
Diffusion Policy Attacker: Crafting Adversarial Attacks for Diffusion-based Policies

Yipu Chen*

Georgia Institute of Technology
ychen3302@gatech.edu

Haotian Xue*

Georgia Institute of Technology
htxue.ai@gatech.edu

Yongxin Chen

Georgia Institute of Technology
yongchen@gatech.edu

Abstract

Diffusion models (DMs) have emerged as a promising approach for behavior cloning (BC). Diffusion policies (DP) based on DMs have elevated BC performance to new heights, demonstrating robust efficacy across diverse tasks, coupled with their inherent flexibility and ease of implementation. Despite the increasing adoption of DP as a foundation for policy generation, the critical issue of safety remains largely unexplored. While previous attack attempts have targeted deep policy networks, DP used diffusion models as the policy network, making it ineffective to be attacked using previous methods because of its chained structure and randomness injected. In this paper, we undertake a comprehensive examination of DP safety concerns by introducing adversarial scenarios, encompassing offline and online attacks, and global and patch-based attacks. We propose DP-Attacker, a suite of algorithms that can craft effective adversarial attacks across all aforementioned scenarios. We conduct attacks on pre-trained diffusion policies across various manipulation tasks. Through extensive experiments, we demonstrate that DP-Attacker has the capability to significantly decrease the performance of DP for all scenarios. Particularly in offline scenarios, DP-Attacker can generate highly transferable perturbations applicable to all frames. Furthermore, we illustrate the creation of adversarial physical patches that, when applied to the environment, effectively deceive the model. Video results are put in: <https://sites.google.com/view/diffusion-policy-attacker>.

1 Introduction

Behavior Cloning (BC) [40] is a pivotal area in robot learning: given an expert demonstration dataset, it aims to train a policy network in a supervised approach. Recently, diffusion models [16, 49] have become dominant in BC, primarily due to their strong capability in modeling multi-modal distribution. The resulting policy learner, termed Diffusion Policy (DP) [9, 18], can generate the action trajectory from a pure Gaussian noise conditioned on the input image(s). An increasing number of works are adopting DP as an action decoder for BC across various domains such as robot manipulation [12, 58, 7], long-horizon planning [35, 26] and autonomous driving [29].

Adversarial attack [31, 14] has been haunting deep neural networks (DNN) for a long time: a small perturbation on the input image will fool the DNN into making wrong decisions. Despite the remarkable success of diffusion policies in BC, their robustness under adversarial attacks [31, 14]

* indicates equal contribution. Correspondence to: ychen3302@gatech.edu

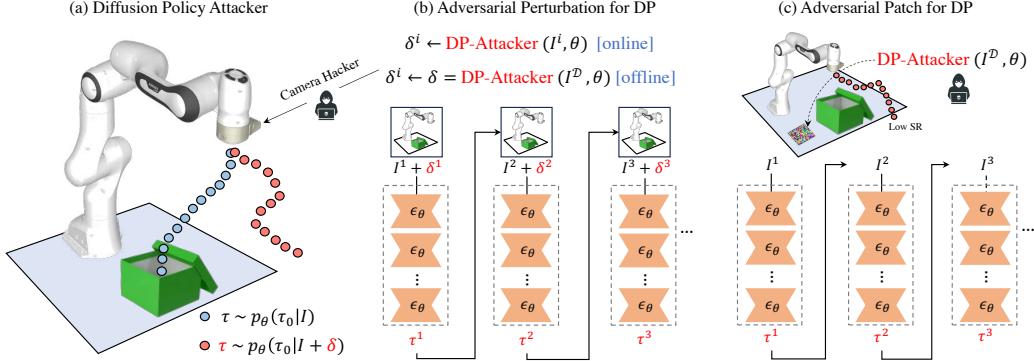


Figure 1: **Adversarial Attacks against Diffusion Policy:** We aim to attack robots controlled with visual-based DP, unveiling hidden threats to the safe application of diffusion-based policies. (a) By hacking the visual inputs, we can fool the diffusion process into generating wrong actions τ (in red). We propose Diffusion Policy Attacker(DP-Attacker), which can effectively attack the DP by (b) hacking the global camera inputs I using small visual perturbations under both online and offline settings or (c) attaching an adversarial patch into the environment. The online settings use current visual inputs at t -th timestep I^t to generate time-variant perturbations δ^t , while the offline settings use only offline data I^D to generate time-invariant perturbations δ .

remains largely unexplored, posing a potential barrier and risk to their broader application. While it is straightforward to attack an end-to-end DNN by applying gradient ascent over the loss function [31, 14], it is non-trivial to craft attacks against a DP, due to its concatenated denoising structure and high randomness. Prior research [25, 24, 56, 55, 45] has focused on attacking the diffusion process of the text-to-image (T2I) diffusion models [42]. However, there are distinct differences between attacking a T2I diffusion model and attacking a Diffusion Policy. Firstly, they concentrate on attacking the diffused value while we aim at attacking the conditional image. In addition, they try to fool the editing process over the clean images (e.g. SDEdit [34]), while we are trying to fool the robot to make wrong actions step by step, each action is generated from the pure gaussian noise. Diffusion policies are also more interactive with the environment. A successful attack not only needs to fool a single inference output but also needs to continuously fool the model to decrease model performance.

In this paper, we focus on crafting adversarial attacks against DP. Specifically, we propose Diffusion Policy Attacker (DP-Attacker), the first suite of white-box-attack algorithms that can effectively deceive the visual-based diffusion policies. We investigate two hacking scenarios as illustrated in Figure 1: (1) digital attack–hacking the scene camera, which means that we can add imperceptible digital perturbations to the visual inputs of DP, and (2) physical attack–hacking the scene by attaching small adversarial patches [4] to the environments (e.g. table). Furthermore, we consider both offline and online settings, for online settings, we can generate time-variant perturbations based on the current visual inputs, on the opposite, for the offline settings we can only add one fixed perturbation across all the frames.

We conducted extensive experiments on DP pre-trained on six robotic manipulation tasks and demonstrated that DP-Attacker can effectively craft adversarial attacks against DP. For digital attacks, DP-Attacker can generate both online and offline attacks that significantly degrade the DP system’s performance. For physical attacks, DP-Attacker is capable of creating adversarial patches tailored for each task, which can be put into the physical environment to disrupt the system. Also, we reveal that the non-robust image encoder makes the DP easy to attack.

2 Related Works

Diffusion-based Policy Generation Diffusion models [49, 16, 48] exhibit superior performance in multiple domains like high-fidelity image generation, video generation and 3D generation [42, 39, 44, 47, 53, 41, 27]. Due to its strong expressiveness in modeling multi-modal distribution, diffusion models have also been successfully applied to robot learning areas such as reinforcement learning [54, 2], imitation learning [9, 58, 20, 38], and motion planning [43, 30, 18]. Among them, Diffusion policy (DP) [9, 58, 23] has gained significant attention due to its straightforward training

methodology and consistent, reliable performance. In this paper, we focus on crafting adversarial attacks against visual-based DP, a technology already integrated into various indoor robot prototypes like Mobile Aloha [12].

Adversarial Examples for Deep Systems Adversarial attacks have been widely studied for deep neural networks (DNNs): given a small perturbation, the DNN will be fooled to make wrong predictions [51, 14]. For DNN-based visual recognition models, crafting adversarial samples is a relatively easy task using gradient-based budget-limited attacks [31, 57, 14, 5, 10, 3]. However, attacking diffusion models consisting of a cascade of DNNs injected with noise, poses a more complex challenge. Recent studies have demonstrated the feasibility of effectively crafting adversarial samples for latent diffusion models using meticulously designed surrogate losses [25, 59, 24, 46, 45, 56, 6]. However, these efforts have primarily focused on image editing or imitation tasks and are limited to working solely in latent space [55]. Here we hope to explore the adversarial attacks against DP under various settings.

Adversarial Threats against Robot Learning Previous research has highlighted adversarial attacks as a significant threat to robot learning systems [8], where small perturbations can cause chaos in applications such as deep reinforcement learning [22, 13, 28, 37, 50, 36], imitation learning [15], robot navigation [21], robot manipulation [19, 33], and multi-agent robot swarms [1]. Despite the rising popularity of policies generated by diffusion models, to the best of our knowledge, there have been no prior efforts aimed at attacking these models in the field of robotics.

3 Preliminaries

3.1 Diffusion Models for Behaviour Cloning

Diffusion models [49, 16] are one type of generative model that can fit a distribution $q(x_0)$, using a diffusion process and a denoising process. Starting from x_K , a pure Gaussian noise, the denoising process can generate samples from the target distribution by K iterations of denoising steps (Here we use K, k to represent steps in diffusion and T, t for running timesteps of the robot scenarios):

$$x_k = \alpha_k(x_{k+1} - \lambda_k \epsilon_\theta(x_k, k) + \mathcal{N}(0, \sigma_k^2 I)), \quad k = 0, 2, \dots, K-1 \quad (1)$$

where $\alpha_k, \lambda_k, \sigma_k$ are hyper-parameters for the noise scheduler. ϵ_θ is a learned denoiser parameterized by θ , which can be trained by optimizing the denoising loss termed $\mathcal{L} = \mathbb{E}_{x,k} \|\epsilon_\theta(x + \epsilon_k, k) - \epsilon_k\|^2$. We define the reverse process in Equation 1 as $x_k = \mathcal{R}_\theta^k(x_{k+1})$ for simplicity.

Diffusion policies [18, 9] noted π_θ apply the diffusion models mentioned above, resulting in $\tau^t \sim \pi_\theta(s^t)$, where $\tau^t \in \mathbb{R}^{D_a \times L_a}$ is the planned action sequences at timestep t in the continuous space, s^t is the current states, and D_a, L_a are the action dimension and action length respectively. Accordingly, the learnable denoiser becomes $\epsilon_\theta(\tau_k, k, s)$, and the denoised diffusion process remains the same. For visual DP, the states s^t are usually images captured by the scene or wrist cameras, so we use I^t throughout to represent the visual inputs at timestep t . Finally, the policy can be formulated as

$$\tau^t \sim \pi_\theta(I^t) = \mathcal{R}_\theta^0(\mathcal{R}_\theta^1 \dots \mathcal{R}_\theta^{K-2}(\mathcal{R}_\theta^{K-1}(x_K, I^t) \dots I^t), I^t). \quad (2)$$

The equation above shows that the predicted action τ^t is the output of chained denoiser models residually conditioned on the current observation I^t . In practice, while DP outputs a long sequence of actions τ , we only execute the first few actions of it in a receding horizon manner to improve temporal action consistency [9].

3.2 Adversarial Attacks Against Diffusion Models

Adversarial samples [14, 31, 5] have been widely studied as a threat to the AI system: for a DNN-based image classifier $y = f_\theta(x)$, one can easily craft imperceptible perturbations \mathcal{P} to fool the classifier to make wrong predictions over $\mathcal{P}(x)$. In digital attack settings [51, 14], the perturbation should be small and always invisible to humans, which can be formulated by the ℓ_∞ -norm as $|\mathcal{P}(x) - x|_\infty < \sigma$ where σ is a small value (e.g. 8/255 for pixel value). Methods like FGSM [14] and PGD [31] can be easily applied to craft such kinds of adversarial attacks. For physical-world

adversarial patches [4, 11, 57, 17], $\mathcal{P}(x)$ is always crafted as attaching a small adversarial patch to the environments, and the patch should be robust to physical-world transformations such as position, camera view, and lighting condition.

Recent works [25, 45] show that it is also possible to craft such kind of adversarial examples to fool latent diffusion models [42] with an encoder \mathcal{E} and a decoder \mathcal{D} : adding small perturbation to a clean image, the denoising process will be fooled to generate bad editing or imitation results. The following Monte-Carlo-based adversarial loss to attack a latent diffusion model:

$$\mathcal{L}_{adv}(x) = \mathbb{E}_k \|\epsilon_\theta(\mathcal{E}(x) + \epsilon_k, k) - \epsilon_k\|_2^2 \quad (3)$$

the mechanism behind attacking latent diffusion models [56] turns out to be the vulnerability of the autoencoder and works only for the diffusion model in the latent space [55]. Also, the settings above differs from our settings of attacking a DP which targets on attacking the conditional image without the ground-truth clean action to get the diffused input of ϵ_θ in Equation 3. In the following section, we show that we can still effectively craft different kinds of adversarial samples based on Equation 3 with some modification.

4 Methods

4.1 Problem Settings

Threat Model In this paper, we assume that we have white-box access to some diffusion policy network. That is, we have access to its parameters and also the data used to train it. Given this trained network, we wish to find adversarial perturbations that, when added to the observation I , will cause the trained diffusion policy to generate unwanted actions (either random or targeted) that impede task completion (lower the task score or success rate). We consider two types of perturbations detailed in Sec. 4.2 and Sec. 4.3.

The most straightforward way to measure the quality of the attack is to use the difference between generated actions from the original actions in an end-to-end manner:

$$\mathcal{L}_{end2end}^{\text{untar}}(I, t) = -\|\pi_\theta(\mathcal{P}(I)) - \tau^{t,*}\|_2^2 \quad (3)$$

where $\tau^{t,*}$ is a known good solution sampled by π_θ given the observation image I , and $\mathcal{P}(\cdot)$ is some perturbation on the observation image. It could be generated either from the trained policy for online attacks or from the training dataset for offline attacks. One can minimize the negative L-2 distance between a generated action and a good action for untargeted attacks. For targeted attacks, the action loss becomes

$$\mathcal{L}_{end2end}^{\text{tar}}(I, t) = \|\pi_\theta(\mathcal{P}(I)) - \tau_{\text{target}}^t\|_2^2 \quad (4)$$

where τ_{target}^t is some target bad action we wish the policy to execute (e.g. always move to left). We can use PGD [31] to optimize for the best perturbation that minimizes this loss. However, due to the inherent long-denoising chain of the diffusion policy π_θ , the calculation of this gradient could be quite costly [45].

In practice, running the end-to-end attacks above is not effective especially when the model is large and when we need to hack the camera at a high frequency. Instead, borrowing ideas from recent works [25, 24, 56] on adv-samples for diffusion models, we propose to use the following optimization objectives:

$$\mathcal{L}_{adv}^{\text{untar}}(I, t) = -\mathbb{E}_k \|\epsilon_\theta(\tau^{t,*} + \epsilon_k, k, \mathcal{P}(I)) - \epsilon_k\|_2^2 \quad (5)$$

where k is the timestep of the diffusion process and t is the timestep of the action runner. We add noise to the good solution $\tau^{t,*}$ and then calculate the L-2 distance between the predicted noise of the

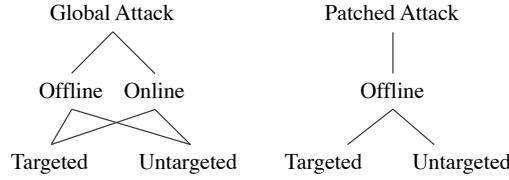


Figure 2: **Design Space of DP-Attacker**: the tree above shows the design space of DP-Attacker, which can be adapted to various kinds of attack scenarios, including global attacks (hacking and cameras) vs patched attacks (hacking the physical environment); offline vs online; targeted vs untargeted.

denoise network and the added noise. Minimizing this loss leads to inaccurate noise prediction of the denoising network and, in turn, leads to bad generated action of the diffusion policy. For targeted attacks, the noise prediction loss is:

$$\mathcal{L}_{\text{adv}}^{\text{tar}}(I, t) = \mathbb{E}_k \|\epsilon_{\theta}(\tau_{\text{target}}^t + \epsilon_k, k, \mathcal{P}(I)) - \epsilon_k\|^2 \quad (6)$$

Minimizing this loss would allow the denoising net to favor the generation of the target action. The gradient of the noise prediction loss is easier to calculate compared to the action loss because of the short one-step chain. This makes it more favorable for conducting attacks.

4.2 Global Attacks

A global attack injects adversarial perturbation δ into the observation image I by adding it on top of the observation image, *i.e.* $\mathcal{P}(I) = I + \delta$. The adversarial noise δ is of the same shape as the original image. To make the attack imperceptible, the adversarial noise's absolute value is limited by some σ . To find such an adversarial noise, we use PGD [31], an optimization-based method to search for an adversarial noise. The adversarial noise can be constructed online during inference or offline using the training dataset. The algorithm for conducting an online global attack is shown in Algorithm 1. The algorithm optimizes for loss in Equation 5 or Equation 6. The algorithm can be modified easily to construct an offline attack. Given the training dataset $D_T = \{(\tau^t, I^t) | t \in T\}$ we can optimize for the loss $\mathcal{L}_{\text{adv}}^{\text{untar}}(I, t) = -\mathbb{E}_{k, (\tau^t, I^t)} \|\epsilon_{\theta}(\tau^t + \epsilon_k, k, \mathcal{P}(I)) - \epsilon_k\|^2$ or $\mathcal{L}_{\text{adv}}^{\text{tar}}(I, t) = \mathbb{E}_{k, (\tau^t, I^t)} \|\epsilon_{\theta}(\tau_{\text{target}}^t + \epsilon_k, k, \mathcal{P}(I)) - \epsilon_k\|^2$. This algorithm is provided in the appendix.

Algorithm 1 Global Adversarial Attack (Online)

Input: given observation image I , diffusion policy π_{θ} , noise prediction net ϵ_{θ} , attack budget σ , step size α , number of steps N
Output: adversarial noise δ

```

 $\delta \leftarrow 0$  ▷ initialize adversarial noise
for  $i = 1$  to  $N$  do
   $\epsilon_k, k \sim \mathcal{N}(0, I)$ ,  $\text{randint}(1, K)$  ▷ sample forward noise and timestep
  if targeted attack then
     $\tau^t \leftarrow \tau_{\text{target}}^t + \epsilon_k$  ▷ forward sample,  $\tau_{\text{target}}^t$  should be given
     $s \leftarrow 1$ 
  else if untargeted attack then
     $\tau^t \sim \pi_{\theta}(I)$  ▷ use diffusion policy to generate a good solution
     $\tau^t \leftarrow \tau^t + \epsilon_k$  ▷ forward sample
     $s \leftarrow -1$ 
  end if
   $\epsilon_p \leftarrow \epsilon_{\theta}(\tau^t, k, \text{clip}(I + \delta, 0, 1))$ 
   $\mathcal{L} \leftarrow s \cdot \|\epsilon_k - \epsilon_p\|^2$ 
   $\delta \leftarrow \text{clip}(\delta - \alpha \cdot \text{sign}(\nabla_{I_{\text{adv}}} \mathcal{L}), -\sigma, \sigma)$  ▷ Projected-Gradient Descent
end for
return  $\delta$ 

```

4.3 Patched Attacks

A patched attack directly puts a specifically designed image patch $x \in \mathbb{R}^{c \times h \times w}$ into the environment. The camera later captures it and causes undesirable motion from the diffusion policy. The patch should be active under different scales, orientations, and observation views. During training, we apply some random affine transform (shift, rotation, scale, and shear) $\mathcal{T} \in \mathbb{T}$. The affine transform uses the center of the image as the origin of the coordinate system. The resulting patch replaces the original observation image using the replacement operator: $\text{replace}(I, x)$ again using the image's center as the origin of the coordinate system. To search for such a patch, we use the training dataset and optimize for the best patch using PGD. The algorithm is illustrated in Algorithm 2.

Algorithm 2 Patch Adversarial Attack

Input: training dataset $D_T = \{(\tau^t, I^t) | t \in T\}$, diffusion policy π_θ , noise prediction net ϵ_θ , set of affine transforms \mathbb{T} , step size α
Output: adversarial patch x

```
 $x \sim \mathcal{N}(0, I)$ 
 $x \leftarrow \text{clip}(x + 0.5, 0, 1)$  ▷ initialize a patch
repeat
   $(\tau^t, I^t), \mathcal{T}, \epsilon_k, k \sim D_T, \mathbb{T}, \mathcal{N}(0, I), \text{randint}(1, K)$  ▷ sample from dataset, transform, forward noise, and time step
  if targeted attack then
     $\tau^t \leftarrow \tau_{\text{target}}^t + \epsilon_k$  ▷ forward sample,  $\tau_{\text{target}}^t$  should be given
     $s \leftarrow 1$ 
  else if untargeted attack then
     $\tau^t \leftarrow \tau^t + \epsilon_k$  ▷ forward sample, use dataset entry
     $s \leftarrow -1$ 
  end if
   $\epsilon_p \leftarrow \epsilon_\theta(\tau^t, k, \text{replace}(I^t, \mathcal{T}(x)))$ 
   $\mathcal{L} \leftarrow s \cdot \|\epsilon_k - \epsilon_p\|^2$ 
   $x \leftarrow \text{clip}(x - \alpha \cdot \text{sign}(\nabla_x \mathcal{L}), 0, 1)$  ▷ Projected-Gradient Descent
until satisfied
return  $x$ 
```

5 Experiments

We test the effectiveness of DP-Attacker with various strengths and configurations on different diffusion policies. Our target models are vision-based diffusion policy models introduced by Chi et al. [9]. We aim to manipulate the visual input so that the generated trajectory will not lead to task completion. We quantitatively evaluate the effectiveness of our attack methods by recording the result task completion scores/successful rate. We also provide scores without attacks for reference and random noise attacks (adding some Gaussian noise to the observation images) as a baseline attack method. We focus on the models released by Chi et al. [9]. However, our attack algorithm applies to other variants of diffusion policies as well.

Environment Setup Our benchmark contains 6 tasks: PushT, Can, Lift, Square, Transport, and Toolhang. These tasks are illustrated in Figure 7 in the Appendix. Robosuite provides all the simulation of these tasks except for PushT [52, 32, 60]. For evaluation, we attack the released checkpoints of diffusion policies trained by Chi et al. [9]. For tasks Can, Lift, Square, and Transport, each has two demonstration datasets: Multi-Human (MH) and Proficient Human (PH). The other two tasks (PushT and Toolhang) has only one PH dataset, respectively. This gives us a total of 10 datasets. In [9], each dataset is used to train two diffusion policies with different diffusion backbone architectures: CNN-based and Transformer-based. We take the best performing checkpoints for these 20 different scenarios released by Chi et. al [9] as our attack targets. For each attack method, we run 50 rollouts and collect the average score or calculate the success rate of the tasks. The rollout length uses the same length as the demonstration dataset [9, 32]. Besides our attack methods, we also run the rollout using clean images for reference and with random noise added as a baseline attack method. The evaluation is done using a single machine with an RTX 3090 GPU and AMD Ryzen 9 5950X to calculate rollouts and run our attack algorithms.

5.1 Global Attack

We first present the results of global attacks. We evaluate both our online attack algorithm (creating adversarial noise on the fly per inference) and offline algorithm (pre-generating a fixed noise that is used for every inference).

Online Attack For online attacks, we use attack parameters $\sigma = 0.03, \alpha = 0.001875, N = 50$. For targeted attacks, we use a normalized target action vector of all ones. We report the performance of the transformer-based models before and after the attack in Table 1. The results of global attacks



Figure 3: **Global Attack (Online)**: We visualize the global attacks in Algorithm 1 within both the PushT and Can environments. Specifically, we present action rollouts for four types of observations: clean observations, observations perturbed with random Gaussian noise, and our optimized perturbations (both untargeted and targeted). While the DPs show robustness to random perturbations, they are vulnerable to adversarial samples generated using DP-Attacker.

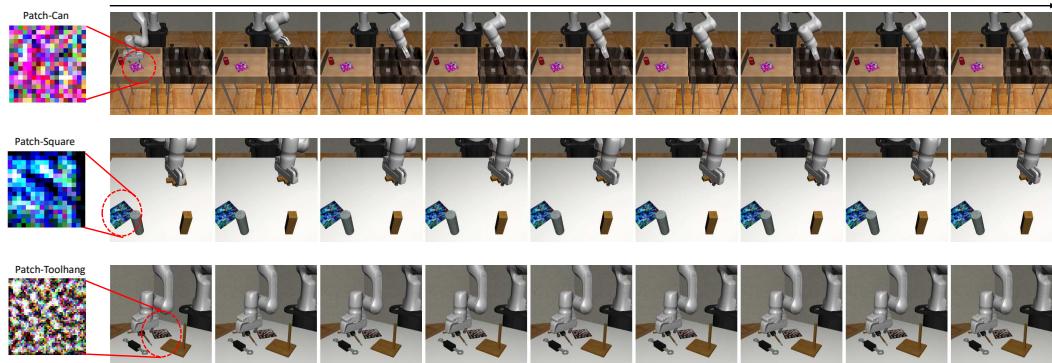


Figure 4: **Physical Adversarial Patches**: we show the patches optimized by Algorithm 2, attaching it to the physical scene will effectively lower the success rate of the target diffusion policy.

Tasks Demonstration Type	PushT		Can		Lift		Square		Transport		Toolhang
	PH	PH	MH	PH	MH	PH	MH	PH	MH	PH	PH
Clean	0.75	0.92	0.92	1	1	0.92	0.72	0.86	0.46	0.86	
Random Noise	0.66	0.88	0.98	1	1	0.82	0.74	0.84	0.48	0.82	
Targeted-Offline	0.46	0.08	0.08	0.94	0.7	0	0	0	0.02	0	
Untargeted-Offline	0.39	0.1	0.46	0.8	0.62	0.04	0	0	0	0	
Targeted-Online	0.10	0	0	0.02	0	0	0	0	0	0	
Untargeted-Online	0.19	0.02	0.02	0.62	0.62	0	0	0	0	0	

Table 1: **Quantitative Results on Global Attacks:** The table includes the attack result for all transformer based diffusion policy networks. Our DP-Attack can significantly lower the performance of the diffusion models.

Backbone Arch	Can		Lift		Square		Toolhang	
	CNN	Transformer	CNN	Transformer	CNN	Transformer	CNN	Transformer
Clean	0.98	0.92	1	1	0.94	0.92	0.8	0.86
Random Noise Patch	0.9	0.94	1	0.9	0.8	0.54	0.56	0.12
Untargeted-Offline	0.16	0.44	1	0.82	0.72	0.34	0.48	0.02

Table 2: **Quantitative Results on Patched Attacks**

on all models are given in the appendix. Example rollouts and images used in the rollouts are shown in Figure 3.

Offline Attack For offline global attacks, we train on the training dataset with batch size 64, $\alpha = 0.0001$, $\sigma = 0.03$ for 10 epochs. The resulting trained adversarial noise is added to the input image for every inference. The results are shown in Table 1. Examples of rollouts and images used in the attack can be found on our website.

We find that diffusion policy is not robust to noises introduced by our DP-Attacker. The performance of diffusion policies is significantly reduced after running global attacks. A disturbance of less than 3% is able to decrease the performance from 100% to 0%. The success of offline global attacks also shows attacks can be cheaply constructed and pose a significant threat to the safety of using diffusion policy in the real world.

5.2 Patched Attack

Training vs. Evaluating Since patched attacks directly put an attack image into the environment, we only consider offline attacks that pre-generate some patch that is used throughout the rollout. We train the patch using Algorithm 2, where the patch is applied to the training image using some randomized affine transform. This allows the gradient to pass through for successful training. Since we have used random affine transforms during training, the patch should be transferable when used in the simulation environment. For evaluation, we create a thin box object with the trained image patch as its texture and put it randomly onto the table.

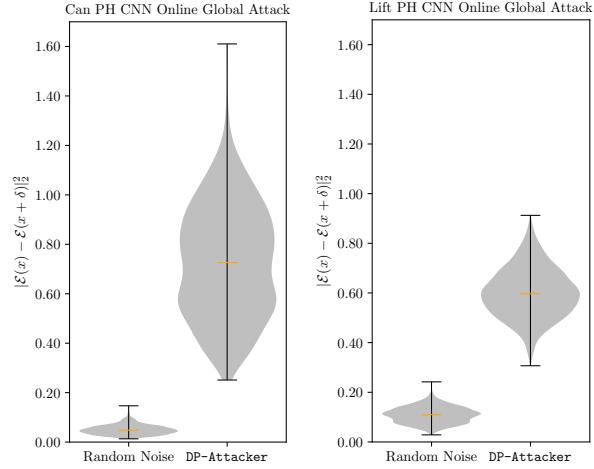


Figure 5: **Difference in Encoded Feature Vector:** we calculate the distance between the clean feature vector and the attacked feature vector. DP-Attacker perturb the feature vector significantly compared to naive random noise attack.

Parameters ($\sigma = 0.03$)	$N = 10$	$N = 20$	$N = 50$
Success Rate	0.94	0.8	0.66
Parameters ($N = 50$)	$\sigma = 0.01$	$\sigma = 0.03$	$\sigma = 0.05$
Success Rate	1	0.68	0.32

Table 3: **Different Parameters for DP-Attack:** We did an ablation study on parameters σ and N , and we can see that smaller steps and budgets are not enough to fool a DP. Larger budgets will dramatically decrease the Success Rate (SR).

Results We construct a patch of size that covers around 5% of the observation image using Algorithm 2. The details of the training can be found in the appendix. We evaluate the effectiveness of our patch attack algorithm on a total of 8 checkpoints, covering the PH dataset across four tabletop manipulation tasks (Can, Lift, Square, and Toolhang) using both CNN and Transformer diffusion backbones. The result success rate (SR) is shown in Table 2. Example rollouts are shown in Table 4.

Simpler tasks such as Can and Lift are quite robust to random noise patch. Our DP-Attacker produces adversarial patches that perform better than random noise in terms of degrading the diffusion policy performance.

5.3 Quantitative Results on Targeted Attacks

We qualitatively evaluate the effectiveness of our targeted attacks. We use our DP-Attacker to run global-online-targeted attacks with varying strength on two model checkpoints: PushT (CNN) and CAN (PH CNN). The target in PushT task is a 2D coordinate around (323.875, 328.75) (note the side length of the PushT environment is 1024), and the target in the CAN task is the target end-effector position around (0.1686, 0.1049, 1.0848) (in meters). In 6, we report how close the actions generated by diffusion policy are to our attack targets during the rollout.

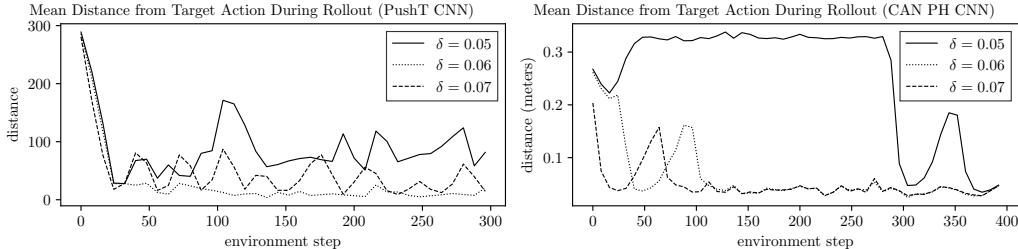


Figure 6: We used DP-Attacker with different attack strengths to run rollouts. We report the average distance between the predicted action sequence and the target action sequence (a sequence of a duplicate target coordinate). The metric is calculated at each inference during the rollout.

Our DP-Attacker is able to manipulate the generated action to be within 20 units and 5 cm of the attack targets, respectively, for the PushT task and CAN task, with an attack strength of $\delta = 0.06$. Example rollouts of these two attack scenarios can be found in the Sec. D of the appendix.

6 Ablation Study

Attack Parameters To investigate the effectiveness of our attack method, we evaluate how the attack parameter plays a role in DP-Attacker. First, we investigate the effect of the number of PGD steps N . We keep the $\sigma = 0.03$, and $\alpha = \frac{2\sigma}{N}$. Second, we investigate the effect of the noise scale σ . We keep $N = 50$, and $\alpha = \frac{2\sigma}{N}$. We evaluate all six attacks on the transformer backbone DP trained on the Lift PH dataset. The result is summarized in table 3.

End to End Loss vs. Noise Prediction Loss We perform a comparison with the end-to-end action loss 3. We evaluate both methods with the same attack parameters ($\sigma = 0.03$, $\alpha = 0.001875$, $N = 50$) on the best-performing transformer backbone trained on the PH dataset of the Lift task. Again, we evaluate 50 randomly initialized environments. The selection of end to end loss with DDPM [16]

Method	Attack Time	Model Success Rate
Clean	-	1
Random Noise	-	1
End to End DDPM (Untargeted)	~70s	1
End to End DDPM (Targeted)	~67s	0.52
End to End DDIM-8 (Untargeted)	~6.5s	0.9
End to End DDIM-8 (Targeted)	~5.8s	0.24
DP-Attacker (Untargeted)	~1.8s	0.62
DP-Attacker (Targeted)	~1.3s	0.02

Table 4: **Compared with End to End Attacks** DP-Attacker runs significantly faster than the end-to-end attacks even if it is accelerated with DDIM. Our DP-Attacker also provides better attack results.

scheduler makes it infeasible for online attacks. In addition, we provide results where we replace the loss-calculating noise scheduler with a DDIM-8 step scheduler [48]. This provides speedup for calculating the end to end loss. The result SR after the attack and the average time used to perform the online attacks are shown in 4. The naive end-to-end loss is significantly lower than our attack algorithms and does not provide better results. We suspect that since diffusion models introduce randomness during the sampling of a trajectory, it is better to attack the noise prediction loss rather than the end to end action loss.

What Is Being Attacked Is the Encoder We try to investigate further what exactly is attacked in our DP-Attacker. Other literature relating to text-to-image diffusion models shows that the encoder is the one being attacked [45, 56]. We suspect the same is happening for diffusion policy. To investigate this, we calculate the L2 distance between the encoded feature vector of clean and attacked images random noise attack, unsuccessful attack parameters, and successful parameters, respectively. The details of the calculation is in the appendix. We do this for 1000 images in the training dataset and plot the distribution of the distances using a violin plot in Figure 5. The significant difference shows that our attack method has drastically changed the representation of the conditional visual feature. This later affects the downstream conditional noise prediction net, causing it to make inaccurate noise predictions. We put details about it in the Appendix.

7 Conclusion and limitations

In this paper, we propose DP-Attacker, a suite of algorithms designed to effectively attack diffusion-based policy generation, an emerging approach in behavior cloning. We demonstrate that DP-Attacker can craft adversarial examples across various scenarios, posing a significant threat to systems reliant on DP. Our findings highlight that despite the inherent randomness and cascaded deep structure of diffusion-based policy generation, it remains vulnerable to adversarial attacks. We emphasize the need for future research to focus on enhancing the robustness of DP to ensure its reliability in real-world applications. There are also some limitations for this paper: our experiments were conducted exclusively within a simulation environment, and we did not extend our testing to real-world scenarios. Additionally, we did not develop or implement any defensive strategies for the proposed tasks, which remains an area for future research and exploration.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their valuable feedback on this paper. This work is supported in part by grants NSF CAREER ECCS-1942523 and NSF FRR-2409016.

References

- [1] M. Abouelyazid. Adversarial deep reinforcement learning to mitigate sensor and communication attacks for secure swarm robotics. *Journal of Intelligent Connectivity and Emerging Technologies*, 8(3):94–112, 2023.

- [2] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.
- [3] A. Arnab, O. Miksik, and P. H. Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 888–897, 2018.
- [4] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.
- [5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.
- [6] J. Chen, J. Dong, and X. Xie. Exploring adversarial attacks against latent diffusion model from the perspective of adversarial transferability. *arXiv preprint arXiv:2401.07087*, 2024.
- [7] L. Chen, S. Bahl, and D. Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.
- [8] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han. Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity*, 2:1–22, 2019.
- [9] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- [10] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.
- [11] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1000–1008, 2020.
- [12] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.
- [13] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.
- [14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [15] G. Hall, A. Das, J. Quarles, and P. Rad. Studying adversarial attacks on behavioral cloning dynamics. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 452–459. IEEE, 2020.
- [16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- [17] Z. Hu, S. Huang, X. Zhu, F. Sun, B. Zhang, and X. Hu. Adversarial texture for fooling person detectors in the physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13307–13316, 2022.
- [18] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.
- [19] Y. Jia, C. M. Poskitt, J. Sun, and S. Chattopadhyay. Physical adversarial attack on a robotic arm. *IEEE Robotics and Automation Letters*, 7(4):9334–9341, 2022.
- [20] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.
- [21] M. I. Khedher and M. Rezzoug. Analyzing adversarial attacks against deep learning for robot navigation. In *ICAART (2)*, pages 1114–1121, 2021.
- [22] M. Lechner, A. Amini, D. Rus, and T. A. Henzinger. Revisiting the adversarial robustness-accuracy tradeoff in robot learning. *IEEE Robotics and Automation Letters*, 8(3):1595–1602, 2023.
- [23] X. Li, V. Belagali, J. Shang, and M. S. Ryoo. Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. *arXiv preprint arXiv:2307.01849*, 2023.

[24] C. Liang and X. Wu. Mist: Towards improved adversarial examples for diffusion models. *arXiv preprint arXiv:2305.12683*, 2023.

[25] C. Liang, X. Wu, Y. Hua, J. Zhang, Y. Xue, T. Song, Z. Xue, R. Ma, and H. Guan. Adversarial example does good: Preventing painting imitation from diffusion models via adversarial examples. In *International Conference on Machine Learning*, pages 20763–20786. PMLR, 2023.

[26] Z. Liang, Y. Mu, H. Ma, M. Tomizuka, M. Ding, and P. Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. *arXiv preprint arXiv:2312.11598*, 2023.

[27] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. Magic3d: High-resolution text-to-3d content creation. *arXiv preprint arXiv:2211.10440*, 2022.

[28] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[29] J. Liu, P. Hang, X. Zhao, J. Wang, and J. Sun. Ddm-lag: A diffusion-based decision-making model for autonomous vehicles with lagrangian safety enhancement. *arXiv preprint arXiv:2401.03629*, 2024.

[30] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du. Potential based diffusion motion planning. 2023.

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[32] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[33] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, and F. Roli. Is deep learning safe for robot vision? adversarial examples against the icub humanoid. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 751–759, 2017.

[34] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[35] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

[36] K. Mo, W. Tang, J. Li, and X. Yuan. Attacking deep reinforcement learning with decoupled adversarial policy. *IEEE Transactions on Dependable and Secure Computing*, 20(1):758–768, 2022.

[37] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

[38] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[39] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[40] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[41] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[42] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[43] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna. Edmp: Ensemble-of-costs-guided diffusion for motion planning. *arXiv preprint arXiv:2309.11414*, 2023.

[44] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[45] H. Salman, A. Khaddaj, G. Leclerc, A. Ilyas, and A. Madry. Raising the cost of malicious ai-powered image editing. *arXiv preprint arXiv:2302.06588*, 2023.

[46] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao. Glaze: Protecting artists from style mimicry by text-to-image models. *arXiv preprint arXiv:2302.04222*, 2023.

[47] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[48] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[49] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[50] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5883–5891, 2020.

[51] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[52] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[53] Y. Wang, Z. Wu, C. Li, and A. Wu. Extraction and recovery of spatio-temporal structure in latent dynamics alignment with diffusion models. *Advances in Neural Information Processing Systems*, 36:38988–39005, 2023.

[54] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[55] H. Xue and Y. Chen. Pixel is a barrier: Diffusion models are more adversarially robust than we think. *arXiv preprint arXiv:2404.13320*, 2024.

[56] H. Xue, C. Liang, X. Wu, and Y. Chen. Toward effective protection against diffusion-based mimicry through score distillation. In *The Twelfth International Conference on Learning Representations*, 2023.

[57] H. Xue, A. Araujo, B. Hu, and Y. Chen. Diffusion-based adversarial sample generation for improved stealthiness and controllability. *Advances in Neural Information Processing Systems*, 36, 2024.

[58] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[59] B. Zheng, C. Liang, X. Wu, and Y. Liu. Understanding and improving adversarial attacks on latent diffusion model. *arXiv preprint arXiv:2310.04687*, 2023.

[60] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

Appendix

We put more video results, including rollouts of the DP-based robots under various attacks crafted by `DP-Attacker`, in the following anonymous link:

<https://sites.google.com/view/dp-attacker-videos/>.

A Broader Impact

Diffusion-based policies (DPs) have emerged as promising candidates for integrating real robots into our daily lives. Even with just a few collected demonstrations, DPs exhibit strong performance across various tasks [12, 58]. However, despite their utilization of diffusion models, which distinguish them from other policy generators, our research highlights their vulnerability to adversarial attacks. We demonstrate practical attacks on DP-based systems, such as hacking cameras to introduce fixed perturbations across all frames (global offline attack) and incorporating patterns into the scene (physical patched attack). It is critical to consider these threats, and we urge future research to prioritize the development of more robust DPs before their widespread application in the real world.

B Algorithms

We also provide the algorithm for training the offline global attacks.

Algorithm 3 Global Adversarial Attack (Offline)

Input: training dataset $D_T = \{(\tau^t, I^t) | t \in T\}$, diffusion policy π_θ , noise prediction net ϵ_θ , step size α , attack budget σ
Output: adversarial noise δ

```
 $\delta \leftarrow 0$  ▷ initialize adversarial noise
repeat
   $(\tau^t, I^t), \epsilon_k, k \sim D_T, \mathcal{N}(0, I)$ ,  $\text{randint}(1, K)$  ▷ sample from dataset, forward noise and timestep
  if targeted attack then
     $\tau^t \leftarrow \tau_{\text{target}}^t + \epsilon_k$  ▷ forward sample,  $\tau_{\text{target}}^t$  should be given
     $s \leftarrow 1$ 
  else if untargeted attack then
     $\tau^t \leftarrow \tau^t + \epsilon_k$  ▷ forward sample
     $s \leftarrow -1$ 
  end if
   $\epsilon_p \leftarrow \epsilon_\theta(\tau^t, k, \text{clip}(I^t + \delta, 0, 1))$ 
   $\mathcal{L} \leftarrow s \cdot \|\epsilon_k - \epsilon_p\|^2$ 
   $\delta \leftarrow \text{clip}(\delta - \alpha \cdot \text{sign}(\nabla_{I_{\text{adv}}} \mathcal{L}), -\sigma, \sigma)$  ▷ Projected-Gradient Descent
until satisfied
return  $\delta$ 
```

C Experimental Details

C.1 Task Descriptions

We investigate a total of six different tasks: PushT, Can, Lift, Square, Transport, and Tool hang. The tasks are illustrated in 7. Here are the descriptions for each task:

- PushT: The simulation happens in 2D. The agent controls a rod (blue circle) to push the grey T block into the targeted green area. The score calculated is the maximum percent of coverage of the green area by the grey T block during a rollout.

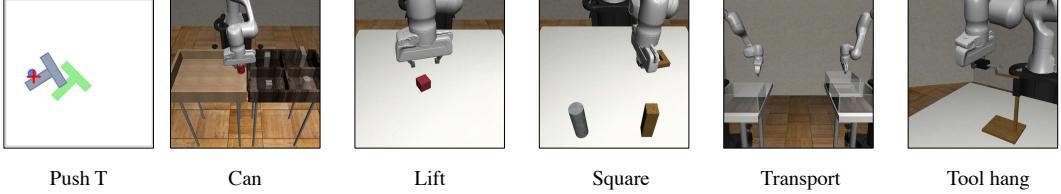


Figure 7: **Simulation renders of the six tasks**

- **Can:** The simulation environment is provided by robosuite [60]. The agent controls the 6-DoF end-effector position and gripper close or open. The goal is to move the randomly positioned can in the left bin into the corresponding bin (lower right) in the right bin.
- **Lift:** The simulation environment is provided by robosuite. The agent controls the 6-DoF end-effector position and gripper close or open. The goal is to lift up the randomly positioned red block.
- **Square:** The simulation environment is provided by robosuite. The agent controls the 6-DoF end-effector position and gripper close or open. The goal is to put the randomly positioned square nut around the square peg.
- **Transport:** The simulation environment is provided by robosuite. The agent controls 2 6-DoF end-effector positions and grippers close or open. The goal is to transport the hammer inside the box one one side to the box on the other side.
- **Tool hang:** The simulation environment is provided by robosuite. The agent controls the 6-DoF end-effector position and gripper close or open. The goal is to construct the tool tower by first inserting an L-shaped bar into the base and later hanging the second tool on the tip of the bar.

C.2 Training Details

Our goal is to construct noises for the observation images. The image encoder uses multiple views when constructing the conditional image feature vector. Below are details of how we construct the adversarial noises.

C.2.1 Global Online Attacks

For global online attacks, we construct noises for all observation views, i.e. suppose the encoder takes two camera views (one side view and one eye-in-hand), and the conditional state length is two, we will construct a total of four noises for adding onto the input images, respectively, before passing it into the policy for action generation.

For the random noise attack, the noise selected is sampled from a standard Gaussian and scaled by $\sigma = 0.03$ and clipped in the range $[-\sigma, \sigma]$. For untargeted online attacks, we use PGD parameters $N = 50, \sigma = 0.03, \alpha = 0.001875$. For targeted online attacks, the targeted selected is an action matrix (actim dim \times action horizon) of all 1's (in normalized action space). The PGD parameters for targeted online attacks are the same as the untargeted online attacks.

C.2.2 Global Offline Attacks

Similar to global online attacks, we also construct noise for all observation views. However, since this is an offline attack, we pre-generate (train) just one set of adversarial noises for each input, and it is used throughout the rollout for the same task. The training parameters for untargeted and targeted attacks are the same: number of epochs = 10, $\alpha = 0.0001$, and batch size = 64. For targeted attacks, we again use a normalized action of all 1's.

C.2.3 Patched Attacks

For patch attack training, we only apply the patch on the most important camera view (side view). However, this is to maintain the consistency for the patch gradient pass. However, the patch is put into

Parameter	Range
x shift	[-0.4, 0.4] percent of image, in centered coordinate.
y shift	[-0.4, 0.4] percent of image, in centered coordinate.
rotation	$[-45^\circ, 45^\circ]$
scale	[1, 1]
shear x	$[-50^\circ, 50^\circ]$
shear y	$[-50^\circ, 50^\circ]$

Table 5: Set of Affine Transforms \mathbb{T} for Patched Attack

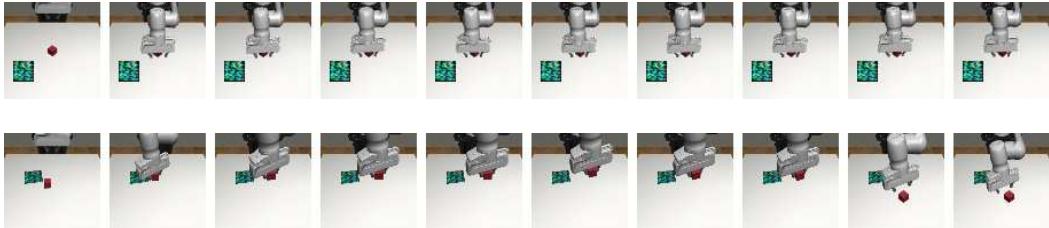


Figure 8: Comparison of Physical Patch Training and Evaluation

the simulation environment for evaluation and can be observed from multiple perspectives. For tasks Can, Lift, and Square, the observation image size is 84×84 , and we choose the training patch size of 17×17 that covers around 4% of the observation. For task Toolhang where the observation image size is 84×84 , we choose the training patch size of 50×50 that covers around 4.3% of the image. The set of transforms \mathbb{T} is summarized in the table 5. The training parameters are epochs = 10, batch size = 64, $\alpha = 0.0001$. For evaluation, we make patch objects of size $0.06m \times 0.06m$ (2.36in \times 2.36in) and put it onto the table. The rotation angle is from $[-45^\circ, 45^\circ]$. For tasks Can, Lift, and Square, the position of the patch can be anywhere on the table. For Toolhang, the position of the patch is constrained to be on the top left of the table so it can be captured by the camera. The size is about the same and we provide the comparison in the figure 8.

D More Results

Full Table for Global Attacks We provide the full table for the results of global attacks in 6 as an extension for 1. CNN-based models are harder to attack. Nevertheless, the result score still decreased significantly.

Targeted Attacks With more attack budget, we can manipulate the robot’s action quite well. For this experiment, we increase the online global attack budget σ to 0.05. With this increased budget, we could manipulate the generated action of DP. This shows the effectiveness of DP-Attacker proposed targeted noise prediction loss. See our website for details.

What Is Being Attacked Is the Encoder To investigate whether it is the encoder that is being attacked in DP-Attacker we perform the following comparison. For a given image, we find the encoded feature vector of the clean image $\mathcal{E}(x)$, clean image + random noise $\mathcal{E}(x + \delta_{\text{rand}})$, and clean image + adversarial noise $\mathcal{E}(x + \delta_{\text{adv}})$ by our DP-Attacker. Next we calculate the L2 distance between the encoded clean image vs the encoded random noise attacked image $|\mathcal{E}(x) - \mathcal{E}(x + \delta_{\text{rand}})|_2^2$, and the L2 distance between the encoded clean image vs the encoded DP-Attacker attacked images $|\mathcal{E}(x) - \mathcal{E}(x + \delta_{\text{adv}})|_2^2$. We collect the these two distances for 1000 images in the training dataset and plot the distribution of the two sets using violin plots 5. The attack we use are random noise attack with $\sigma = 0.03$ and online targeted global attacks with $\sigma = 0.03$, $N = 50$, $\alpha = 0.001875$, $n_{\text{target}} = 1$. We do this for two dataset, one is the CAN ph with CNN backbone where our DP-Attacker successfully performs the attack, and the other is the Lift ph with CNN backbone where our DP-Attacker fails to construct successful attacks (see table 6). The difference in distribution shows that successful attacks is correlated with the successful disturbance of the encoder.

Tasks Demonstration Type	PushT PH	PushT PH	Can MH	Can PH	Lift MH	Lift PH	Square MH	Square PH	Transport MH	Transport PH	Toolhang PH
Clean	0.75/0.83	0.92/0.98	0.92/0.98	1/1	1/1	0.92/0.94	0.72/0.84	0.86/0.88	0.46/0.82	0.86/0.8	
Random Noise	0.66/0.87	0.88/1	0.98/0.98	1/1	1/1	0.82/0.94	0.74/0.76	0.84/0.84	0.48/0.68	0.82/0.72	
Targeted-Offline	0.46/0.68	0.08/0.2	0.08/0.16	0.94/1	0.7/1	0/0.9	0/0.66	0/0.66	0.02/0.64	0/0	
Untargeted-Offline	0.39/0.73	0.1/0	0.46/0.34	0.8/1	0.62/0.98	0.04/0.62	0/0.68	0/0	0/0	0/0	
Targeted-Online	0.10/0.45	0/0	0/0	0.02/1	0/1	0/0.54	0/0.08	0/0	0/0	0/0	
Untargeted-Online	0.19/0.48	0.02/0.02	0.02/0.02	0.62/1	0.62/1	0/0.38	0/0.08	0/0.04	0/0.04	0/0	

Table 6: **Quantitative Results on Global Attacks:** The table includes the attack results for both CNN and transformer-based diffusion policy networks. The format is transformer/CNN. Our DP-Attack can significantly lower the performance of the diffusion models.

Method	Attack Time	Model Score
Clean	-	0.75
Random Noise	-	0.66
End to End DDPM	~44.6s	0.11
End to End DDIM-8	~4.3s	0.11
DP-Attacker Targeted-Online	~1.5s	0.11

Table 7: **Compared with End to End Attacks** DP-Attacker runs significantly faster than the end-to-end attacks even if it is accelerated with DDIM. Our DP-Attacker also provides better attack results

Speed and Effectiveness Comparison With End to End Loss We perform another set of comparison with the end to end loss to show the both the speed benefit and effectiveness of our DP-Attacker. We conduct online targeted attacks on the Transformer-based DP for the PushT task. The PGD parameters for end to end attacks at $N = 50, \sigma = 0.03, \alpha = 0.001875$. The result average model score in 50 simulations and attack time is shown in 7. The evaluation is done on a machine with RTX4080 mobile GPU, and Intel i9-13900HX CPU.

Rollouts of Quantitative Targeted Attacks The following are the rollouts of videos corresponding to Sec. 5.3. Please open this PDF with Adobe Acrobat Reader to view the animated frames.



Figure 9: $\delta = 0.05$ (PushT)



Figure 10: $\delta = 0.06$ (PushT)



Figure 11: $\delta = 0.07$ (CAN PH)

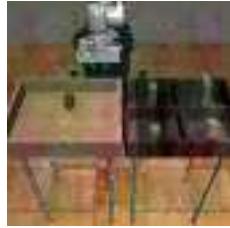


Figure 12: $\delta = 0.05$ (CAN PH)

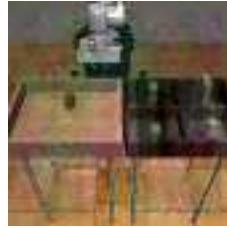


Figure 13: $\delta = 0.06$ (CAN PH)

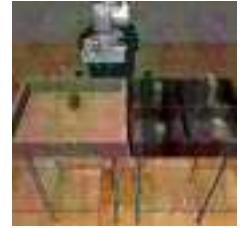


Figure 14: $\delta = 0.07$ (CAN PH)

Transferability of Offline Attacks Across Different Backbones We also evaluate the transferability of DP-Attacker generated offline attacks across different diffusion backbones. We first test the transferability of offline global attacks. We use DP-attacker to generate untargeted global offline attacks ($\delta = 0.03$) on two checkpoints (CAN-MH-CNN and CAN-MH-TF). Then, we evaluate these models using the two generated adversarial perturbations, and the success rate is listed in Table 8. We also test the transferability of patched attacks. We use DP-Attacker to generate adversarial patches on two checkpoints (CAN-PH-CNN and CAN-PH-TF). Then, we evaluate these models using the two generated adversarial perturbations, and the success rate is listed in Table 9.

Attacked Model Runner Model	CNN	TF	Original SR
CNN	0.34	0.78	0.98
TF	0.32	0.46	0.92

Table 8: Model success rate of the CAN task trained on the MH dataset. We generated offline global attacks for each backbone first, and tested them on both models. The attack transfer case is marked in blue. We also report the model’s original SR (without perturbation).

Attacked Model Runner Model	CNN	TF	Original SR
CNN	0.16	0.54	0.98
TF	0.42	0.44	0.92

Table 9: Model success rate of the CAN task trained on the PH dataset. We generated physical patched attacks for each backbone first, and tested them on both models. The attack transfer case is marked in blue. We also report the model’s original SR (without perturbation).

Note that from Table 2, the random noise patch does not affect the performance of DP on the CAN task very much. However, our DP-Attacker generated patches are able to decrease model performance in transfer settings (marked in blue) in Tables 8 and 9. This shows the effectiveness of our DP-Attacker and its potential ability in black-box attacks.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We made accurate claims in the abstract and introduction.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We stated our limitation in the last paragraph of the main paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: We don't have theoretical results in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: **[Yes]**

Justification: We show experimental details in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release the code to the public.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have the training details in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the averaged metrics on different datasets, and the difference is significant.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We put the settings in the main paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: This paper conforms with the ethics code.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: We discuss the broader impact in the main paper.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: all the models and datasets used in this paper are properly credited and licenses are respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new datasets/models are proposed in this paper.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and Research with Human Subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer:

answerNA

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.