Supplementary Material - KDAT: Inherent Adversarial Robustness via Knowledge Distillation with Adversarial Tuning for Object Detection Models

Yarin Yerushalmi Levi[1], **Edita Grolman**[1], **Idan Yankelev**[1], **Amit Giloni**[2], **Omer Hofman**[2], **Toshiya Shimizu**[3], **Asaf Shabtai**[1], **Yuval Elovici**[1]
[1]Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel.
[2] Fujitsu Research of Europe.
[3] Fujitsu Unlimited.
{yarinye, edita, idanyan}@post.bgu.ac.il, {shabtaia, elovici}@bgu.ac.il,
{amit.giloni, omer.hofman, shimizu.toshiya}@fujitsu.com,

Abstract

In this supplementary material, we present additional information associated with KDAT. First, we provide additional background regarding the adversarial attacks used in the paper. Then, we provide a Figure of KDAT's pipeline for two-stage models. Next, we provide additional details about our evaluation and additional results. In addition, we provide details about the hyper-parameter tuning process. Finally, we created a code demo to demonstrate KDAT performance and we present its results. Our code, including the abovementioned demo are available upon request.

1 Background and Related works

In this section, we extend the background and related work from the paper by reviewing different adversarial patch attacks.

Original Adversarial Patch (Google). (Brown et al. 2017) The adversarial patch attack was originally defined as the following optimization problem:

$$\hat{P} = \arg\max_{P} \mathbb{E}_{x,t,l}\left[\log Pr(\hat{y}|A(P,x,l,t))\right] \quad (1)$$

where $A(P,x,l,t)$ denote a function applaying the patch P to original image x at location l with transformation t. Transformation t is one of a set of transformations defined in order to improve the patch's robustness to the real world, including scaling and rotating it. $Pr(\hat{y}|A)$ is the probability of classifing input A as class $\hat{y}$. The original attack was designed for image classifiers but successfully misled other computer vision models (as demonstrated in our evaluation).

Masked Projected Gradient Decent (M-PGD). (Madry et al. 2017) M-PGD is the masked variant of the standard PGD perturbation which 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:

$$P^{t+1} = clip(P^t + \alpha\nabla_{P^t}L(\hat{y}, h(A(P^t,x,l,t)),0,1) \quad (2)$$

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 the clip is a function to clip the patch values to the range of (0,1). by using this formula, the attack can be targeted (minimizing the loss between the desired model predictions and the current model predictions) and untargeted (maximizing the loss between the ground truth annotations and the current model predictions).

DPatch. (Liu et al. 2018) DPatch attack enhanced Google patch for object detection models, targeting the bounding boxes as well. The patch is iteratively trained to harm the model's ability to predict the right class and bounding box of the targeted object, optimized to increase the loss (or decrease the loss for the manipulated target) using FGSM (Goodfellow, Shlens, and Szegedy 2014) attack steps.

$$\hat{P}_{untargeted} = \arg\max_{P} \mathbb{E}_{x,s}[L(A(x,s,P);\hat{y},\hat{B})] \quad (3)$$

$$\hat{P}_{targeted} = \arg\min_{P} \mathbb{E}_{x,s}[L(A(x,s,P);y_t,B_t)] \quad (4)$$

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, y_t and B_t are ground truth labels and bounding boxes.

Printable Patches (Thys, Van Ranst, and Goedemé 2019) To further optimize the adversarial patches for real-world use cases, thys et al. utilized 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:

$$L = \alpha L_{\text{nps}} + \beta L_{\text{tv}} + L_{\text{obj}} \quad (5)$$

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.

Naturalistic Patches (Hu et al. 2021) 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.

T-SEA (Huang et al. 2023) TSEA 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 (Yamada et al. 2019) model to help improve patch transferability.

2 Methodology

In this section, we extend the methodology section of the paper by demonstrating the proposed solution architecture for two-stage models (like Faster R-CNN). Figure 1 presents the two-stage pipeline of KDAT, which utilizes the enhanced features map from the RPN as well as the objectness values.

3 Evaluation

In this section, we extend the evaluation section of the paper by adding additional details about the datasets used, the specific implementation details of our method for each of those datasets, as well as additional results.

3.1 COCO

Evaluation Settings. For the COCO (Lin et al. 2014), we created various patches that aim to mislead the model in different ways. Each adversarial example was generated by a benign image forwarded through the model to obtain a benign prediction. According to the benign prediction, the adversarial patch was placed on the object with the highest confidence value. The size of the patch was set to $\alpha \cdot \min(BB_w, BB_h)$ where BB_w and BB_h is the height and width of the target object, according to the benign prediction, and α is a value ranging from 0.3 to 0.5. We created a corresponding masked example with a black patch instead of the adversarial one for both the Feature Map (FM) Loss and for assessing the attack's success. An attack was considered successful only if it harmed the model's prediction, while the matching masked image did not. Only adversarial examples that successfully fooled the model were kept, resulting in at least seven attacked images per benign image for each attack. The validation and testing images were created using the COCO's validation set, using a similar process to create adversarial examples.
For the adaptive attacks, we used 100 randomly selected images from COCO's validation set. We used the M-PGD attack with a square patch size of 75x75. We set the initial number of iterations to 100 and, each time, assess whether the patch successfully misleads the model up to 500 iterations. If so, the current number of iterations is reported. Otherwise, we increase the number of iterations by 20 and test again. We fixed the seed and the patch initial and averaged this test 5 times for each image to negate any random influence on the results.

DETR Implementation Details. All the required models were trained for 30 epochs on RTX-4090 GPU using an AdamW optimizer with a weight decay of 1e-4. The learning rate was scheduled using a StepLR scheduler with a step size of 5 with an initial value of 1e-6 for the model's backbone and 1e-5 for the rest of the model.

Distance metrics and weights for each component were set as follows: For the FM loss component, we used PyTorch's cosine similarity (CS) loss function. For the CLS loss, we used the KL-div loss and a temperature value was set to 50. In the FA, we used a combination of CS and $\mathcal{L}_2$ norm as suggested in (Park, Kang, and Paik 2024). B and A were set at 0.75 and 0.25, respectively, to favor benign performance. The weights of the components in the overall loss function were set to 0.1, 0.6, 0.6, and 0.6 for OD Loss, FM Loss, CLS Loss, and FA, respectively. The size of the subset of adversarial images used in each epoch was set to 5. The seed was fixed to 42 in all of the experiments. The confidence threshold was set to 0.7 for DETR and 0.8 for Faster R-CNN, as done in the original models. For LGS, we set the block size to 30, overlap to 5, threshold to 0.2, and smoothing factor to 2.3.

Faster R-CNN Implementation Details. All the required models were trained for 30 epochs on RTX-4090 GPU using an SGD optimizer with a momentum of 0.9 and weight decay of 5e-4. The learning rate was scheduled using a cosine annealing learning rate with an initial value of 1e-5 for the model's backbone and 1e-4 for the rest of the model.

The distance metrics and weights in each component were set as follows: In the FM Loss component, we used PyTorch's cosine similarity (CS) loss function. For the CLS loss, we used KL-div loss, and a temperature value was set to 100. In the FA, we used a combination of CS and $\mathcal{L}_2$ norm as suggested in (Park, Kang, and Paik 2024). B and A were set to 0.75 and 0.25, respectively, to favor benign performance. The weights of the components in the overall loss function were set to 0.1, 0.6, 0.6, and 0.6 for OD Loss, FM Loss, CLS Loss, and FA, respectively. The size of the subset of adversarial images used in each epoch is set to 5. A similar hyperparameter tuning process was conducted for both Faster R-CNN (Ren et al. 2015) and DETR, as described in section 5. The seed was fixed to 42 in all of the experiments. The confidence threshold was set to 0.7 for DETR and 0.8 for Faster R-CNN, as done in the original models. For LGS, we set the block size to 30, overlap to 5, threshold to 0.2, and smoothing factor to 2.3.

Tuning with Google and M-PGD Results. Following Table 1 from the article, when training on the DPatch attack, the following subsections analyze the use of Google and M-PGD during training of the methods, affecting only training-

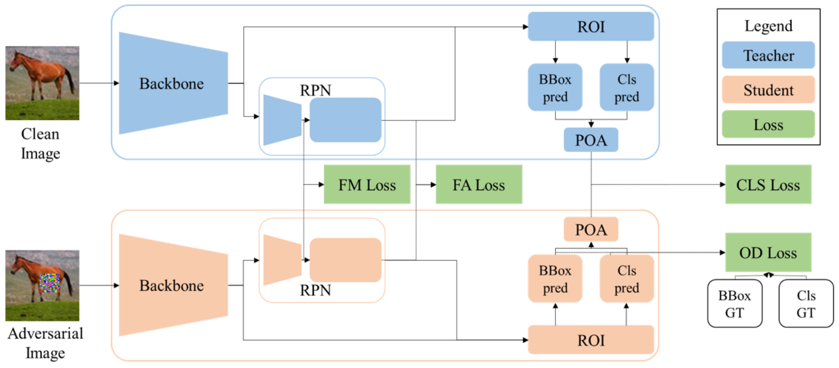

Figure 1: Proposed solution architecture for two-stage OD models.

based adversarial robustness defenses. Table 1 presents the results for training the defenses on the COCO dataset with the Google attack and evaluating on the other attacks (DPatch, or MPGD) as well as Google to demonstrate how each defense handles unseen attacks. The table consists of the performance of two evaluated OD models (rows), where each defense method's IT and mAP under three different adversarial attacks are presented. As we only used images that successfully misled the undefended model, i.e., each attack succeeded on different images, we separated each attack column into sub-columns: benign, adv (adversarial), and mean (of both benign and adv).

When examining DETR results, KDAT performed best on the DPatch attack while achieving second best on the rest of the attacks. On the other hand, in the Faster R-CNN results, KDAT reached the best performance on DPatch's and Google's attacked images and M-PGD benign images. Overall, when taking into account the results on both Faster R-CNN and DETR, although in some cases KDAT had less optimal results, it does not have any inference time overhead and reached a good trade-off between benign and adversarial performance.

on the other hand, Table 2 presents the results for training the defenses on the COCO dataset with the M-PGD attack and evaluating on the other attacks (DPatch and Google) as well as M-PGD to demonstrate how each defense handles unseen attacks. The table consists of the performance of two evaluated OD models (rows), where each defense method's IT and mAP under three different adversarial attacks are presented. As we only used images that successfully misled the undefended model, i.e., each attack succeeded on different images, we separated each attack column into sub-columns: benign, adv (adversarial), and mean (of both benign and adv).

When examining DETR results, KDAT performed best on the DPatch attack while achieving second best on the rest of the attacks. On the other hand, in the Faster R-CNN results, KDAT reached the best performance on DPatch's and Google's attacked images and M-PGD benign images. Overall, when taking into account the results on both Faster R-CNN and DETR, although in some cases KDAT had less optimal results, it does not have any inference time overhead and reached a good trade-off between benign and adversarial performance.

Table 3 presents the results for training KDAT on the COCO dataset with each of the three attacks. The table consists of the performance of two evaluated OD models (rows), where each defense method's mAP under three different adversarial attacks is presented. As we only used images that successfully misled the undefended model, i.e., each attack succeeded on different images, we separated each attack column into sub-columns: benign, adv (adversarial), and mean (of both benign and adv). As can be seen, training KDAT on COCO with the DPatch patch attack led to the overall best results when considering the effectiveness against all the three attacks.

Tuning with Multiple Attacks Results. As we aim to challenge the model during its training phase, we investigate whether exposing it to diverse attacks can improve its adversarial robustness. Our analysis shows that training on any two of the three adversarial patch attacks used yields worse results than demonstrated in Table 4.

Loss Convergence Curves. Figures 2 and 3 demonstrate the loss convergence curves of each of the KDAT loss components as well as the total loss value during the fine-tuning process for DETR and Faster R-CNN, respectively. As can be seen in the DETR's loss plot, the value stabilizes quickly after the first 6 epochs, while in the Faster R-CNN, it is a bit more noisy.

Adadptive Attacks Results. Figure 4 presents the mean NOI across the adversarial examples that successfully mislead the models. As can be seen, KDAT managed to increase the NOI required for a successful attack by 8-10% without any modification to the original model.

Combining Defense Methods Results. Our method's objective is to teach the model to adjust its prediction inherently, while methods that modify the inference process use more spatial features for the same purpose. In this part, we evaluated the combination of different methods that alter the training process with methods that alter the inference process (post hoc defenses). Our assumption was that our method adjusts the training process in a way that would

Model	Method	Inference Time		DPatch			Google			M-PGD		
		ms	fps	Benign	Adv	Mean	Benign	Adv	Mean	Benign	Adv	Mean
DETR	Undefended	**34.37**	**29.10**	0.589	0.352	0.471	0.548	0.310	0.429	0.534	0.327	0.431
	STD AT	**34.37**	**29.10**	0.579	0.452	0.516	0.522	0.415	0.469	0.483	0.385	0.434
	LGS	635.75	1.57	0.565	0.440	0.503	0.507	0.360	0.434	0.471	0.359	0.415
	Grad Defense	**34.37**	**29.10**	0.587	0.420	0.504	0.494	0.360	0.427	0.498	0.365	0.432
	AD-YOLO	**34.37**	**29.10**	0.611	0.495	0.553	**0.563**	**0.450**	**0.507**	0.528	**0.440**	**0.484**
	SAC	52.80	18.94	0.589	0.425	0.507	0.548	0.347	0.448	**0.534**	0.335	0.435
	OS	2569.23	0.39	0.558	0.407	0.483	0.523	0.348	0.436	0.505	0.362	0.434
	PAD	42609.24	0.02	0.552	0.462	0.507	0.506	0.337	0.422	0.467	0.39	0.429
	KDAT (Ours)	**34.37**	**29.10**	**0.615**	**0.509**	**0.562**	0.539	0.424	0.482	0.53	0.425	0.478
Faster RCNN	Undefended	**43.14**	**23.18**	0.483	0.229	0.356	0.519	0.164	0.342	0.496	0.223	0.360
	STD AT	**43.14**	**23.18**	0.349	0.272	0.311	0.304	0.207	0.256	0.363	0.303	0.333
	LGS	618.46	1.62	0.433	0.278	0.356	0.429	0.265	0.347	0.422	0.310	0.366
	Grad Defense	**43.14**	**23.18**	0.460	0.288	0.374	0.471	0.253	0.362	0.480	0.267	0.374
	AD-YOLO	**43.14**	**23.18**	0.408	0.288	0.348	0.407	0.326	0.367	0.418	0.302	0.360
	SAC	57.73	17.32	0.483	**0.315**	**0.399**	0.518	0.226	0.372	0.496	0.354	**0.425**
	OS	4057.20	0.25	**0.494**	0.279	0.387	**0.519**	0.309	0.414	0.494	0.279	0.387
	PAD	42796.55	0.02	0.409	0.272	0.341	0.409	0.272	0.341	0.423	**0.360**	0.391
	KDAT (Ours)	**43.14**	**23.18**	0.480	**0.315**	0.398	0.500	**0.342**	**0.421**	**0.505**	0.306	0.406

Table 1: Results on the COCO dataset using Google attack for training

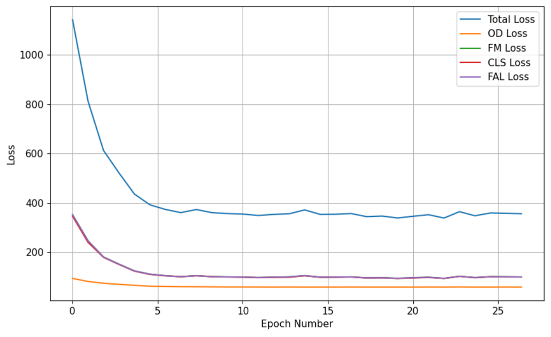

Figure 2: DETR loss convergence curves, train on COCO dataset using KDAT.

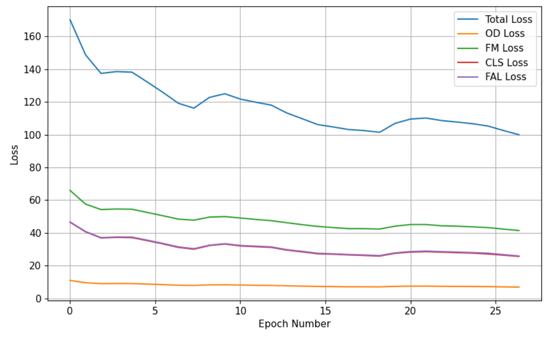

Figure 3: Faster R-CNN loss convergence curves, train on COCO dataset using KDAT.

be most compatible with existing methods. Figures 5 and 6 show the mean performance, between benign and adversarial, of combining KDAT with 4 SOTA defense methods that alter the inference process on Faster R-CNN and DETR, respectively. As can be seen, KDAT contributes the most to the adversarial robustness of the models when combined with post hoc defenses, spechily on Faster R-CNN. Notably, SAC (Liu et al. 2022) had the most synergy with our method, outperforming any existing method by itself.

Ablation Study. As described in the paper, we investigate the effect of each component of our proposed method. To demonstrate the contribution of each component, the target model was finetuned four times; each time, we used only three of the four loss components, assessing the value of the fourth one. Each experiment is referred to as a different version of our method named KDAT w/o the missing component. Table 5 presents the average mAP of the undefended model and each version of our method on benign and adversarial images on COCO and Faster R-CNN. As can be seen, all of the components of our method contribute to the overall performance gain. Although waiver of the L_{CLS} component achieved relatively close results on the COCO dataset, the complete KDAT method achieved the highest results on both adversarial and benign images.

3.2 INRIA

Evaluation Settings. For the INRIA (Dalal and Triggs 2005) dataset, we used pre-crafted adversarial patches from (Thys, Van Ranst, and Goedemé 2019; Hu et al. 2021; Huang et al. 2023). The patches were crafted to fool the Faster R-CNN model; thus, we only evaluated it on the two-

Model	Method	Inference Time		DPatch			Google			M-PGD		
		ms	fps	Benign	Adv	Mean	Benign	Adv	Mean	Benign	Adv	Mean
DETR	Undefended	**34.37**	**29.10**	0.589	0.352	0.471	0.548	0.310	0.429	0.534	0.327	0.431
	STD AT	**34.37**	**29.10**	0.593	0.483	0.538	**0.563**	**0.482**	**0.523**	0.514	0.401	0.458
	LGS	635.75	1.57	0.565	0.440	0.503	0.507	0.360	0.434	0.471	0.359	0.415
	Grad Defense	**34.37**	**29.10**	0.570	0.404	0.487	0.486	0.374	0.430	0.486	0.337	0.412
	AD-YOLO	**34.37**	**29.10**	0.593	0.514	0.554	0.532	0.417	0.475	0.497	0.401	0.449
	SAC	52.80	18.94	0.589	0.425	0.507	0.548	0.347	0.448	**0.534**	0.335	0.435
	OS	2569.23	0.39	0.558	0.407	0.483	0.523	0.348	0.436	0.505	0.362	0.434
	PAD	42609.24	0.02	0.552	0.462	0.507	0.506	0.337	0.422	0.467	0.390	0.429
	KDAT (Ours)	**34.37**	**29.10**	**0.620**	**0.519**	**0.570**	0.558	0.423	0.491	0.525	**0.445**	**0.485**
Faster RCNN	Undefended	**43.14**	**23.18**	0.483	0.229	0.356	0.519	0.164	0.342	0.496	0.223	0.360
	STD AT	**43.14**	**23.18**	0.314	0.255	0.285	0.296	0.171	0.233	0.353	0.281	0.317
	LGS	618.46	1.62	0.433	0.278	0.356	0.429	0.265	0.347	0.422	0.310	0.366
	Grad Defense	**43.14**	**23.18**	0.470	0.265	0.368	0.490	0.243	0.367	0.463	0.254	0.359
	AD-YOLO	**43.14**	**23.18**	0.397	**0.315**	0.356	0.418	0.297	0.358	0.390	0.333	0.362
	SAC	57.73	17.32	0.483	**0.315**	0.399	0.518	0.226	0.372	0.496	0.354	**0.425**
	OS	4057.20	0.25	**0.494**	0.279	0.387	**0.519**	**0.309**	**0.414**	0.494	0.279	0.387
	PAD	42796.55	0.02	0.409	0.272	0.341	0.409	0.272	0.341	0.423	**0.360**	0.391
	KDAT (Ours)	**43.14**	**23.18**	0.493	0.311	**0.402**	0.518	0.279	0.399	**0.508**	0.336	0.422

Table 2: Results on the COCO dataset using M-PGD attack for training.

Model	Method	DPatch			Google			M-PGD		
		Benign	Adv	Mean	Benign	Adv	Mean	Benign	Adv	Mean
DETR	Undefended	0.589	0.352	0.471	0.548	0.310	0.429	0.534	0.327	0.431
	KDAT(DPatch)	0.613	0.501	0.557	**0.579**	**0.435**	**0.507**	**0.559**	0.435	**0.497**
	KDAT(Google)	0.615	0.509	0.562	0.539	0.424	0.482	0.530	0.425	0.478
	KDAT(M-PGD)	**0.620**	**0.519**	**0.570**	0.558	0.423	0.491	0.525	**0.445**	0.485
FasterRCNN	Undefended	0.483	0.229	0.356	0.519	0.164	0.342	0.496	0.223	0.360
	KDAT(DPatch)	**0.506**	**0.344**	**0.425**	0.501	0.316	0.409	**0.520**	**0.343**	**0.432**
	KDAT(Google)	0.480	0.315	0.398	0.500	**0.342**	0.421	0.505	0.306	0.406
	KDAT(M-PGD)	0.493	0.311	0.402	**0.518**	0.279	0.399	0.508	0.336	0.422

Table 3: Overall results of our method on the COCO dataset when trained on different attacks.

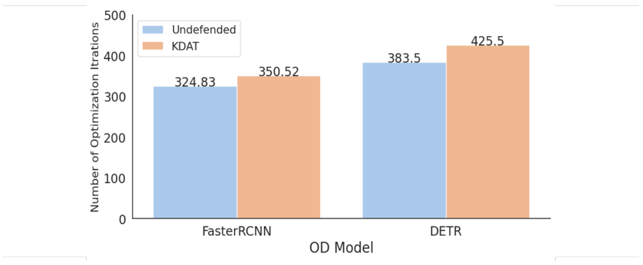

Figure 4: NOI required to generate a successful attack.

stage version of our method. The patches were placed across all the persons in the images according to the ground truth.

Implementation Details. All the required models were trained for 30 epochs on RTX-4090 GPU using an AdamW optimizer with a momentum of 0.9 and weight decay of 5e-4. The learning rate was scheduled using a cosine annealing learning rate with an initial value of 9.5e-6 for the model's backbone and 7.5e-5 for the rest of the model.

The distance metrics and weights for each component were set as follows: We used PyTorch's $\mathcal{L}_1$ loss function for the FM and FA Loss components. For the CLS loss, a combination of CS and $\mathcal{L}_2$ norm as suggested in (Park, Kang, and Paik 2024), and a temperature value was set to 97. B and A were set to 0.86 and 0.14, respectively, to favor benign performance. The weights of the components in the overall loss function were set to 0.09934, 0.7547, 0.88125, and 0.90782 for OD Loss, FM Loss, CLS Loss, and FA, respectively. The size of the subset of adversarial images used in each epoch is set to 5. The seed was fixed to 42 in all of the experiments.

Trained Attacks	Method	Inference Time		DPatch			Google			MPGD		
		ms	fps	Benign	Adv	Mean	Benign	Adv	Mean	Benign	Adv	Mean
DPatch & Google	Undefended	**43.14**	**23.18**	0.483	0.229	0.356	0.519	0.164	0.342	0.496	0.223	0.360
	STD AT	**43.14**	**23.18**	0.389	0.297	0.343	0.354	0.222	0.288	0.374	0.294	0.334
	LGS	618.46	1.62	0.433	0.278	0.356	0.429	0.265	0.347	0.422	0.310	0.366
	Grad Defense	**43.14**	**23.18**	0.486	0.303	0.394	0.481	0.301	0.391	0.498	0.279	0.389
	AD-YOLO	**43.14**	**23.18**	0.411	0.359	0.385	0.417	**0.364**	0.390	0.423	0.341	0.382
	SAC	57.73	17.32	0.483	0.315	0.399	0.518	0.226	0.372	0.496	0.354	**0.425**
	OS	4057.20	0.25	0.494	0.279	0.387	**0.519**	0.309	**0.414**	0.494	0.279	0.387
	PAD	42796.55	0.02	0.420	0.359	0.389	0.409	0.272	0.341	0.423	**0.360**	0.391
	KDAT (Ours)	**43.14**	**23.18**	**0.502**	**0.342**	**0.422**	0.505	0.318	0.412	**0.507**	0.312	0.410
DPatch & M-PGD	Undefended	**43.14**	**23.18**	0.483	0.229	0.356	0.519	0.164	0.342	0.496	0.223	0.360
	STD AT	**43.14**	**23.18**	0.368	0.296	0.332	0.333	0.221	0.277	0.398	0.306	0.352
	LGS	618.46	1.62	0.433	0.278	0.356	0.429	0.265	0.347	0.422	0.310	0.366
	Grad Defense	**43.14**	**23.18**	0.470	0.295	0.382	0.473	0.257	0.365	0.473	0.277	0.375
	AD-YOLO	**43.14**	**23.18**	0.397	0.350	0.374	0.410	**0.330**	0.370	0.425	0.355	0.390
	SAC	57.73	17.32	0.483	0.315	0.399	0.518	0.226	0.372	0.496	0.354	**0.425**
	OS	4057.20	0.25	0.494	0.279	0.387	**0.519**	0.309	**0.414**	0.494	0.279	0.387
	PAD	42796.55	0.02	0.420	**0.359**	0.389	0.409	0.272	0.341	0.423	**0.360**	0.391
	KDAT (Ours)	**43.14**	**23.18**	**0.495**	0.336	**0.416**	0.508	0.319	**0.414**	**0.511**	0.335	0.423
Google & M-PGD	Undefended	**43.14**	**23.18**	0.483	0.229	0.356	0.519	0.164	0.342	0.496	0.223	0.360
	STD AT	**43.14**	**23.18**	0.361	0.290	0.326	0.364	0.199	0.282	0.381	0.295	0.338
	LGS	618.46	1.62	0.433	0.278	0.356	0.429	0.265	0.347	0.422	0.310	0.366
	Grad Defense	**43.14**	**23.18**	0.464	0.300	0.382	0.477	0.235	0.356	0.469	0.267	0.368
	AD-YOLO	**43.14**	**23.18**	0.411	0.323	0.367	0.436	**0.342**	0.389	0.420	0.352	0.386
	SAC	57.73	17.32	0.483	0.315	0.399	0.518	0.226	0.372	0.496	0.354	**0.425**
	OS	4057.20	0.25	**0.494**	0.279	0.387	**0.519**	0.309	0.414	0.494	0.279	0.387
	PAD	42796.55	0.02	0.420	**0.359**	0.389	0.409	0.272	0.341	0.423	**0.360**	0.391
	KDAT (Ours)	**43.14**	**23.18**	0.490	0.318	**0.404**	0.515	0.340	**0.428**	**0.498**	0.343	0.421

Table 4: Results on the COCO dataset when trained on multiple attacks (Faster R-CNN).

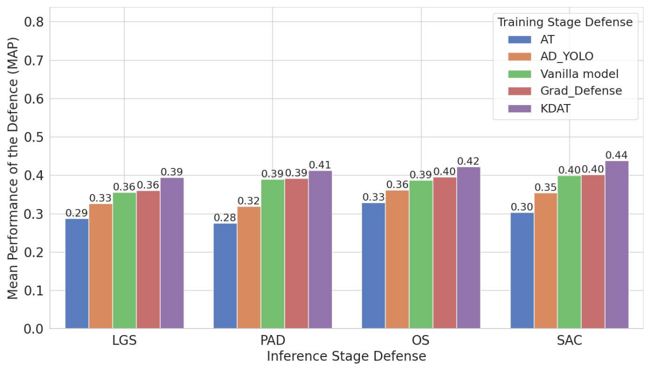

Figure 5: Mean performance, between benign and adversarial, of the combination of methods that alter the training process with methods that alter the inference process on Faster R-CNN.

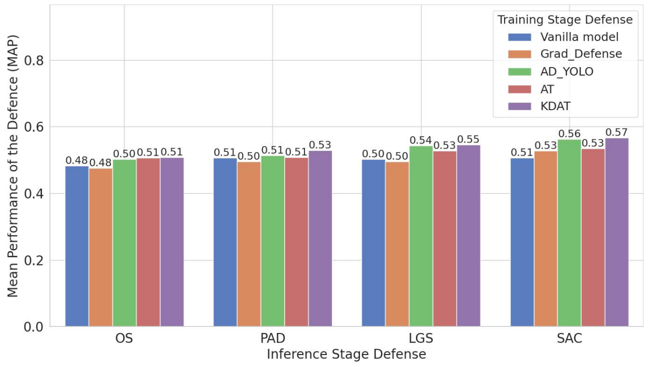

Figure 6: Mean performance, between benign and adversarial, of the combination of methods that alter the training process with methods that alter the inference process on DETR.

The confidence threshold was set to 0.7 for DETR and 0.8 for Faster R-CNN, as done in the original models. For LGS, we set the block size to 30, overlap to 5, threshold to 0.2, and smoothing factor to 2.3.

Method	Benign	Adv	Mean
Undefended	0.499	0.205	0.352
KDAT w/o L_{OD}	0.495	0.279	0.387
KDAT w/o L_{FM}	0.504	0.328	0.416
KDAT w/o L_{CLS}	0.504	0.331	0.418
KDAT w/o L_{FA}	0.490	0.331	0.411
KDAT (Ours)	**0.509**	**0.334**	**0.422**

Table 5: The ablation study results on COCO (Faster R-CNN).

Method	Inference Time		Benign	Adv	Mean
	ms	FPS			
Undefended	**43.14**	**23.18**	0.951	0.165	0.558
AT	**43.14**	**23.18**	0.840	0.172	0.506
LGS	618.46	1.62	0.942	0.497	0.719
Grad Defense	**43.14**	**23.18**	0.713	0.177	0.445
AD-YOLO	**43.14**	**23.18**	0.750	0.115	0.433
SAC	57.73	17.32	0.951	0.350	0.651
OS	4057.20	0.25	0.954	0.185	0.570
PAD	42796.55	0.02	0.950	**0.797**	**0.873**
KDAT (Ours)	**43.14**	**23.18**	**0.961**	0.410	0.686

Table 6: Results on the INRIA dataset with T-SEA attack (Faster R-CNN).

Generlization to New attacks - T-SEA patches. Table 6 presents the average mAP on the benign and adversarial images and summarizes the performance of various defense methods applied to the Faster R-CNN model under a T-SEA patch attack. We utilized the models described in the main paper for the training-based defenses, including KDAT (Section 4.2 - Multiple Patches). These models were trained on a subset of images attacked with natural patches (P1-P3) to assess the defenses' ability to generalize to new adversarial attacks targeting the defended model.

Notably, KDAT improved the adversarial robustness of the object detection model without requiring additional adjustments, increasing the MAP on adversarial images from 0.165 to 0.410. While PAD and LGS demonstrated better detection performance than KDAT, both defenses negatively impacted the inference time which is critical for real-time object detection tasks.

Ablation Study. As described in the paper, we investigate the effect of each component of our proposed method. To demonstrate the contribution of each component, the target model was finetuned four times; each time, we used only three of the four loss components, assessing the value of the fourth one. Each experiment is referred to as a different version of our method named KDAT w/o the missing component. Table 7 presents the average mAP of the undefended model and each version of our method on benign and adversarial images on INRIA and Faster R-CNN. The complete KDAT method achieved the highest results on both adversarial and benign images.

Method	Benign	Adv	Mean
Undefended	0.951	0.548	0.750
KDAT w/o L_{OD}	0.900	0.651	0.775
KDAT w/o L_{FM}	0.805	0.597	0.701
KDAT w/o L_{CLS}	0.687	0.572	0.630
KDAT w/o L_{FA}	0.775	0.640	0.708
KDAT (Ours)	**0.961**	**0.858**	**0.910**

Table 7: The ablation study results on INRIA (Faster R-CNN).

3.3 Superstore

Evaluation Settings. The superstore (Hofman et al. 2024) contains physical adversarial examples, products with adversarial patches placed on top of them, aiming to mislead smart shopping carts into classifying expensive products as cheap ones. The adversarial patches were created and printed for both the training and test sets with sizes ranging from 200 × 200 to 400 × 400 pixels.

Implementation Details. All the required models were trained for 30 epochs on RTX-4090 GPU using an SGD optimizer with a momentum of 0.9 and weight decay of 5e-4. The learning rate was scheduled using a cosine annealing learning rate with an initial value of 1.5e-6 for the model's backbone and 6e-5 for the rest of the model.

Each component's distance metrics and weights were set as follows: For the FM and FA Loss component, a combination of CS and $\mathcal{L}_2$ norm as suggested in (Park, Kang, and Paik 2024). For the CLS loss, we used KL-div loss, and a temperature value was set to 14. B and A were set to 0.81 and 0.19, respectively, to favor benign performance. The weights of the components in the overall loss function were set to 0.07965, 0.17799, 0.72672, and 0.58513 for OD Loss, FM Loss, CLS Loss, and FA, respectively. The size of the subset of adversarial images used in each epoch is set to 5. The seed was fixed to 42 in all of the experiments. The confidence threshold was set to 0.7 for DETR and 0.8 for Faster R-CNN, as done in the original models. For LGS, we set the block size to 30, overlap to 5, threshold to 0.2, and smoothing factor to 2.3.

4 The Hyperparameter Tuning Process

As described in the paper, our method includes several sets of hyperparameters that need to be determined. Due to the vast search space and our limited computational resources, we narrowed down our choices in each category to a few options that should meet the requirements of the research process, though they may not be the most optimal. For the INRIA and Superstore datasets, we used the Optuna framework (Akiba et al. 2019).

Training Parameters. We evaluated SGD and AdamW as the training optimizers with learning rates ranging from 1e-5 to 1-6 for the backbone and 1e-4 to 1-6 for the rest of the model.

Dataset Parameters. The number of clean images was initially set to 200, and we tested the variant with 500 instead, but it performed worst on the validation set. We used 5 adversarial examples per clean image as we tried different values of 1 and 3 and reached lower results on the validation set. We didn't try values higher than 5 because we aimed to use a different subset of the original corresponding adversarial set of images (at least 7 images) for each epoch.

The Distance Metrics. We tested the use of $\mathcal{L}_1$ loss, $\mathcal{L}_2$ loss, and CS, as well as a combination of MSE and CS as suggested by (Park, Kang, and Paik 2024). For the classification component, we evaluated the use of distribution-oriented loss calculations such as KL-div and Wasserstein distance, as well as the previous options.

The Weights of Each Component. We normalized the loss values; thus, the weight given to each component can be at the same scale. We evaluated the options of 0.3-1 for each component (0-0.3 for the OD loss), selecting the optimal configuration. The range of values evaluated for B and A, which set the tradeoff between benign and adversarial components, are 0.75-0.85 and 0.25-0.15, respectively.

5 Discussion

In this section, we expand on the discussion from our paper, addressing points that were constrained by space limitations in the main text and incorporating insights derived from the results presented in this supplementary material.

Comparing Defense Mechanisms. The two primary approaches to enhancing the adversarial robustness of object detection models are: (1) modifying the model's training process, and (2) employing post-hoc methods to adjust the inference process. While post-hoc methods can be more easily integrated into existing and new models, they come with significant trade-offs, including increased computational resource requirements and reduced inference speed, which can be critical in time-sensitive scenarios.

In contrast, modifying the training process requires retraining or fine-tuning the model with additional datasets, which demands extra effort upfront. KDAT is a method that modifies the training process through a fine-tuning step, manages to outperform defenses from both approaches in most cases, and enhances benign performance, making it a practical choice for real-world applications where computational resources and inference time are critical constraints. Its one-time setup effort ensures long-term effectiveness and usability.

Tuning with Multiple Attacks. As presented in Table 4, when using multiple attacks (two attacks), the results obtained were worse compared to using a single attack (Section 3.1). We hypothesize that when a student is exposed to varying types of information (sourced from different attacks) that can be contrasted, it introduces unwanted noise into the training process. This likely leads to poorer results compared to a student who receives consistent and directionally guided information.

Choosing Attack for Tuning. Since in most of the examined compared methods (including KDAT), training with a single attack obtained higher results compared to training with multiple attacks, we address the process of choosing the attack to train with. We hypothesize that selecting the attack to use in the tuning process should be approached in the same way as choosing hyperparameters and researched accordingly.

6 Code and Demo

The code for this paper, including the relevant models for using KDAT, will be provided upon request. It includes a demo Python file for Faster R-CNN and DETR, along with the pre-trained weights for these models.

6.1 DETR

Figure 8 demonstrates the predictions of DETR on both benign and attacked images, using KDAT to compare to the undefended model (Vanilla) on the COCO validation set.

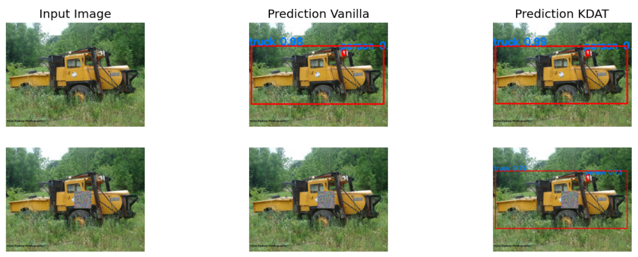

Figure 7: KDAT demonstration on COCO and DETR.

6.2 Faster R-CNN

COCO. Figure 8 demonstrates the predictions of Faster R-CNN on both benign and attacked images, using KDAT compared to the undefended model (Vanilla) on the COCO validation set.

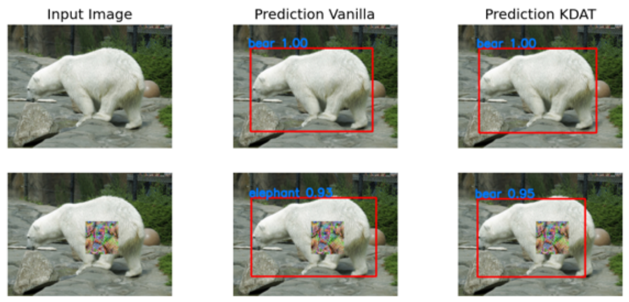

Figure 8: KDAT demonstration on COCO and Faster R-CNN.

INRIA. Figure 9 demonstrates the predictions of Faster R-CNN on both benign and attacked images, using KDAT compared to the undefended model (Vanilla) on the INRIA test set.

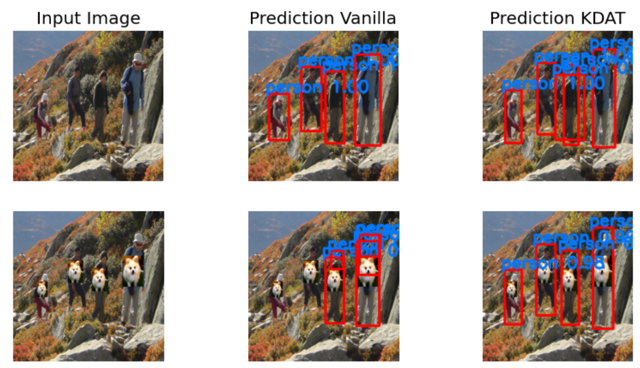

Figure 9: KDAT demonstration on INRIA and Faster R-CNN.

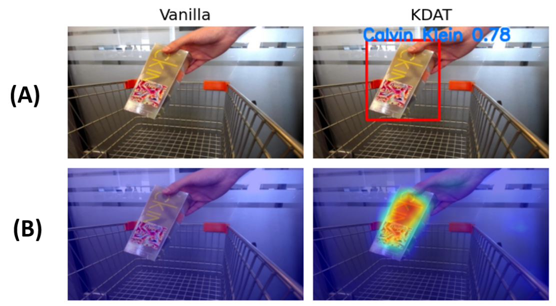

Figure 10: KDAT demonstration on Superstore and Faster R-CNN.

Superstore. Figure 10 demonstrates the predictions (A) and the corresponding saliency-map (B) of Faster R-CNN on the attacked image using KDAT compared to the undefended model (Vanilla) on Superstore test set.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2623–2631.

Brown, T. B.; Mané, D.; Roy, A.; Abadi, M.; and Gilmer, J. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665*.

Dalal, N.; and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, 886–893. Ieee.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hofman, O.; Giloni, A.; Hayun, Y.; Morikawa, I.; Shimizu, T.; Elovici, Y.; and Shabtai, A. 2024. X-detect: Explainable adversarial patch detection for object detectors in retail. *Machine Learning*, 1–20.

Hu, Y.-C.-T.; Kung, B.-H.; Tan, D. S.; Chen, J.-C.; Hua, K.-L.; and Cheng, W.-H. 2021. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7848–7857.

Huang, H.; Chen, Z.; Chen, H.; Wang, Y.; and Zhang, K. 2023. T-sea: Transfer-based self-ensemble attack on object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 20514–20523.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, J.; Levine, A.; Lau, C. P.; Chellappa, R.; and Feizi, S. 2022. Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14973–14982.

Liu, X.; Yang, H.; Liu, Z.; Song, L.; Li, H.; and Chen, Y. 2018. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Park, S.; Kang, D.; and Paik, J. 2024. CSKD: Cosine Similarity-Guided Knowledge Distillation for Robust Object Detectors.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Thys, S.; Van Ranst, W.; and Goedemé, T. 2019. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 0–0.

Yamada, Y.; Iwamura, M.; Akiba, T.; and Kise, K. 2019. Shakedrop regularization for deep residual learning. *IEEE Access*, 7: 186126–186136.