta_p + \alpha \cdot \text{sign}(\mathbf{g}_{\text{adv}})$ 
14:       $\delta_g \leftarrow \text{clip}(\delta_g, -\epsilon, \epsilon)$ 
15:       $\delta_p \leftarrow \text{clip}(\delta_p, -\beta, \beta)$ 
16:      Generate mask  $\mathbf{M}$  by the steps in Sec. 3.3
17:       $\delta \leftarrow \text{Apply}(\delta_p \odot \mathbf{M}, \mathbf{x}) + \delta_g$ 
18:    end for
19:  end for
20: end for
```

---

### C. Additional Training Settings

Following Li et al. [29], we initialized the backbone of the detector using the adversarially pre-trained model provided by Salman et al. [40] and trained the detector on MS-COCO for 48 epochs using the AdamW optimizer with an initial learning rate of 0.0001. The learning rate followed a multi-step decay schedule, decreasing by a factor of 0.1 after the 40th epoch. During training, input images were resized such that their shorter side was 800 pixels, while the longer side was at most 1333 pixels.

It is important to note that successful adversarial training on object detectors requires adversarially pre-trained backbones (APB), such as an adversarially trained ResNet-50 on the large-scale ImageNet-1K dataset [11]. However, the backbones of current YOLO-series models are custom-

Method	Faster-RCNN	FCOS	DN-DETR
$l_\infty$ -AT (AdvOD) [29]	34h	26h	32h
PBCAT (Ours)	44h	38h	48h

Table A2. The comparison between training time (in hours) of PBCAT and AdvOD.

Method	PGDPatch	
	All	Person
Vanilla	18.4	17.5
$l_\infty$ -AT (AdvOD) [29]	30.7	30.1
PBCAT (Ours)	<b>37.8</b>	<b>34.5</b>

Table A3. The detection accuracies ( $AP_{50}$ ) of models trained with various methods on MS-COCO.

designed networks rather than widely adopted architectures like ResNet-50, and no APB checkpoints are available for these models. Training an APB from scratch on ImageNet-1K is beyond the scope of this work. Nonetheless, we believe that PBCAT could also enhance the robustness of YOLO detectors given sufficient computational resources or available APB checkpoints.

We compared the training time of PBCAT with Li et al. [29] in Tab. A2. All experiments were conducted on 8 NVIDIA 3090 GPUs. The slight increase in training time for PBCAT is primarily due to the additional computation required for patch location selection, which is currently implemented on the CPU. This overhead could be further reduced by parallel implementation on the GPU. Overall, we conclude that PBCAT incurs a training cost comparable to standard adversarial training.

## D. Additional Results and Details

### D.1. Additional Results on MS-COCO

Tab. A3 shows the results of models trained with various methods on MS-COCO. “All” and “Person” in Tab. A3 denote the averaged  $AP_{50}$  on each object category and the  $AP_{50}$  on the particular person category, respectively.

### D.2. Implementation Details of Baselines

For non-AT defense methods with publicly available source code [32, 44, 49, 54, 55, 60], we downloaded and modified the implementations to conduct adaptive attacks. Specially, we used the official checkpoints provided in the public repositories of SAC<sup>3</sup> and NAPGuard<sup>4</sup>. In addition, we used the Matlab code<sup>5</sup> to evaluate Jedi [44].

For non-AT defense methods without released source code [19, 21, 35, 53], we reproduced their implementa-

<sup>3</sup><https://github.com/joellliu/SegmentAndComplete>

<sup>4</sup><https://github.com/wsynuiag/NAPGaurd>

<sup>5</sup><https://github.com/ihsenLab/jedi-CVPR2023>

Method	T-SEA [18]	NatPatch [15]
Vanilla	31.7	54.4
PBCAT (Ours)	<b>90.9</b>	<b>86.3</b>

Table A4. The detection accuracies ( $AP_{50}$ ) of the Faster R-CNN model on transfer-based patch attacks on the Inria dataset.

Source \ Target			
	Faster-RCNN	FCOS	DN-DETR
Faster-RCNN	77.6	80.7	83.1
FCOS	80.0	58.0	79.3
DN-DETR	69.2	59.9	56.3

Table A5. The detection accuracies ( $AP_{50}$ ) under AdvPatch on the Inria dataset in the transfer-based attack setting. The adversarial examples generated on the source models were fed into the target models.

Ratio	AdvPatch	AdvTexture	AdvCaT
Top-30%	51.3	43.7	52.6
Top-70%	63.0	19.0	43.7
Top-50%	<b>77.6</b>	<b>60.2</b>	<b>56.4</b>

Table A6. The detection accuracies ( $AP_{50}$ ) of models trained with different ratio of sub-patches.

tions based on the descriptions in their respective papers. Note that when reproducing AD-YOLO [19] we replaced its YOLO model with Faster R-CNN for a fair comparison. Other experimental settings closely followed those in the original works.

For SOTA  $l_\infty$  AT baseline AdvOD, we directly used the released checkpoint from Li et al. [29]<sup>6</sup>. For MTD, we reproduced it using the code from Li et al. [29] without using adversarially pre-trained backbones.

### D.3. Visualization Results of the Detector

Some visualization results of the Faster R-CNN trained with PBCAT are shown in Fig. A1. We can see that the detector performed quite well under the large-area and strong adversarial texture attacks.

### D.4. Robustness against Transfer Attacks

To further validate that the models trained with PBCAT can have strong robustness against more unseen physically realizable attacks, we additionally evaluated two transfer-based patch attacks [15, 18] on the model trained with PBCAT. The adversarial patches were generated based on their original settings on the vanilla (clean) Faster R-CNN model. The patches were applied to the images in the Inria dataset to evaluate the  $AP_{50}$  of Faster R-CNN trained by PBCAT. The results are shown in Tab. A4. We can see that these

<sup>6</sup><https://github.com/thu-ml/oddefense>

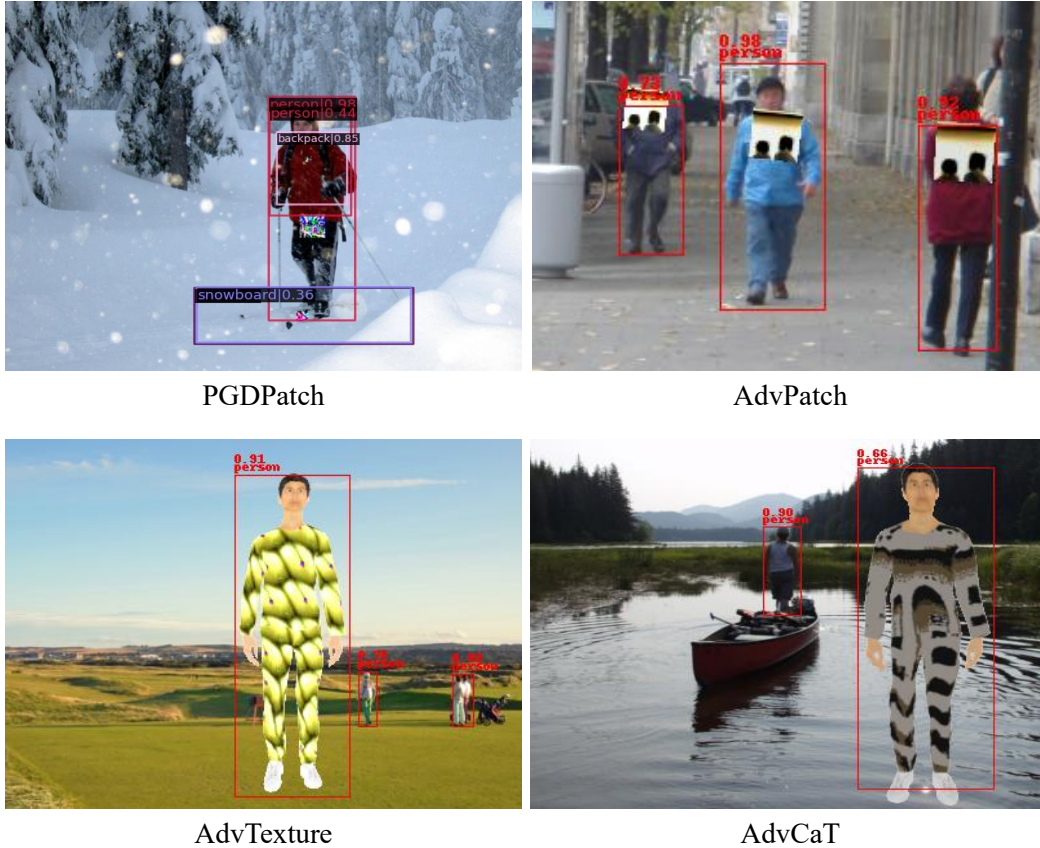


Figure A1. The detection results of the model trained with PBCAT under various physically realizable attacks. The detected bounding boxes with confidence larger than 0.5 are visualized.

transfer-based patch attacks almost lose the ability to fool the detectors trained with PBCAT.

Additionally, we used the three types of detectors we trained in this work (as discussed in Sec. 4.4), Faster R-CNN, FCOS [46], DN-DETR [25], to perform the black-box transfer attacks. Here we used the AdvPatch attack on the Inria dataset. The results are shown in Tab. A5. We can also observe that the models trained with our PBCAT can defend these black-box transfer-based attacks better than white-box attacks.

### D.5. Ablation on Top- $k$ Selection

In Sec. 3.3, we select the top 50% of sub-patches based on their height values. To investigate the effect of this design choice, we conduct an ablation study varying the top- $k$  selection ratios. As shown in Tab. A6, choosing the top 50% achieves a strong balance between performance and robustness. Notably, the computational cost remains nearly constant across different selection ratios.

## E. Social Impact

Our method increases the robustness of object detectors against physically realizable attacks. This could potentially lead to more effective surveillance systems, which could encroach upon personal privacy if misused. However, we believe the concrete positive impact on security generally outweighs the potential negative impacts. Robust object detectors can enhance various beneficial applications, such as autonomous driving, video surveillance for public safety, and other critical systems. Nonetheless, it is crucial to develop and deploy such technologies responsibly, with ethical considerations to mitigate potential misuse and protect individual privacy.