DIFFUSION TO CONFUSION: NATURALISTIC ADVERSARIAL PATCH GENERATION BASED ON DIFFUSION MODEL FOR OBJECT DETECTOR

Appendix

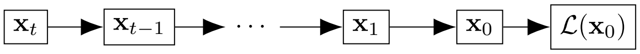


Fig. 1. The computational graph to compute the loss of an ODE.

1. ADJOINT METHOD FOR BACKPROPAGATION

We can implement the DM in ODE form. That is, the target Ordinary Differential Equation (ODE) is given by $d\mathbf{x} = \mathbf{f}_\theta(\mathbf{x}_t, t)dt$, where $\mathbf{f}_\theta$ is the ODE driven by the DDIM formulation. Consider the computational graph of the DDIM diffusion-generating process without randomness. Then, the next step is uniquely determined by the current state. If we produce some final loss L using the last state $\mathbf{x}_0$, for clarity, we explicitly draw the dependency in Figure 1. Chen *et al.* [1] demonstrate how to backpropagate through the generating process through the Neural ODE by solving another adjoint ODE, with only $\mathcal{O}(1)$ space memory and linear time complexity. In this case, the adjoint state is defined as $\mathbf{a}_t = \frac{d\mathcal{L}}{d\mathbf{x}_t}$. It satisfied the adjoint ODE,

$$\frac{d\mathbf{a}_t}{dt} = -\mathbf{a}_t \frac{\partial \mathbf{f}_\theta(\mathbf{x}_t, t)}{\partial \mathbf{x}_t}. \tag{1}$$

Thus far, DM has exhibited superior properties compared to various generative models, without imposing excessive computation burdens. This makes it a perfect candidate as an adversarial patch generator.

To adopt the adjoint method to optimize per iteration of the latent, we use the Euler method to backpropagate the adjoint method.

In addition, our approach involves updating the latent representation of a patch at time t through a forward operation from latent l_t to the patch and then to the loss as follows:

$$\text{patch} = \text{vae\_decode}\left(l_t + \frac{1}{2}\int_t^0 \text{score}(l_u, u)\, du\right) \tag{2}$$

$$\text{loss} = \text{loss\_fn}(\text{patch}, \text{batch}) \tag{3}$$

The computational load of backpropagating Equation 2 is considerably higher than that of Equation 3. We employ a partial backpropagation strategy for Equation 3 with each batch, followed by a complete backpropagation of Equation 2 after processing a large dataset. This approach significantly improves the efficiency of updating Δl_t, namely gradient accumulation (AMB-GA) for the AMB.

Listing 1. Classic Method

```python
for epoch in range(EPOCH):
    for batch in inria_dataloader:
        x_0 = diffusion(x_half)
        loss = adv_patch(x_0, batch)
        loss.backward()
        optimizer.step()
```

Listing 2. Revised Method

```python
for epoch in range(EPOCH):
    x_0 = diffusion(x_half) # time bottleneck
    derivative = sum(
        torch.autograd.grad(
            adv_patch(x_0, batch), x_0,
            retain_graph=False
        )[0]
        for batch in inria_dataloader
    ) # This does not take much time
    x_0.backward(derivative)
    optimizer.step()
```

Fig. 2. The PyTorch code to accelerate adjoint-method-based backpropagation for the diffusion model. We wrap the detail of diffusion steps in Figure 1 as `x_0 = diffusion(x_half)`. We abstract the scene rendering and other operations to generate a loss function as `loss = adv_patch(x_0, batch)`. Our revision is to speed up the data processing time instead of updating the patch per sample in the batch.

Therefore, despite the $\mathcal{O}(1)$ memory in need, we found backpropagating through the generating process is time-consuming. Although the adjoint method cannot speed up the backpropagation, we can still increase the data processed per backpropagation. In our implementation, we use the full dataset to implement an iteration on the starting latent as shown in Figure 2. The result is 30 times more instances processed per second as shown in Table 1.

2. MORE EXPERIMENTS

2.1. Cross-model Generalizability

We further show the results using the proposed approach with AMB in Table 3 show that our generated patches still have a higher attack performance than the previous state-of-the-art GAN-based naturalistic adversarial patch generation method by Hu *et al.* [5] where the diagonal entries of the tables cor-

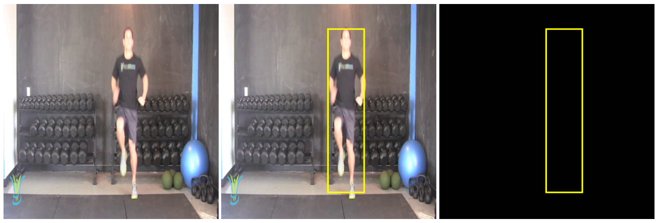
Table 1. Time Comparisons of the classic and the revised methods for adjoint method-based backpropagation for the diffusion model (Figure 2).

Method	Batch size	Throughput (sample/second)
Classic	12	1.01
Revised	**614**	**30.94**

Table 2. **Samples images of the INRIA [2] and MPII datasets [3]**.

INRIA [2]

MPII [3]



respond to the performances in the white-box attack setting, and the off-diagonal ones refer to the black-box attack setting in which an adversarial patch generated upon attacking one detector transfers well to another.

These results suggest that our patches are naturalistic and stealthy enough to fail SAC. This is possibly due to the fact that their model was trained on adversarial noise patches which have very different distributions from ours, which may limit its ability to generalize on other types of adversarial patches.

2.2. Cross-dataset Evaluation

To enable cross-dataset evaluation, we use the same testing split from Hu *et al.* [5] for the MPII dataset [3], in which we use the object detectors' predictions on clean test data (i.e., without applying any patch) as the reference labels. Table 2 shows examples of the test images in both datasets and their corresponding reference labels.

The results in Table 4 indicate that the performance of the generated patches on MPII may not necessarily generalize well to other datasets, possibly because there are differences in attributes such as the position, number, viewing angle of people, *etc*.

2.3. More Robustness Results against Existing Defenses

Besides using the off-the-shelf pretrained SAC model, we further finetune the SAC model with our diffusion-based adversarial patches for more challenging evaluation upon the proposed method. The patches are optimized with 20 epochs using 5-step AMB. We randomly pasted one of the 1,000 generated patches onto each image from a COCO sub-dataset of 10,000 samples, obtaining the results in Table 5. Although the attack performances slightly drop, the proposed methods are still robust against the finetuned SAC.

Besides the quantitative results, we also present the qualitative results of the robustness against the state-of-the-art adversarial patch removal algorithm, segment-and-complete (SAC) [7], for the proposed approach and other adversarial patch generation methods. As illustrated in Table 6 and 7, all the approaches can easily evade the detection of vanilla YOLOv2. However, when SAC is applied, the patches generated by [4] and NPAP [5] (i.e., NPAP is a GAN-based naturalistic adversarial patch generation method.) are detected and removed so that the pedestrians are still detected by YOLOv2. In contrast, the proposed approach can effectively defend against the attack of SAC.

2.4. Classifier-free Guidance Scale

Ho and Salimans [8] proposed classifier-free guidance (CFG) to control the output content of the diffusion model. They randomly mask out the conditional input during training the denoising U-Net, making the U-Net capable of both unconditional and conditional denoising. During sampling, a CFG weight w controls the amount of extrapolation from the unconditional noise prediction towards the conditional one. We sweep over a range of w (i.e., from the values of $\{1, 2, ..., 20\}$) to see if higher text condition guidance helps boost the naturalness of the generated patches, as shown in Figure 3.

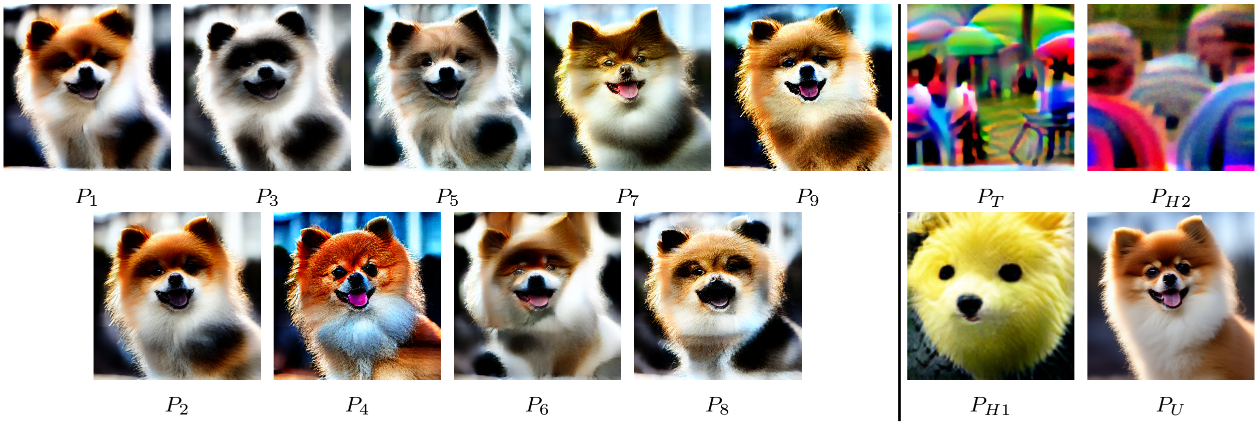
2.5. Memory Reduction Analysis

Our memory analysis on the GPU demonstrates the efficiency of the proposed AMB algorithm on FP16 Stable Diffusion (SD). For the forward process of AMB Equation 2, the memory requirement is 2.9 GB, and for Equation 3, it is 5.0 GB for batch size 16. These numbers are significantly lower than the memory demands of traditional techniques. Without AMB, the memory usage is estimated $(4.1 \times N + C)$ GB, where N is the number of denoising steps and C depends on the implementation of Equation 3 and VAE. In our setting, $C = 6.2$. As backpropagation releases the computational graph, the memory usage will decrease to approximately the model size. As the denoising steps grows, it quickly becomes infeasible for a consumer-grade 24GB NVidia RTX 3090 GPU.

Table 3. Cross-model attack performance of the patches generated using **AMB** (mAP%, lower is better). Best results are underlined, and the top-2 best results are in **bold**.

	Method	YL2	YL3	YL3t	YL4	YL4t	FRCNN	DDETR	YL5s	YL7t	**Avg.**
	(P_T) Thys *et al.* [4] *	**2.68**	**22.51**	**8.74**	**12.89**	**3.25**	39.41	34.46	34.00	**20.81**	**19.86**
	(P_{H1}) Hu *et al.* [5] †	34.76	37.79	21.69	46.80	8.67	59.97	55.85	38.19	29.20	36.99
	(P_{H2}) Huang *et al.* [6] *	**7.52**	**10.71**	13.83	31.91	17.25	**24.86**	**14.12**	**29.66**	29.39	**19.92**
Ours (trained on)	(P_1) YL2	8.56	39.52	25.57	65.92	26.49	47.36	59.56	62.88	42.53	42.04
	(P_2) YL3	24.9	27.59	22.11	58.89	22.7	51.54	57.26	59.21	35.46	39.96
	(P_3) YL3t	33.29	35.85	**8.54**	64.78	17.68	59.28	58.9	66.87	29.17	41.6
	(P_4) YL4	50.39	61.92	50.42	**18.79**	53.87	61.46	59.84	66.51	49.2	52.49
	(P_5) YL4t	28.06	34.68	18.48	62.86	**9.33**	58.51	56.03	59.07	32.65	39.96
	(P_6) FRCNN	17.71	39.61	32.65	66.77	35.41	**23.91**	56.52	55.63	34.92	40.35
	(P_7) DDETR	33.12	59.12	44.16	60.31	48.42	41.21	**21.19**	44.78	44.25	44.06
	(P_8) YL5s	26.04	31.56	21.45	57.61	23.21	55.22	52.2	**38.66**	25.41	36.82
	(P_9) YL7t	28.72	50.12	36.85	47.11	35.56	58.85	54.82	56.49	**19.48**	43.11
	(P_U) Unoptimized Patch	62.74	71.15	65.60	63.55	63.84	67.01	68.49	74.38	59.10	66.21

*Trained on YL2. †Trained on YL4t. All results are the best available.



2.6. Computation Efficiency

It is worth noting that the cross-model results reported in the main paper are obtained by applying gradient checkpointing. Without applying checkpointing or adjoint method, it is unable to generate the patch upon a single NVidia RTX 3090 GPU. In Table 8, we show the required time to generate a patch in 5 denoising steps and 100 epochs over the full INRIA dataset using a single 3090 GPU for gradient checkpointing, AMB, and AMB-GA, respectively. For 30x faster speed, we refer to the acceleration achieved by AMB-GA over AMB.

3. ABLATION STUDY

In this section, we thoroughly investigate the effectiveness of different components for the proposed approach.

3.1. Diffusion Parameters

Noise level. Through our experiments, we find that different noise levels significantly affect the training results. We observe that setting $t_{\texttt{start}}$ to a small value (typically less than $0.3T$) does not introduce sufficient randomness to the process, resulting in images similar to those in prior study [5] which generates patches by adjusting the latent directly. Conversely, setting $t_{\texttt{start}}$ to a large value (usually greater than $0.7T$) leads to excessive randomness, causing gradient vanishing and prolonging training time. These effects are particularly pronounced at extreme values of t (0 and T), where setting $t_{\texttt{start}} = 0$ eliminates noise altogether, and setting $t_{\texttt{start}} = T$ results in pure noise and gradient elimination. Therefore, in our experiments, we set $t \in [0.4T, 0.6T]$ as the "sweet spot", since it strikes a balance between introducing enough randomness and avoiding excessive noise that results in optimization difficulties. **Step size.** We provide a holistic view for the effect of different $t_{\texttt{start}}$ and step size s in Figure 4, where we use CLIP [9] embeddings to calculate the simi-

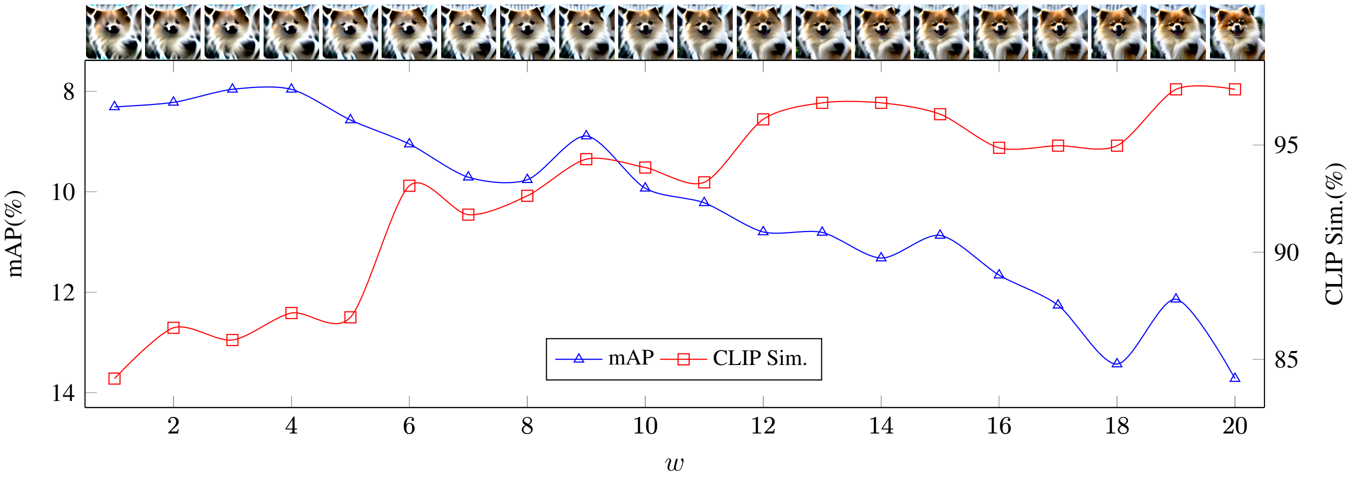


Fig. 3. **Effects of the classifier-free diffusion guidance scale.**

Table 4. Cross-dataset attack performance (mAP%) using YL4t. Best results are in **bold**.

Method	Trained on	INRIA			MPII			Mix		
	Tested on	INRIA	MPII	Mix	INRIA	MPII	Mix	INRIA	MPII	Mix
Hu *et al.* [5]		8.67	**0.51**	**2.69**	22.05	7.92	14.12	18.45	6.32	11.68
Ours		**8.45**	2.38	4.99	**15.59**	**6.10**	**10.24**	**12.38**	**4.04**	**7.64**

Table 5. The Attack Performance for YL2 in MAP(%) after applying finetuned SAC [7].

Method	without SAC	Original SAC	Finetuned SAC
Ours (AMB)	8.56	10.14	29.91
Ours	11.62	11.62	29.19

larity between the initial patch P_{init} and the optimized patch. CLIP is a foundation model frequently used as the go-to feature extractor. We use the similarity as an automatic objective metric that reflects the preservation of image quality and semantic. We visually show that s has little impact on the generated patch and therefore average the statistics along its axis in the line chart of Figure 4. Small t_{start} leads to suboptimal image quality due to its lack of randomness, and big t_{start} causes the image to disregard our optimization objective and its semantic, despite generating good-looking pictures.

4. PHYSICAL EXPERIMENTS

To show that our method generalizes well to real-world settings, we use commercial heat transfer technology to print the adversarial patches onto T-shirts. In addition to clothes, we also realize the adversarial patch as canvas and paper. Figure 5 shows pictures of the printed materials. We also illustrate the evaluation results of our physical experiments under the real-world scenes for the captured videos. The sampled frames are shown in Figure 6. We find the proposed method can consistently and successfully evade the detection by an object detector.

5. REFERENCES

[1] Chen et al., “Neural ordinary differential equations,” *NeurIPS*, 2018.

[2] Dalal et al., “Histograms of oriented gradients for human detection,” in *CVPR*, 2005.

[3] Andriluka et al., “2d human pose estimation: New benchmark and state of the art analysis,” in *CVPR*, 2014.

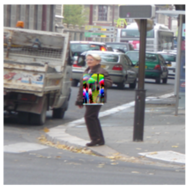
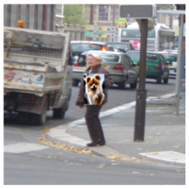
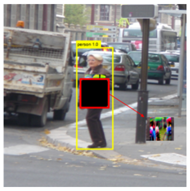
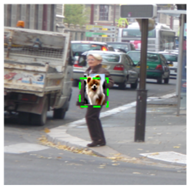
[4] Thys et al., “Fooling automated surveillance cameras: adversarial patches to attack person detection,” in *CVPR Workshop*, 2019.

[5] Hu et al., “Naturalistic physical adversarial patch for object detectors,” in *ICCV*, 2021.

[6] Huang et al., “T-sea: Transfer-based self-ensemble attack on object detection,” in *CVPR*, 2023.

[7] Liu et al., “Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection,” in *CVPR*, 2022.

Table 6. Qualitative comparison of the robustness against SAC [7] for [4] and our approach where YL2 denotes YOLOv2.

	[4]	Ours
YL2		
YL2+SAC		

[8] Ho et al., "Classifier-free diffusion guidance," in *NeurIPS Workshop*, 2021.

[9] Radford et al., "Learning transferable visual models from natural language supervision," in *ICML*, 2021.

[10] Redmon et al., "Yolo9000: better, faster, stronger," in *CVPR*, 2017.

[11] Redmon et al., "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[12] Bochkovskiy et al., "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[13] Jocher et al., "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," 2022.

[14] Wang et al., "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[15] Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.

[16] Zhu et al., "Deformable DETR: deformable transformers for end-to-end object detection," in *ICLR*, 2021.

[17] Rombach et al., "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

Table 7. Qualitative comparison of the robustness against SAC [7] for NPAP [5] and our approach where YL2 denotes YOLOv2.

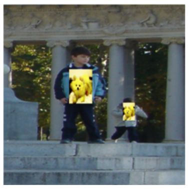
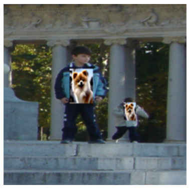
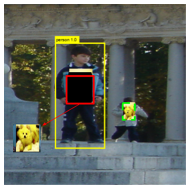
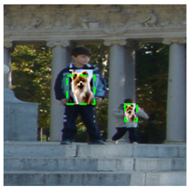
	NPAP [5]	Ours
YL2		
YL2+SAC		

Table 8. The computation time required for different generation strategy where vanilla denotes gradient checkpointing.

Setting	Vanilla	AMB	AMB-GA
Time	8hr32min	2hr48min	14min

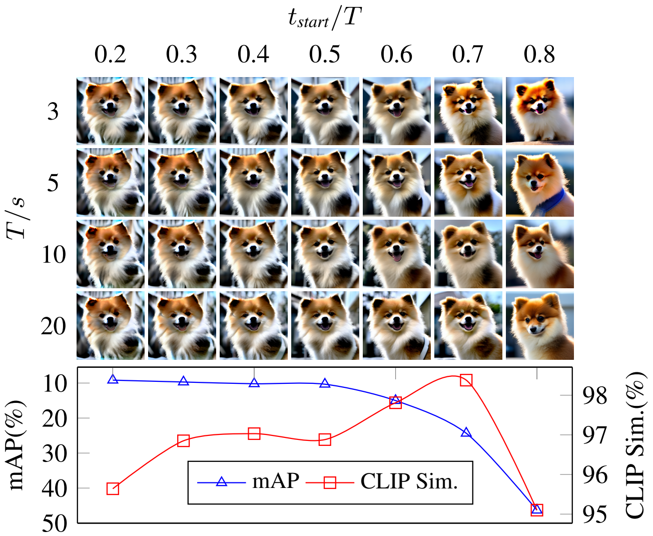


Fig. 4. Effects of the noise level and step size. In the bottom line chart, each data point is an average score of the four patches on the same vertical axis.

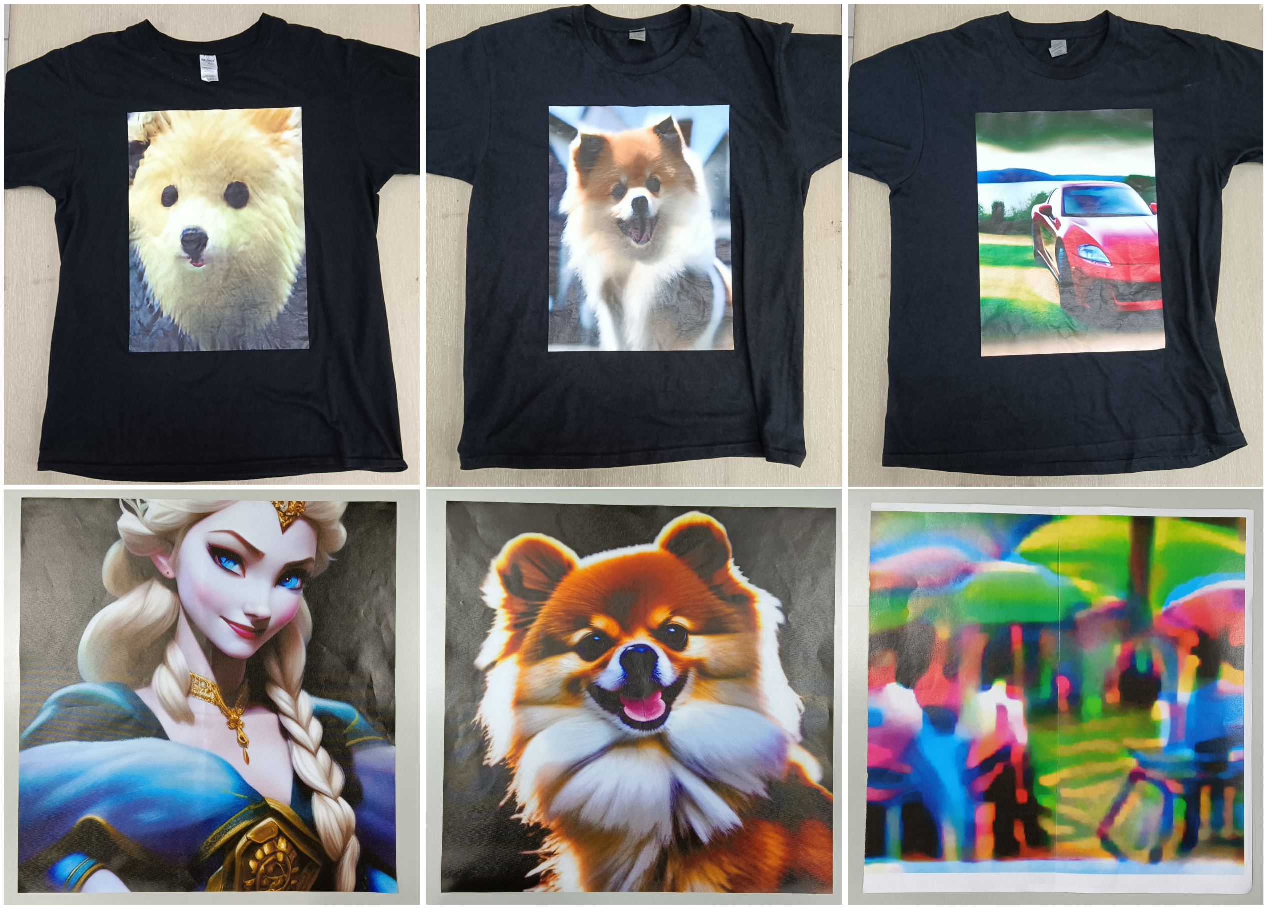


Fig. 5. **Printed materials, from top left to right bottom are cloth, cloth, cloth, canvas, canvas, and paper, respectively.**

Table 9. **Model repositories and weights.**

Model family	Code repository	Pretrained weights
YOLOv2 [10]	https://gitlab.com/EAVISE/adversarial-yolo	https://pjreddie.com/media/files/yolov2.weights
YOLOv3 [11]	https://github.com/eriklindernoren/PyTorch-YOLOv3	https://pjreddie.com/media/files/yolov3.weights https://pjreddie.com/media/files/yolov3-tiny.weights
YOLOv4 [12]	https://github.com/Tianxiaomo/pytorch-YOLOv4	https://www.dropbox.com/s/jp30sq9k21op55j/yolov4.weights https://www.dropbox.com/s/t90a1xazhbh2ere/yolov4-tiny.weights
YOLOv5 [13]	https://github.com/ultralytics/yolov5	https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt
YOLOv7 [14]	https://github.com/WongKinYiu/yolov7	https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7-tiny.pt
FasterRCNN [15]	https://pytorch.org/vision/0.12/_modules/torchvision/models/detection/faster_rcnn.html	https://download.pytorch.org/models/fasterrcnn_resnet50_fpn_coco-258fb6c6.pth
Deformable DETR [16]	https://huggingface.co/SenseTime/deformable-detr	https://huggingface.co/SenseTime/deformable-detr
Latent Diffusion [17]	https://github.com/CompVis/latent-diffusion	https://ommer-lab.com/files/latent-diffusion/nitro/txt2img-f8-large/model.ckpt
Stable Diffusion [17]	https://github.com/CompVis/stable-diffusion	https://huggingface.co/CompVis/stable-diffusion-v1-4 https://huggingface.co/hakurei/waifu-diffusion-v1-3 https://huggingface.co/stabilityai/stable-diffusion-2-base

Table 10. Initialized images and their corresponding text prompts.

Generated image	Prompt	Negative Prompt
	`A high quality photo of a Pomeranian, clean background.`	`Bad anatomy, bad proportions, deformed, disfigured, duplicate, mutation, ugly, weird eyes.`
	`A high quality photo of a car.`	`Bad anatomy, bad proportions, deformed, disfigured, duplicate, mutation, ugly.`
	`1girl, aqua eyes, baseball cap, blonde hair, closed mouth, earrings, green background, hat, hoop earrings,jewelry, looking at viewer, shirt, short hair, simple background, solo, upper body, yellow shirt, without hand.`	`Bad anatomy, bad proportions, deformed, disfigured, duplicate, mutation, ugly, weird eyes.`

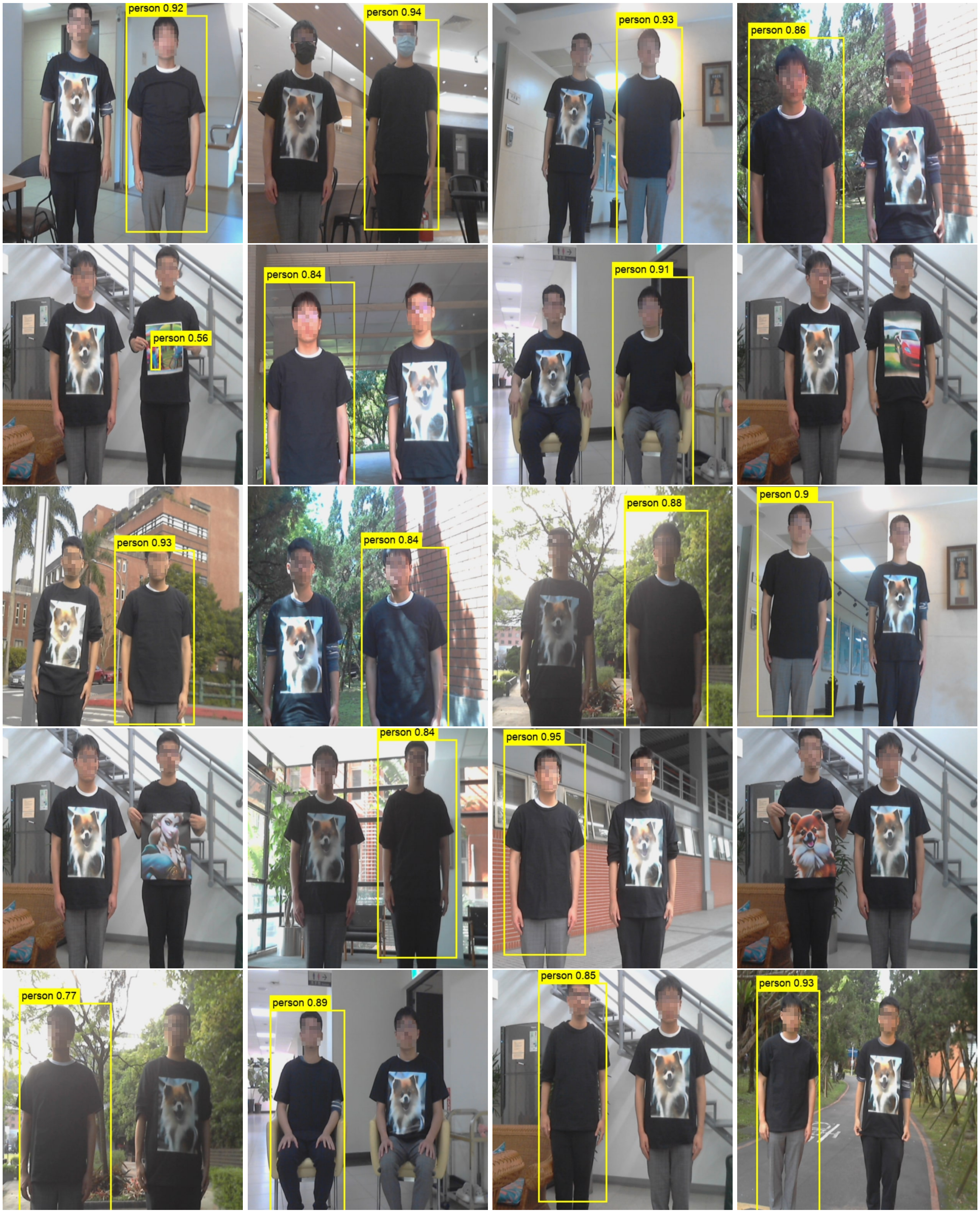


Fig. 6. **We further illustrate the physical evaluation results in the main paper using the sampled frames from different videos under various scenes.**