TEST-TIME ADVERSARIAL TAKEOVER: A REAL-TIME HIJACKING INTERFACE AGAINST ROBOTIC DIFFUSION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based action generation has become a foundational component of embodied AI, but its reliance on visual conditioning leaves deployed visuomotor policies vulnerable to adversarial manipulation. Most prior attacks focus on disruption: they perturb the observation stream to reduce task success or induce erratic behavior. We study a stronger threat, Test-time Adversarial Takeover (TAKO), in which an attacker obtains a real-time steering interface over a frozen robot policy and turns it into a remotely piloted instrument. TAKO learns a small vocabulary of reusable universal patches through differentiable diffusion inference; at test time, the attacker switches among these patches in the camera stream to compose attacker-chosen trajectories. This works because the perturbation acts on the visual conditioning pathway, where the induced bias can persist through iterative generative inference. We further show that the natural targeted baseline, target-policy matching, fails because the victim policy cannot reliably supervise itself on out-of-distribution target shifts. Across four tasks (2D manipulation, simulated aerial delivery, simulated ground navigation, and physical-world ground navigation), two visual encoders (ResNet-18 and EfficientNet-B0 + Transformer), and three generative inference families (DDPM, DDIM, and flow matching), human operators achieve 100% takeover success on attacker-defined objectives in every evaluated setting.

1 INTRODUCTION

Diffusion-based action generation has become a foundational building block in embodied AI. Since the introduction of Diffusion Policy (Chi et al., 2023), diffusion models have been widely adopted as critical action generation modules across a growing range of robotic systems, including vision-language-action (VLA) models and imitation learning policies. Recently, vision has dominated many robotic policy learning tasks because it captures rich, fine-grained spatiotemporal information without explicit programming. In practice, virtually all deployed policies rely on image encoders such as ResNets, ViTs, or similar backbones to condition action generation on visual observations. Consequently, vision inputs form a natural perturbation surface, so that any policy that conditions on visual input is inherently susceptible to adversarial manipulation through that channel.

Existing adversarial attacks on robot policies have explored this vulnerability, but their threat models are limited to causing task failure or triggering task execution via language prompts in VLA systems. These attacks are undoubtedly important in safety-critical contexts, but they are quite limited in scope. Subverting a policy can cause failure, but it does not have the potential to cause targeted and sustained consequences. Language prompts can trigger unintended task execution, but they can only be applied to VLA models, cannot achieve out-of-distribution target trajectories, and can be detected by semantic checks on the language input.

A more severe threat is the potential to perform an arbitrary task that exceeds the existing learned policy. A straightforward and conventional attack formulation of this threat is to directly match the actions of a target-shifted policy that achieves the attacker's desired trajectory. We refer to this as the Target-Policy Matching (TPM) attack. However, it is hard under the TPM formulation: 1. The target policy is agnostic. The attacker must identify a target policy to steer the victim robot, which may be out-of-distribution for the victim robot. It is very likely for the victim robot to fail to generalize to the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

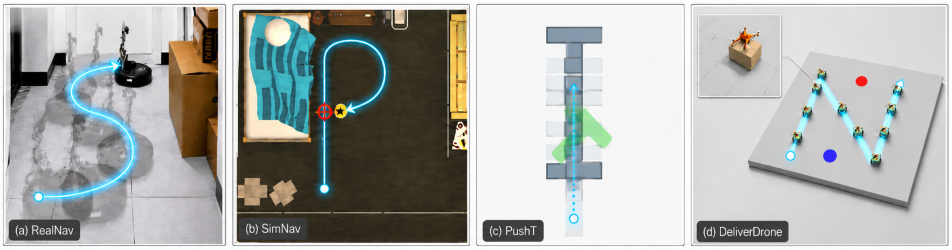

Figure 1: **Real-time adversarially steering imitation learning robots to follow arbitrary trajectories.** The tasks include: (a) RealNav, (b) SimNav, (c) PushT, (d) DeliverDrone. The trajectories are controlled by a human operator and resemble four letters: S, P, I, and N.

shifted target configuration. 2. Real-time constraint for test-time deployment. The attacker must steer the victim robot on a specific trajectory over a long time horizon. Since the attack must be effective at test time, the attacker typically cannot rely on iterative optimization or online feedback to refine it. However, relying on open-loop static attack intervention can lead to cumulative error and often fails to achieve the target goal.

We demonstrate a fundamentally different attack formulation, **T**est-time **A**dversarial Ta**k**e**O**ver (TAKO), to identify this class of threat. Rather than disrupting a deployed policy, an attacker can *take it over*. By switching among a small set of pre-optimized universal adversarial patches injected into the camera stream at test time, the attacker obtains a real-time steering interface over the policy's output. The universal patches serve as a reusable action vocabulary: each patch corresponds to an action-primitive bias, such as "move forward" or "move left", and switching among them composes a trajectory that can reach arbitrary attacker-defined targets in real time. The victim robot transforms from an autonomous agent into a remotely piloted instrument under the attacker's continuous control. The attack can be initiated at any time: once the adversarial patch is presented to the camera, the policy continues to generate smooth, executable action sequences that the attacker can control. This changes the deployment concern from accidental failure to unauthorized takeover — a critical shift from a safety problem to a security problem. As long as the attack is engaged, an independent operator can steer the robot to arbitrary attacker-defined targets in real time by switching among the universal patches, without the need to solve a complex optimization problem or generalization challenge at test time. See Figure 1 for a demonstration of such an attack.

The vulnerability stems from the architecture of diffusion policies. The adversarial perturbation acts on the conditioning signal (visual features), so the iterative diffusion denoising process cannot correct the induced bias, but is rather steered by the adversarial input. Each patch is optimized with multiple states sampled in the environment, so the resulting bias becomes a universal and reusable action-primitive that is effective across different states.

We instantiate TAKO on diffusion-based imitation learning policies across three virtual tasks and one physical task (four tasks demonstrated in Figure 1): a physical world ground navigation task (RealNav), a simulated ground navigation task (SimNav), a 2D manipulation task (PushT), and a simulated drone delivery task (DeliverDrone). In this setting, the robot's autonomous success at executing the attacker-defined target policy is nearly 0%, whereas human operators using TAKO achieve 100% success in steering the robot to the same targets through our patch-vocabulary-based interface. The conventional TPM baseline fails entirely because the victim policy cannot generalize to shifted target configurations.

Our contributions are summarized as follows:

- We introduce TAKO as a new adversarial paradigm for robot policy risks, shifting the objective from causing failure to obtaining sustained, operator-controllable authority over policy behavior (See Table 1 for the comparison with prior work).
- We show that diffusion-based visuomotor policies admit a simple steering interface: a pre-optimized vocabulary of adversarial patches that can be switched online to compose arbitrary attacker-directed trajectories.
- We demonstrate that this takeover paradigm is not task or architecture-specific, validating it across multiple tasks in virtual and physical environments, spanning distinct visual encoders

Table 1: Comparison of the relevant attacks with TAKO

Demonstrated property	Enchanting(Lin et al., 2017)	DP-Attacker(Chen et al., 2024)	Jones et al.(Jones et al., 2025)	Dirty-Road(Sato et al., 2021)	**Ours**
Physical realizability	✗	✓	✗	✓	✓
Diffusion-policy attack	✗	✓	✗	✗	✓
Target-policy achieving	✗	✗	✓	✗	✓
Real-time Intervention	✗	✗	✗	✗	✓
Composable action vocabulary	✗	✗	✗	✗	✓

(ResNet-18 and EfficientNet-B0+Transformer) and three generative inference families (DDPM, DDIM, flow matching).

2 Related Work

Diffusion models for robot action generation. Diffusion Policy (Chi et al., 2023) established diffusion-based action generation as a core paradigm for visuomotor behavior cloning. Subsequent work expanded this family to new inputs, architectures, and faster inference variants (including flow matching) (Ze et al., 2024; Ke et al., 2024; Prasad et al., 2024; Zhang et al., 2024; Wang et al., 2024c; Zhu et al., 2025; Liu et al., 2025). Related generative action heads also appear in generalist robot policies and VLAs (Octo Model Team et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; Li et al., 2024; Hou et al., 2024). Because these systems all rely on visual conditioning to drive action generation, vulnerabilities in that pathway can transfer across embodiments and inference families.

Adversarial patches in visual perception. Universal adversarial patches were introduced for image classifiers (Brown et al., 2017) and later made physically robust via EOT-style optimization (Athalye et al., 2018; Eykholt et al., 2018). Follow-up work attacked detection and dense prediction pipelines (Liu et al., 2018; Thys et al., 2019; Xu et al., 2020; Yamanaka et al., 2020; Cheng et al., 2022; Ranjan et al., 2019; Hu et al., 2022; 2023) and modern encoders (Fu et al., 2022; Zhou et al., 2023). In robotics control, prior patch attacks mostly target low-dimensional steering outputs or lane-departure-style objectives (Sato et al., 2021; Kong et al., 2020; Pavlitskaya et al., 2020).

Adversarial attacks on robot policies. Prior adversarial robotics work focuses on disruption: reducing task success or inducing failures under perturbed observations. Early results were shown in deep RL (Huang et al., 2017; Lin et al., 2017; Zhang et al., 2020), and later extended to imitation and diffusion policies (Chen et al., 2024; Kalra et al., 2025). In VLA systems, both patch attacks and jailbreak-style prompting can alter behavior (Wang et al., 2024a; Jones et al., 2025; Zhang et al., 2025; Robey et al., 2024), while backdoor approaches require training-time compromise (Kiourti et al., 2020; Wang et al., 2024b).

3 Method

3.1 Problem Formulation and Threat Model

We consider a diffusion-based visuomotor policy $\pi_{\boldsymbol{\theta}}$ that maps an observation $\mathbf{o}_t = (I_t, s_t)$, comprising an RGB image $I_t \in \mathbb{R}^{H_I \times W_I \times 3}$ and an optional proprioceptive state s_t, to an action sequence $\mathbf{a}_{t:t+H_a} \in \mathbb{R}^{H_a \times d_a}$ over an execution horizon of H_a steps. The action sequence is generated by a conditional diffusion process. Writing ϕ for the vision encoder and ϵ for the conditional denoiser (both subnetworks of $\pi_{\boldsymbol{\theta}}$), inference proceeds along a fixed chain

$$I_t \xrightarrow{\phi} z_t \oplus s_t \to c_t \xrightarrow{\epsilon,\, \tau_{1:K}} \hat{\mathbf{a}}_{t:t+H_a}, \tag{1}$$

where $z_t = \phi(I_t)$ are the visual features, c_t is the conditioning signal formed from z_t concatenated with the proprioceptive state s_t when present, and $\tau_{1:K}$ denotes a K-step deterministic sampler (DDPM, DDIM, or a forward-Euler integrator of a flow-matching velocity field). The policy parameters $\boldsymbol{\theta}$ are frozen after deployment, and the policy is operated in receding-horizon fashion: the first $H_e \leq H_a$ actions of $\hat{\mathbf{a}}_{t:t+H_a}$ are executed, then the policy is re-invoked on the next observation.

The attacker has white-box access to $\pi_{\boldsymbol{\theta}}$ for offline patch optimization, and the ability to overlay a fixed-shape perturbation $\boldsymbol{\delta} \in \mathbb{R}^{h \times w \times 3}$ at a fixed image location on each frame of the input stream

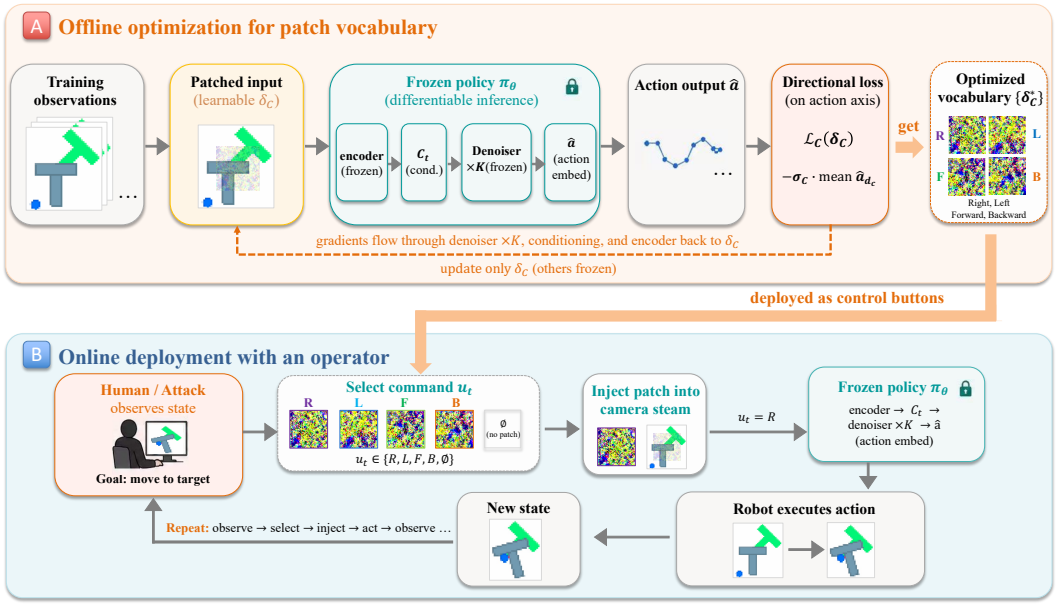

Figure 2: Attack pipeline. **Top:** offline universal-patch optimization via differentiable inference. Gradients flow from the directional loss through the frozen denoising chain and vision encoder back to the patch. **Bottom:** closed-loop online interactive deployment. The attacker selects a vocabulary entry via keyboard; the victim policy thus can be steered in real time.

at inference time. We write $\text{apply}(I_t, \boldsymbol{\delta})$ for the patched image and, by extension, $\text{apply}(\mathbf{o}_t, \boldsymbol{\delta}) = (\text{apply}(I_t, \boldsymbol{\delta}), s_t)$ for the patched observation. The perturbation is bounded under $\|\boldsymbol{\delta}\|_\infty \leq \varepsilon$. The proprioceptive state s_t is not perturbed; the model weights, training data, and task specification are not modified. Once the attack is engaged, the patch is composited into every observation that reaches the policy; once disengaged, the policy receives the clean stream again and resumes its trained behavior.

The conventional disruption threat model treats the attacker as an adversary who seeks a single perturbation $\boldsymbol{\delta}$ that drives the policy off-task. Concretely, given a task-failure functional $\mathcal{L}_{\text{fail}}$ such as a negative task reward or distance to a forbidden region, the disruption attacker solves

$$\boldsymbol{\delta}^*_{\text{disrupt}} = \arg\max_{\|\boldsymbol{\delta}\|_\infty \leq \varepsilon} \mathbb{E}_{\mathbf{o}\sim\mathcal{D}_{\text{deploy}}} \left[\mathcal{L}_{\text{fail}}\big(\pi_{\boldsymbol{\theta}}(\text{apply}(\mathbf{o}, \boldsymbol{\delta}))\big) \right]. \tag{2}$$

The output of Equation (2) is a single perturbation that provides only an open-loop departure from the trained policy.

However, a more severe threat model allows the attacker to perform an arbitrary task by designing perturbations that guide the policy towards a desired outcome. It is much more difficult than disruption, because the attacker must solve for a perturbation that not only causes failure but also induces specific, targeted behavior.

3.2 A Natural Baseline: Target Policy Matching

A straightforward approach to achieve the target policy is to directly optimize the perturbation such that the resulting action matches the desired reference policy. We refer to this approach as *target policy matching* (TPM). Concretely, let $\mathbf{o}^{\text{shift}_c}$ denote an observation in which the goal has been displaced along direction c, and let

$$\bar{\mathbf{a}}^{\,c} = \pi_{\boldsymbol{\theta}}\big(\mathbf{o}^{\text{shift}_c}\big) \tag{3}$$

be the corresponding reference policy generated by querying the frozen victim policy model. The natural baseline TPM solves

$$\boldsymbol{\delta}_c^{\text{tgt}} = \arg\min_{\|\boldsymbol{\delta}_c\|_\infty \leq \varepsilon} \mathbb{E}_{\mathbf{o}\sim\mathcal{D}} \left\| \pi_{\boldsymbol{\theta}}\big(\text{apply}(\mathbf{o}, \boldsymbol{\delta}_c)\big) - \bar{\mathbf{a}}^{\,c} \right\|_2^2. \tag{4}$$

This is the standard targeted-attack template for the targeted threat model.

However, this approach can easily fail in practice for two reasons. First, the reference policy $\bar{\mathbf{a}}^{\,c}$ is generated by querying the victim policy, which may not produce the desired behavior when the goal is shifted, especially if the shifted goal is out-of-distribution. Second, even if the reference policy is generated successfully, the optimization in Equation (4) may fail to find a perturbation that reliably induces the desired behavior across the long horizon of the action sequence, especially when the policy is operated in receding-horizon mode and the accumulated drift from the reference trajectory causes the resulting policy runs to diverge from the target behavior. Given the context of real-time deployment, an online adaptation of the reference trajectory is not feasible. In practice, we find that TPM fails to produce a usable patch for any of our testbed of four tasks.

3.3 TAKO Stage 1: Offline Optimization for Universal Patch Vocabulary

See Figure 2 for the pipeline of TAKO. TAKO replaces reference matching with a control vocabulary: each command is a universal patch that induces a consistent action effect in the policy's output. Stage 1 learns this vocabulary offline so that each patch becomes a reusable control primitive. The patches are optimized by

$$\boldsymbol{\delta}_c^* \;=\; \arg\min_{\|\boldsymbol{\delta}_c\|_\infty \le \varepsilon} \; \mathcal{L}_c(\boldsymbol{\delta}_c). \tag{5}$$

In general, $\mathcal{L}_c$ can be any loss that encourages the patch to induce a bias in the action output. We use a directional loss tailored to the tasks in our testbed, which require moving in four cardinal directions on a plane. For each command c, we fix an axis assignment $d_c \in \{1, \dots, d_a\}$ and a sign $\sigma_c \in \{+1, -1\}$ for each command c. Let $\hat{\mathbf{a}}(\mathbf{o}, \boldsymbol{\delta}) = \pi_{\boldsymbol{\theta}}(\mathrm{apply}(\mathbf{o}, \boldsymbol{\delta}))$ denote the patched action sequence and let $\hat{a}_{t',d}(\mathbf{o}, \boldsymbol{\delta})$ denote its (t', d) entry. We define

$$\mathcal{L}_c(\boldsymbol{\delta}_c) \;=\; -\,\sigma_c \cdot \mathbb{E}_{\mathbf{o}\sim\mathcal{D}} \, \frac{1}{H_a} \sum_{t'=1}^{H_a} \hat{a}_{t',d_c}(\mathbf{o}, \boldsymbol{\delta}_c), \tag{6}$$

where $\mathcal{D}$ is an observation pool sampled from the training distribution, and the expectation is estimated by mini-batches in practice. For the four-direction vocabulary, $\mathcal{C} = \{\text{forward}, \text{backward}, \text{left}, \text{right}\}$ corresponds to two axes $d_c \in \{x, y\}$ and the two signs $\sigma_c \in \{+1, -1\}$.

Three properties make Equation (6) a clean replacement for Equation (4). First, the loss is defined only on the single action-chunk output, so it never invokes the long-horizon policy on a shifted reference scene. Second, the action bias does not require a specific target, but rather only a consistent directional push, so the patch is not penalized for producing incoherent action sequences as long as the mean bias is in the right direction. Third, averaging over $\mathbf{o} \sim \mathcal{D}$ makes the patch universal rather than trajectory-specific.

To optimize Equation (6), gradients are backpropagated through the full inference chain in Equation (1) to the patch. In practice, we unroll the sampler used at deployment time: DDIM (Song et al., 2021) for DDPM-trained policies and forward-Euler integration for flow-matching policies (Lipman et al., 2023). Because the patch acts on the visual conditioning pathway, the same recipe applies across score-based and flow-matching inference families.

3.4 TAKO Stage 2: Online Compositional Deployment

Stage 2 turns that offline vocabulary into a real-time control interface. At deployment time, the attacker holds the optimized patches $\{\boldsymbol{\delta}_c^*\}_{c\in\mathcal{C}}$ and selects an element $u_t \in \mathcal{C} \cup \{\varnothing\}$ at each replanning step. The patched observation

$$\tilde{\mathbf{o}}_t \;=\; \begin{cases} \mathrm{apply}(\mathbf{o}_t, \boldsymbol{\delta}_{u_t}^*) & u_t \in \mathcal{C} \\ \mathbf{o}_t & u_t = \varnothing \end{cases} \tag{7}$$

is fed to the victim policy, which returns $\hat{\mathbf{a}}_{t:t+H_a} = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{o}}_t)$ and executes the first H_e actions before the next replanning step. The attacker can choose u_t from the realized state, but the policy itself only sees $\tilde{\mathbf{o}}_t$ and treats it as clean input. When $u_t = \varnothing$, the policy immediately returns to normal behavior on the next chunk.

This protocol has two properties that distinguish takeover from conventional attack formulation. *Composability*: the realized trajectory is determined by the command sequence $(u_1, u_2, \dots)$, so

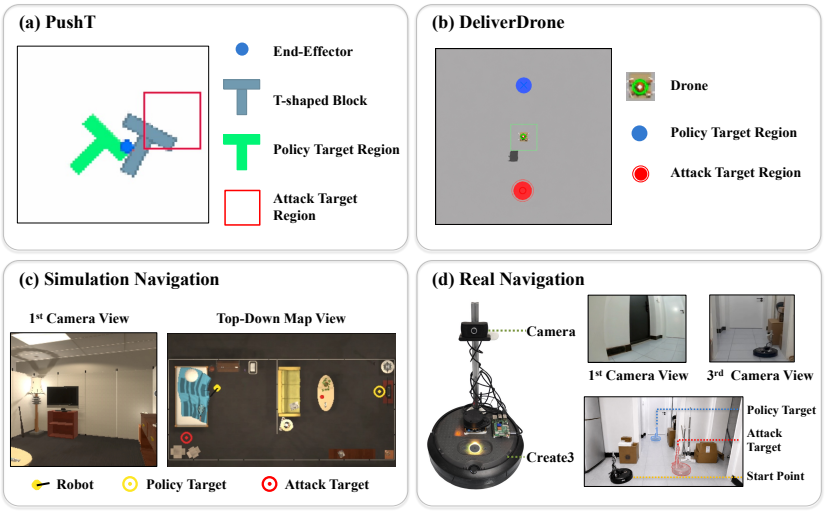

Figure 3: Illustration of the valuation tasks: (a) PushT: 2D manipulation with a T-shaped block. (b) DeliverDrone: simulated aerial delivery. (c) SimNav: AI2-THOR indoor navigation with first-person and top-down views. (d) RealNav: physical indoor navigation with an iRobot Create 3. The attack targets are only visible to human operators.

Table 2: Setup overview for the tasks

	PushT	DeliverDrone	SimNav	RealNav
Environment	Sim.	Sim. (Gazebo)	Sim. (AI2-THOR)	Physical
Embodiment	2D agent	Sim. quadrotor	FPV agent	iRobot Create 3
Observation	2×96^2 RGB + pos	4×96^2 RGB + vel + goal	2×256^2 FPV + pos	2×96^2 FPV + pos
Action space	2D pixel velocity	8 XY waypoints (m)	(v, ω) velocity	2D body-frame waypoint
Vision encoder	ResNet-18	EffNet-B0 + Trans.	ResNet-18	ResNet-18
Inference families tested	DDPM, DDIM, FM	DDPM, DDIM, FM	DDPM, DDIM, FM	DDPM, DDIM, FM
Patch size	48×48	48×48	128×128	48×48
Patch area ratio	25%	25%	25%	25%
ε	64/255	64/255	64/255	64/255

patches function as a switchable vocabulary rather than one-shot perturbations. *Closed-loop control*: the attacker chooses u_t online from the current state rather than committing to a fixed sequence of commands in advance, so the attack can correct for deviations from any single patch along the trajectory.

As long as the vocabulary is sufficiently expressive to reach the attacker's desired targets, the composability and closed-loop control of this protocol guarantee that the attacker can reach those targets regardless of how the policy drifts under each individual patch. The attacker's authority to steer online from the current state allows them to correct for any drift or misgeneralization of individual patches, so errors do not accumulate along the trajectory as they do under TPM.

4 Experiments

4.1 Experimental Setup

Evaluation tasks. We evaluate TAKO on four diverse tasks: PushT (2D manipulation), DeliverDrone (simulated aerial delivery), SimNav (simulated egocentric indoor navigation), and RealNav (physical-world ground navigation). The four tasks differ on five axes: **(1) Vision encoder**: ResNet-18 (PushT, SimNav, RealNav) versus EfficientNet-B0 + 4-layer Transformer (DeliverDrone). **(2) Generative inference family**: every task is evaluated under DDPM, DDIM, and flow matching. **(3) Embodiment**: 2D manipulation agent, simulated quadrotor, first-person-view navigation agent, and a real navigation robot(iRobot Create 3 chassis). **(4) Action parameterization**: 2D pixel velocity (PushT, position-style), metric XY waypoints (DeliverDrone, position-style), continuous velocity (v, ω) (SimNav, velocity-style), and body-frame waypoints (RealNav, position-style). The inclusion of

Table 3: Patch directional success rate/% (mean over 4 directions, with 95% Wilson confidence intervals)

	PushT	DeliverDrone	SimNav	RealNav
DDPM (100 steps)	98.5 [96.6, 99.5]	99.5 [97.6, 99.8]	65.8 [60.1, 70.8]	75.5 [70.2, 79.9]
DDIM (3–8 steps)	99.0 [97.1, 99.7]	99.5 [97.6, 99.8]	69.3 [63.9, 74.3]	74.5 [69.5, 79.3]
Flow matching	99.2 [97.6, 99.8]	100.0 [98.7, 100]	70.3 [64.9, 75.2]	97.3 [94.8, 98.6]

a velocity-action task is deliberate—it tests whether the attack transfers across position-style and velocity-style action spaces, which respond to a fixed action perturbation in qualitatively different ways. **(5) Environment**: simulation (PushT, DeliverDrone, SimNav) versus the physical world (RealNav). Figure 3 shows the task environments and Table 2 summarizes the configuration; per-task implementation details are in Appendix A1.2.

Human interactive takeover protocol. See Figures A1 and A2 for the human task interface. Four researchers act as adversarial operators. Each trial presents an environment with the policy's original target and an attacker-defined target visible only to the operator (e.g., a blue marker placed at a predetermined position different from the original target). The operator steers the robot toward the target attacker in real time by pressing keyboard keys bound to pre-optimized universal patches. The patch is applied to the robot's visual input only when the corresponding key is held. We report success rate across all operators and trials (10 trials per operator per task, 40 trials total per task).

Additional experimental setup See Appendix A1.1 for Policy implementation details, Patch vocabulary optimization details, and task implementation details.

4.2 Patch Vocabulary Effectiveness

We first quantify whether the optimized patches induce a consistent directional bias in the victim policy's action output, and whether this property transfers across architectures and inference procedures. The result certifies that all patches behave as a directional action vocabulary; the closed-loop adversarial behavior they support is then evaluated in Section 4.3.

For each (task, direction, inference family) tuple, we applied the corresponding patch to a held-out observation pool ($N = 300$ samples per task). For each sample, we performed policy inference, extracted the predicted action sequence, and computed the mean shift along the action axis aligned with the commanded direction (relative to the clean policy's output for the same observation). We report *Directional success* as the fraction of samples whose mean shift averaged over the predicted action window matches the commanded direction. Per-direction breakdowns are reported in Appendix A3.

Table 3 reports directional success across the four tasks and three inference families; each cell is averaged over the four cardinal directions. The patches induce above-chance directional bias in every cell. This shows that the patch optimization algorithm robustly yields action-primitive vocabulary across different vision encoders and generative inference families.

4.3 Interactive Takeover Results

We then evaluated whether a human operator, given the directional vocabulary as a real-time control interface, can drive the robot to attacker-defined targets that no single patch was optimized to reach. For each task, we applied a three-layer protocol: (1) Verify the victim policy is competent on its original task; (2) Deploy the TPM baseline and report its Attack Success Rate (ASR); (3) Deploy TAKO in the human-operator protocol and report the ASR. ASR is defined as the percentage of trials in which the agent successfully reaches the attacker-defined target.

Table 4 shows the results. It carries three observations: First, every victim policy was competent on its nominal task (top row). Second, the TPM baseline achieved an ASR of *zero* across all four tasks. Third, our action vocabulary, composed at deployment time by a human operator, achieved 40/40 ASR across all tasks; the robots reached the attacker's targets that had never been trained to reach. We further analyze the underlying failure modes of TPM and compare it with TAKO in Section 4.4.

Table 4: Task-level success rate. †Success defined as IoU score ≥ 0.85.

	PushT	DeliverDrone	SimNav	RealNav
Clean task performance	10/10†	10/10	20/20	20/20
TPM ASR	0/40	0/40	0/40	0/40
TAKO ASR (human)	**40/40**	**40/40**	**40/40**	**40/40**

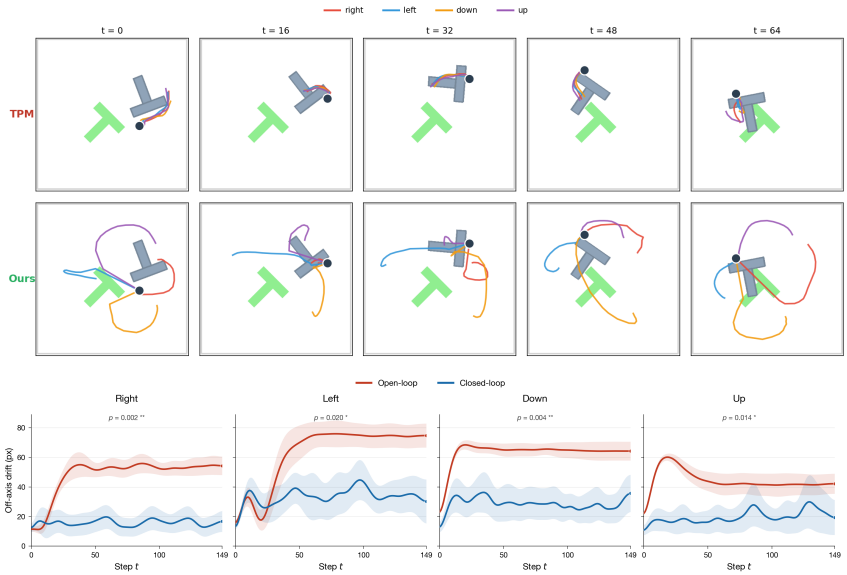

Figure 4: Visualization of the analysis. **Top:** Per-direction action predictions across five timesteps of a single rollout. TPM traces collapse into nearly identical paths (top row), whereas our traces fan out into four distinct directions (bottom row). **Bottom**: off-axis drift under open-loop vs. closed-loop attacker control with the same patch vocabulary. For each command, off-axis drift is the agent's vertical displacement from start under "right"/"left", and horizontal displacement under "up"/"down". Each curve is the mean over 10 paired DDIM-seed trials; the shaded band is 95% confidence interval.

4.4 Analysis of TPM and TAKO

The failure of TPM in fact provides experimental evidence of the two problems argued in Section 3.2: (i) OOD-supervision pathology — the reference policy obtained from the frozen policy on a goal-shifted scene does not encode the desired direction; and (ii) accumulative drift—a static reference, fit at training time, becomes stale as the robot evolves under closed-loop deployment. We investigate each problem in the PushT task separately.

We first verify that the supposed target oracle, obtained by querying the frozen victim policy model in a goal-shifted environment, cannot itself perform the new task. Running the frozen policy without any patch in environments where the green T target is moved to four directions, yields $0/40$ task success across all four directions. In fact, the four actions of the victim policy given various targets were nearly identical, indicating that the policy is goal-locked. It was trained on demonstrations to a single fixed goal and generalizes poorly to novel goals. See Figure 4 top for visualization of the actions. The TPM actions given different targets remain identical, while our patch vocabulary consistently generates distinct action sequences.

Even granting TPM a perfect direction-specific patch, the open-loop deployment of a single fixed reference $\bar{\mathbf{a}}^c$ accumulates drift along the horizon. We isolate this factor by replacing the reference oracle with our optimized patches and varying *only* the deployment loop: *Open-loop* holds the same direction patch on every step; *Closed-loop* lets a human operator toggle and switch patches in real time. Figure 4 bottom plots off-axis drift over 150 steps (10 paired DDIM-seed trials per direction). Open-loop drift rapidly increases, while closed-loop drift stays low in every direction with statistically

significant gaps (per-direction paired Wilcoxon $p < 0.02$; combined p=1.4×10^{-9}). Even with a stable action bias, open-loop alone fails to remain on-axis, while closed-loop attacker control is the structural fix.

TPM's failure is not a hyperparameter issue or a weakness of the optimization; it is a fundamental incompatibility between target-policy matching and achieving the attacker's target in a robotic context, where the latter needs a generalizable closed-loop deployment. By contrast, our objective for the universal patches does not require the victim policy to generate any out-of-distribution inputs; rather, it yields robust action-primitive biases. Moreover, the human operator plays a key role in achieving the closed-loop goal under the TAKO formulation.

4.5 Ablation Studies

We ablate the design choices on PushT by varying patch size, perturbation budget ε, the number of denoising steps used during patch optimization, patch placement, agent-position conditioning, and the expectation-over-transformation (EOT) number for the diffusion noise seed. See Appendix A2 for the detailed results of all factors along with an online human verification study.

We summarize the findings as follows. (1) Small patches can still hijack direction. A 32×32 patch (11% image area) reaches 94% directional success; even 16×16 (2.8%) retains 79%. Due to the need for accurate action, PushT requires online human task verification, which reports difficulty when the patch size is below 48×48. (2) Perturbation budget saturates above $\varepsilon = 64/255$. The directional shift grows monotonically with ε but flattens above $64/255$ (δ_{dir} rises from 146.1 px at $64/255$ to only 187.1 px at $255/255$); $\varepsilon = 64/255$ remains a balanced default for visibility versus effectiveness. (3) The denoising chain does not erase adversarial bias. Increasing DDIM steps from 3 to 20 leaves δ_{dir} essentially flat (140.4 $\rightarrow$ 151.7 px). (4) Patch effectiveness varies over positions. Placing the patches at the center, right edge, or bottom-right yields an effective directional bias, with a Directional success rate over 93%, while placing them at the top-edge and Top-left yields a weaker bias. (5) Agent-position conditioning and EOT have a negligible effect.

5 Conclusion

Prior adversarial attacks on robot policies primarily ask whether an adversary can make a robot fail. This paper identifies a more serious threat: *Test-time Adversarial Takeover* (TAKO), in which a human operator obtains a real-time steering interface over a victim diffusion-based policy by switching among pre-optimized universal patch vocabulary injected into the camera stream. We demonstrate this takeover interface across multiple evaluated settings, including 2D manipulation, aerial delivery, and ground navigation, in both the simulation and the physical world.

Conceptually, TAKO reframes adversarial control as an action vocabulary rather than a single target-matching perturbation. Instead of requiring the attacker to specify and track a full counterfactual target policy, a small set of reusable action primitives can be composed online to produce attacker-directed behavior. More broadly, these findings suggest that adversarial bias introduced through visual conditioning can persist during generative inference sufficiently to support sustained steering. We hope this work motivates future research on stronger evaluation methods, defenses, and security mechanisms for generative robot policies under interactive adversarial control.

Limitation Our experiments assume white-box access for offline patch optimization and digital injection into the camera stream, and are evaluated in controlled simulation and physical setups. Realization of the takeover without these assumptions, such as via physical patches or black-box optimization, is an important direction for future work.

AI Use Statement

We used large language models (LLMs) in four ways. (1) *Writing*: to polish the wording of the paper. (2) *Retrieval and discovery*: LLM-based search tools helped us find related work; the authors read every cited paper and checked each citation. (3) *Research execution*: LLM coding assistants helped write experiment code, figure scripts, and LaTeX formatting; the authors reviewed, tested, and ran all code, and produced all reported results. (4) *Drafting*: LLMs drafted the ethics, reproducibility, and AI use statements, which the authors revised. LLMs were not used to come up with the research

ideas, and the authors designed the method and experiments and verified all claims. The authors take full responsibility for the content of this paper.

Ethics Statement

This work exposes a security vulnerability in diffusion-based robot policies. We report it to raise awareness and to motivate defenses, following the practice of responsible disclosure in adversarial machine learning research. All experiments ran in simulation or in controlled laboratory settings on robots that the authors own; no public-facing or third-party robotic system was attacked. The human operators in the takeover trials were members of the research team, and no personal data was collected. We believe the benefits of documenting this vulnerability outweigh the risks of disclosure, since awareness is a prerequisite for effective defenses.

Reproducibility Statement

Section 4.1 and Appendix A1 specify the victim policies, the tasks, the patch optimization hyperparameters, and the human takeover protocol. Appendix A2 and Appendix A3 report the full ablations and per-direction results, and Appendix A4 lists the resources, datasets, and code release plan. Offline patch optimization and evaluation are seeded.

References

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning (ICML)*, pp. 284–293, 2018. URL https://arxiv.org/abs/1707.07397. arXiv:1707.07397.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. doi: 10.48550/arXiv.2410.24164. URL https://arxiv.org/abs/2410.24164.

Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. In *NeurIPS Workshop on Machine Learning and Computer Security*, 2017. URL https://arxiv.org/abs/1712.09665. arXiv:1712.09665.

Yipu Chen, Haotian Xue, and Yongxin Chen. Diffusion policy attacker: Crafting adversarial attacks for diffusion-based policies. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3800. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d83fd70a31c64e020844ec80705ba87f-Abstract-Conference.html. arXiv:2405.19424.

Zhiyuan Cheng, James Liang, Hongjun Choi, Guanhong Tao, Zhiwen Cao, Dongfang Liu, and Xiangyu Zhang. Physical attack on monocular depth estimation with optimal adversarial patches. In *European Conference on Computer Vision (ECCV)*, pp. 514–532, 2022. URL https://arxiv.org/abs/2207.04718. arXiv:2207.04718.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. URL https://arxiv.org/abs/2303.04137. arXiv:2303.04137.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1625–1634, 2018. URL https://arxiv.org/abs/1707.08945. arXiv:1707.08945.

Yonggan Fu, Shunyao Zhang, Shang Wu, Cheng Wan, and Yingyan Celine Lin. Patch-fool: Are vision transformers always robust against adversarial perturbations? In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2203.08392. arXiv:2203.08392.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Hengjun Pu, Chengyang Zhao, Ronglei Tong, Yu Qiao, Jifeng Dai, and Yuntao Chen. Diffusion transformer policy. *arXiv preprint arXiv:2410.15959*, 2024. doi: 10.48550/arXiv.2410.15959. URL https://arxiv.org/abs/2410.15959.

Zhanhao Hu, Siyuan Huang, Xiaopei Zhu, Fuchun Sun, Bo Zhang, and Xiaolin Hu. Adversarial texture for fooling person detectors in the physical world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13307–13316, 2022.

Zhanhao Hu, Wenda Chu, Xiaopei Zhu, Hui Zhang, Bo Zhang, and Xiaolin Hu. Physically realizable natural-looking clothing textures evade person detectors via 3d modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16975–16984, 2023.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *ICLR Workshop*, 2017. URL https://arxiv.org/abs/1702.02284. arXiv:1702.02284.

Eliot Krzysztof Jones, Alexander Robey, Andy Zou, Zachary Ravichandran, George J. Pappas, Hamed Hassani, Matt Fredrikson, and J. Zico Kolter. Adversarial attacks on robotic vision language action models. *arXiv preprint arXiv:2506.03350*, 2025. doi: 10.48550/arXiv.2506.03350. URL https://arxiv.org/abs/2506.03350.

Akansha Kalra, Basavasagar Patil, Guanhong Tao, and Daniel S. Brown. How vulnerable is my learned policy? universal adversarial perturbation attacks on modern behavior cloning policies. *arXiv preprint arXiv:2502.03698*, 2025. doi: 10.48550/arXiv.2502.03698. URL https://arxiv.org/abs/2502.03698.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3D diffuser actor: Policy diffusion with 3D scene representations. In *Conference on Robot Learning (CoRL)*, 2024. URL https://arxiv.org/abs/2402.10885. arXiv:2402.10885.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980. arXiv:1412.6980.

Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. TrojDRL: Trojan attacks on deep reinforcement learning agents. In *Design Automation Conference (DAC)*, 2020. URL https://arxiv.org/abs/1903.06638. arXiv:1903.06638.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017. doi: 10.48550/arXiv.1712.05474. URL https://arxiv.org/abs/1712.05474.

Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. PhysGAN: Generating physical-world-resilient adversarial examples for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14254–14263, 2020. URL https://arxiv.org/abs/1907.04449. arXiv:1907.04449.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024. doi: 10.48550/arXiv.2411.19650. URL https://arxiv.org/abs/2411.19650.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3756–3762, 2017. doi: 10.24963/ijcai.2017/525. URL https://doi.org/10.24963/ijcai.2017/525.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t. arXiv:2210.02747.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.07864. arXiv:2410.07864.

Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. DPatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018. doi: 10.48550/arXiv.1806.02299. URL https://arxiv.org/abs/1806.02299.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2405.12213. arXiv:2405.12213.

Svetlana Pavlitskaya, Sefa Ünver, and J. Marius Zöllner. Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1–8, 2020. doi: 10.1109/ITSC45102.2020.9294426. URL https://doi.org/10.1109/ITSC45102.2020.9294426.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2405.07503. arXiv:2405.07503.

Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J. Black. Attacking optical flow. In *International Conference on Computer Vision (ICCV)*, pp. 2404–2413, 2019. URL https://arxiv.org/abs/1910.10053. arXiv:1910.10053.

Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. Jailbreaking LLM-controlled robots. *arXiv preprint arXiv:2410.13691*, 2024. doi: 10.48550/arXiv.2410.13691. URL https://arxiv.org/abs/2410.13691.

Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty road can attack: Security of deep learning based automated lane centering under physical-world attack. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 3309–3326, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/sato.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. In *Conference on Robot Learning (CoRL)*, 2023. URL https://arxiv.org/abs/2306.14846. arXiv:2306.14846.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.02502. arXiv:2010.02502.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. NoMaD: Goal masked diffusion policies for navigation and exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. URL https://arxiv.org/abs/2310.07896. arXiv:2310.07896.

Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *CVPR Workshop on The Bright and Dark Sides of Computer Vision*, 2019. URL https://arxiv.org/abs/1904.08653. arXiv:1904.08653.

Taowen Wang, Cheng Han, James Chenhao Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. *arXiv preprint arXiv:2411.13587*, 2024a. doi: 10.48550/arXiv.2411.13587. URL https://arxiv.org/abs/2411.13587. ICCV camera ready.

Xianlong Wang, Hewen Pan, Hangtao Zhang, Minghui Li, Shengshan Hu, Ziqi Zhou, Lulu Xue, Peijin Guo, Aishan Liu, Leo Yu Zhang, and Xiaohua Jia. Robot collapse: Supply chain backdoor attacks against VLM-based robotic manipulation. *arXiv preprint arXiv:2411.11683*, 2024b. doi: 10.48550/arXiv.2411.11683. URL https://arxiv.org/abs/2411.11683. Introduces the TrojanRobot framework.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024c. doi: 10.48550/arXiv.2410.21257. URL https://arxiv.org/abs/2410.21257.

Tom Ward and Jay Lemmon. dm_env_rpc: A networking protocol for agent-environment communication, 2019. URL http://github.com/deepmind/dm_env_rpc.

Tom Ward, Andrew Bolt, Nik Hemmings, Simon Carter, Manuel Sanchez, Ricardo Barreira, Seb Noury, Keith Anderson, Jay Lemmon, Jonathan Coe, et al. Using Unity to help solve intelligence. *arXiv preprint arXiv:2011.09294*, 2020. URL https://arxiv.org/abs/2011.09294.

Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial T-shirt! evading person detectors in a physical world. In *European Conference on Computer Vision (ECCV)*, pp. 665–681, 2020. URL https://arxiv.org/abs/1910.11099. arXiv:1910.11099.

Koichiro Yamanaka, Ryutaroh Matsumoto, Keita Takahashi, and Toshiaki Fujii. Adversarial patch attacks on monocular depth estimation networks. *IEEE Access*, 8:179094–179104, 2020. doi: 10.1109/ACCESS.2020.3027791. URL https://arxiv.org/abs/2010.03072. arXiv:2010.03072.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2403.03954. arXiv:2403.03954.

Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Changgan Yin, Minghui Li, Lulu Xue, Yichen Wang, Shengshan Hu, Aishan Liu, Peijin Guo, and Leo Yu Zhang. BadRobot: Jailbreaking embodied LLMs in the physical world. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2407.20242. arXiv:2407.20242.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21024–21037, 2020. URL https://arxiv.org/abs/2003.08938. arXiv:2003.08938.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3D flow-based policy via consistency flow matching for robot manipulation. *arXiv preprint arXiv:2412.04987*, 2024. doi: 10.48550/arXiv.2412.04987. URL https://arxiv.org/abs/2412.04987.

Ziqi Zhou, Shengshan Hu, Minghui Li, Hangtao Zhang, Yechao Zhang, and Hai Jin. AdvCLIP: Downstream-agnostic adversarial examples in multimodal contrastive learning. In *ACM Multimedia*, pp. 6311–6320, 2023. doi: 10.1145/3581783.3612454. URL https://doi.org/10.1145/3581783.3612454. arXiv:2308.07026.

Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Scaling diffusion policy in transformer to 1 billion parameters for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. URL https://arxiv.org/abs/2409.14411. arXiv:2409.14411.

This appendix is organized into four sections. Appendix A1 supplements the experimental setup of Section 4.1 with policy implementation details, patch optimization specifics, and per-task configurations. Appendix A2 provides the full data behind the ablation findings of Section 4.5. Appendix A3 expands the patch directional success rates of Section 4.2 to per-direction granularity. Appendix A4 reports compute resources, dataset licenses, and reproducibility information.

A1 Additional Experimental Details

A1.1 Additional Experimental Setup

Policy implementation details. All four tasks share the same underlying recipe of Chi et al. (2023)—a conditional diffusion action head conditioned on visual features—but they differ in their visual encoder. PushT, SimNav, and RealNav use a ResNet-18 with GroupNorm followed by a conditional 1D UNet action head. DeliverDrone uses the NoMaD architecture (Sridhar et al., 2024), whose visual encoder is an EfficientNet-B0 followed by a 4-layer Transformer (Shah et al., 2023). We load the publicly released NoMaD pretrained checkpoint and finetune on our task-specific data. We do not use NoMaD's goal-image mode, so random goal masking is also disabled during training; the policy input is image and state only, identical to vanilla Diffusion Policy. For *every* task, we train policy checkpoints under all three generative paradigms—DDPM, DDIM, and flow matching (Lipman et al., 2023).

Patch vocabulary optimization. For each task, we optimized four patches $\{\boldsymbol{\delta}_c^*\}_{c\in\mathcal{C}}$ corresponding to the four directional commands in $\mathcal{C}$. Each patch is a square of side length equal to half the input image width, so that the patch covers 25% of the image area in every task. The perturbation magnitude is bounded by $\varepsilon = 64/255$ in L_∞ norm, which produces visible but not overwhelming artifacts. All patches are optimized using Adam (Kingma & Ba, 2015) with learning rate 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, for 500 gradient steps with batch size 8. The observation pool $\mathcal{D}$ consists of 1000 randomly sampled frames from the training demonstrations. Task-specific deviations are noted in the per-task descriptions below. Optimization takes approximately 10 minutes per universal patch on a single NVIDIA A40 GPU.

A1.2 Task implementation Details

PushT (2D manipulation): A circular agent operates in a 512×512 pixel workspace and must push a T-shaped block to a fixed target pose. The policy observes two consecutive 96×96 RGB frames and the agent's 2D position, and predicts 16 steps of 2D pixel-space velocity, of which 8 are executed per planning cycle. This task serves as the primary concept validation environment.

PushT policy. We train the Diffusion Policy of Chi et al. (2023) for 300 epochs on the standard PushT demonstration dataset (200 episodes). The vision encoder is ResNet-18 with GroupNorm (replacing BatchNorm for small-batch inference stability). The denoiser is a conditional 1D UNet with 256 base channels and attention at 8× downsampling. DDPM training uses 100 diffusion steps; DDIM inference uses 10 steps by default. The observation horizon is 2 frames; the prediction horizon is 16 steps; the action execution horizon is 8 steps.

DeliverDrone (simulated aerial delivery): A simulated quadrotor navigates a ∼10×10 m outdoor terrain in Gazebo Harmonic to deliver a package to a blue landing pad at (3.0, 0.0) m. The drone observes four stacked 96×96 top-down RGB frames, a 3-dimensional ego-velocity state, and a pre-captured goal image of the landing area. The policy outputs 8 XY waypoints in metric coordinates. This task instantiates a realistic intelligent logistics scenario where camera-stream compromise could enable adversarial rerouting of autonomous deliveries.

DeliverDrone policy. We follow the NoMaD (Sridhar et al., 2024) codebase. The vision encoder is EfficientNet-B0 pretrained on ImageNet, followed by a 4-layer Transformer with 256-dimensional embeddings and 4 attention heads. The diffusion action head uses a 3-layer MLP denoiser with 256 hidden units. We initialize from the publicly released NoMaD checkpoint and finetune for 50 epochs on 100 task-specific demonstration trajectories collected by the authors; we do not redo the original NoMaD pretraining. The observation horizon is 4 frames; the prediction horizon is 8 waypoints.

SimNav (simulated ground navigation): An FPS agent navigates a bedroom/living-room environment in AI2-THOR (Kolve et al., 2017) (Unity 2021.3) to reach a goal position specified by XZ coordinates. The policy observes two consecutive 256×256 first-person-view (FPV) RGB frames and the agent's 2D position, and predicts 16 steps of continuous velocity (v, ω) (linear and angular), of which 8 are executed per planning cycle. The environment communicates via a dm_env_rpc (Ward & Lemmon, 2019; Ward et al., 2020) gRPC bridge on the Unity side, enabling real-time closed-loop control. This task tests the attack under egocentric visual observations with high-resolution input and continuous velocity control—a distinct perceptual and actuation regime from the allocentric top-down views of PushT and DeliverDrone.

SimNav policy. SimNav uses the same architecture as PushT—ResNet-18 vision encoder (GroupNorm) with a conditional 1D UNet—but trained via flow matching rather than DDPM. The policy maps 2×256×256 FPV RGB frames and a 2D agent position to 16 steps of continuous velocity (v, ω), executing 8 per cycle. Training data consists of human mouse demonstrations collected in AI2-THOR via a custom dm_env_rpc bridge. We merge the original forward-goal collection with a reverse-goal collection, yielding a zarr replay buffer with 152 episodes and 13,518 frames; the merged policy is trained for 100 epochs and reaches final training loss 0.13. Policy evaluation uses an 8-step forward-Euler ODE solve. Patch optimization uses lr=1.0 (raw $[0, 255]$ pixel input), 128×128 center patch (25% of 256×256 image), raw-pixel $\varepsilon = 64$, 300 steps with batch size 8 drawn from a pool of 300 observations. Patches are optimized with 3-step flow inference and evaluated with 8-step inference.

RealNav (physical ground navigation): An iRobot Create 3 chassis navigates an indoor corridor using only a forward-facing RGB camera and onboard odometry as policy inputs. The deployed policy uses the same image-based Diffusion Policy family as PushT—a ResNet-18 vision encoder with GroupNorm and a conditional 1D UNet action head—but the final online model is trained with flow matching. The policy observes two consecutive 96×96 FPV RGB frames and the robot pose, and predicts 16 steps of 2D body-frame waypoints, of which 8 are executed per planning cycle. This task validates the attack on a physical platform where sensor latency, lighting variation, wireless ROS 2 communication, and real-world robot dynamics are present.

RealNav policy. The RealNav policy is trained from 30 human-teleoperated corridor trajectories collected on the physical robot. The raw bags contain 23,376 camera images and 764 s of driving, covering 97.8 m of total path length. We resample the data to 4 Hz and obtain 3,031 processed frames, split into 27 training trajectories and 3 held-out test trajectories. Input images are stored as 96×96 RGB frames; during training the encoder applies a 76×76 random crop. The action dimension is 2, representing a body-frame waypoint $(\Delta x, \Delta y)$. The observation horizon is 2 frames, the prediction horizon is 16 waypoints, and the execution horizon is 8 waypoints.

The final online RealNav policy is a flow-matching variant of Diffusion Policy with a ResNet-18 GroupNorm visual encoder and a conditional 1D UNet action model. It is trained for 400 epochs with AdamW (lr $= 10^{-4}$, batch size 8, cosine schedule, 50 warmup steps). Flow-matching inference uses 4 Euler steps. In deployment, the policy runs at 8 Hz on the workstation, selects a body-frame waypoint from the predicted sequence, and converts it to differential-drive velocity commands through a waypoint controller. The robot-side Raspberry Pi publishes camera and RPLIDAR streams; the RPLIDAR is used only by an emergency-stop node and is not consumed by the policy.

Patch optimization uses Adam with lr=0.01 for 500 steps on normalized $[0, 1]$ RGB input, a 48×48 center patch (25% of the 96×96 policy input), $\varepsilon = 64/255$, batch size 8, and an observation pool sampled from the RealNav corridor training set.

RealNav Physical platform. RealNav runs on an iRobot Create 3 chassis with a Raspberry Pi 4B companion computer (Ubuntu 22.04 LTS, ROS 2 Humble). The Pi carries a single USB-attached 170-degree wide-angle camera that publishes 640×480 YUYV frames at 30 Hz on `/camera/image_raw/compressed`; this is the only sensor consumed by the policy. The camera is mounted on an aluminum mast approximately 42 cm above the robot base and 15 cm behind the robot center, and its intrinsics are calibrated with an equidistant fisheye model. The Create 3 contributes its on-board odometry and bumper/cliff switches; an RPLIDAR is also present but is wired only to an emergency-stop node for obstacle safety and is not provided to the policy.

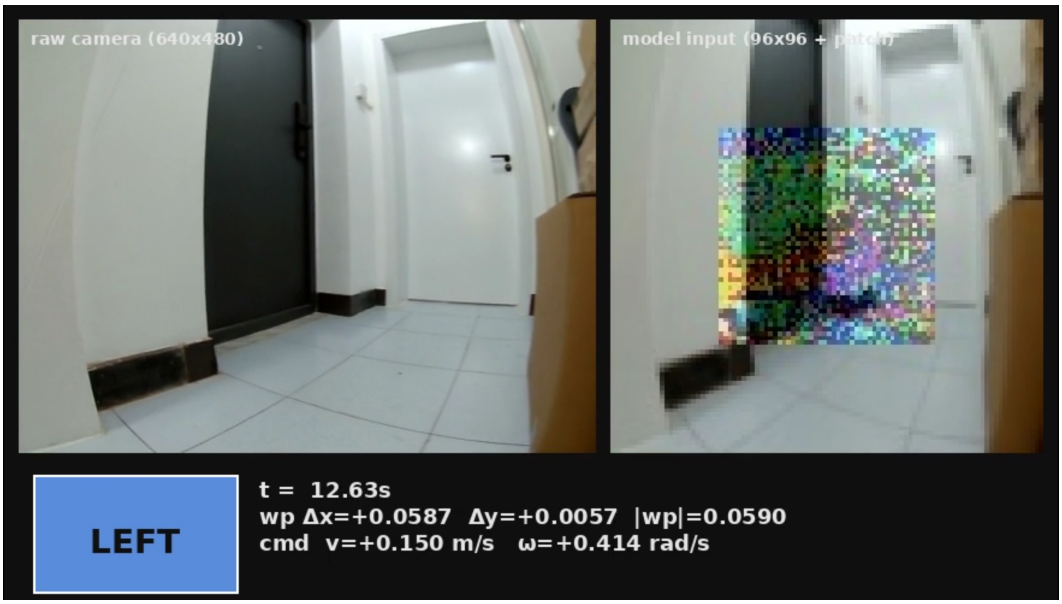

Figure A1: RealNav operation interface. Left: raw camera view. Right: model input with the patch. The "Left" label indicates the current attack direction, and the side panel shows the robot's real-time status.

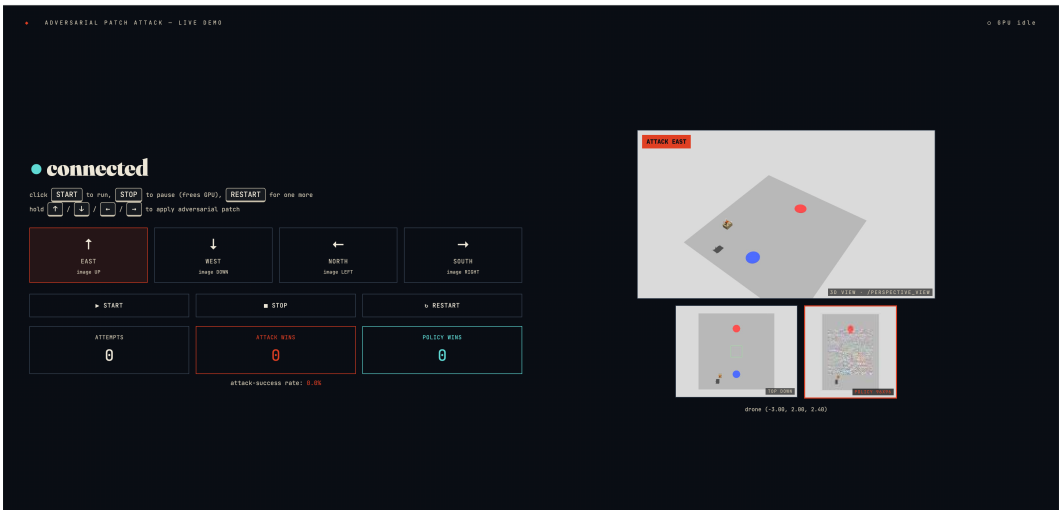

Figure A2: Drone operation interface. Left: operator instructions. Right: raw camera view with or without the patch. The same interface is used for the SimNav and PushT experiments.

Human-in-the-loop Human Operation Interface We built human-in-the-loop operation interfaces for all four experiments, through which an operator can control the victim policy by switching directional adversarial patches in real time. PushT uses a FastAPI + WebSocket backend that streams the simulation view and control to a web frontend directly. DeliverDrone bridges ROS 2 topics to the web frontend via rosbridge_server. SimNav communicates with Unity through the dm_env_rpc gRPC bridge, with a Python backend forwarding frames to the web frontend. In RealNav, the interface operates on the live camera stream (Figure A1). The web-based interfaces for the three simulated tasks are shown in Figure A2.

Table A1: Ablation summary on PushT. *(a)* Patch size sweep at $\varepsilon = 64/255$, center placement: even an 11%-area patch hijacks direction at 94%. *(b)* Perturbation budget at 48×48, center placement: directional shift saturates above $\varepsilon = 64/255$. *(c)* DDIM denoising steps used during patch optimization, at 48×48 and $\varepsilon = 64/255$: directional shift is essentially flat from 3 steps onward, indicating that the denoising chain does not attenuate adversarial bias in the conditioning signal. *(d)* Patch placement at 32×32, target direction = right: center is the direction-agnostic optimum. Default configurations are shown in **bold**.

(a) Patch size

Patch size	Dir. rate (%)	δ_{dir} (px)	$\|\Delta\|_2$ (px)
16×16	79	37.1	61.4
32×32	94	97.8	128.3
48×48	**98**	**146.1**	**172.5**
64×64	99	213.7	241.2
96×96	100	294.5	312.8

(b) Perturbation budget ε

ε	Dir. rate (%)	δ_{dir} (px)	$\|\Delta\|_2$ (px)
$16/255$	71	13.4	28.9
$32/255$	89	62.3	85.7
$64/255$	**98**	**146.1**	**172.5**
$128/255$	99	178.4	203.1
$255/255$	100	187.1	211.6

(c) Denoising steps

DDIM steps	Dir. rate (%)	δ_{dir} (px)	$\|\Delta\|_2$ (px)
1	78	52.3	89.1
3	97	140.4	168.2
5	98	143.8	170.9
10	**98**	**146.1**	**172.5**
20	98	151.7	178.3

(d) Patch placement at 32×32

Placement	Dir. rate (%)	δ_{dir} (px)	$\|\Delta\|_2$ (px)
Center	**93**	**96.6**	**117.4**
Right-edge	100	119.7	123.7
Bottom-right	100	87.7	107.0
Top-edge	77	29.1	57.2
Top-left	89	6.4	8.4

A2 Additional Ablation Study

This section provides the full data behind the ablation findings of Section 4.5. Table A1 reports the four main sensitivity factors (patch size, perturbation budget, denoising steps, patch placement). Table A2 and Table A3 report the agent-position conditioning and EOT ablations, respectively. Table A4 pairs offline metrics with online human operation outcomes to verify that offline metrics predict online controllability. All experiments are on PushT, varying one factor at a time with defaults held at patch size 48×48, $\varepsilon = 64/255$, DDIM 10 steps, evaluated on 100 held-out observations unless otherwise stated. We report three metrics: δ_{dir} (mean action shift along the target direction, in pixels), directional success rate (fraction of samples shifted in the correct direction), and $\|\Delta\|_2$ (total L_2 action displacement).

Agent-position conditioning. The PushT policy reads a 2D agent position alongside the visual observation. We test whether patch effectiveness relies on a specific position input by replacing the true position with a fixed value (center / corner) or with a uniformly random value drawn at each call. Table A2 reports a $\leq 6\%$ change in δ_{dir} across all three settings, confirming that the visual perturbation dominates over the proprioceptive anchor: the attack does not need any specific position input to succeed.

Table A2: Agent-position conditioning ablation on PushT (fixed patch size 48×48, $\varepsilon = 64/255$, DDIM 10 steps).

Position conditioning	δ_{dir} (px)	Dir. rate (%)	$\|\Delta\|_2$ (px)
Fixed (center)	146.1	98	172.5
Fixed (corner)	145.4	97	170.8
Random (uniform)	154.3	98	181.2

Expectation-over-transformation (EOT). We increase the EOT seed batch from 1 to 8 during patch optimization. Table A3 shows no measurable improvement, consistent with the near-deterministic DDIM sampling ($\eta = 0$) used throughout.

Table A3: EOT ablation on PushT (fixed patch size 48×48, $\varepsilon = 64/255$, DDIM 10 steps).

EOT seeds	δ_{dir} (px)	Dir. rate (%)	$\|\Delta\|_2$ (px)
1 (fixed)	141.0	98	167.3
4	140.2	97	166.8
8	139.4	98	165.9

Online human-in-the-loop verification. The takeover claim is fundamentally about whether a human operator can drive the policy in real time. We therefore validate that the offline metrics (δ_{dir} and directional success rate) predict actual online controllability. We train patches at four sizes (16×16 to 48×48, $\varepsilon = 64/255$, DDIM 3 steps) and test them in the interactive PushT demo, where the operator uses arrow keys to switch attack direction in real time. Table A4 reports two binary online outcomes: *direction hijack* (operator can steer the agent in the commanded direction) and *task completion* (operator can push the T-block to a chosen target region).

Table A4: Online human-in-the-loop verification of offline metrics on PushT. The composite dir. rate$\times$ δ_{dir} predicts online controllability better than either factor alone.

Patch size	Area	δ_{dir}	Dir. rate (%)	Composite	Dir. hijack	Task compl.
16×16	2.8%	37	79	29	✗	—
32×32	11.1%	93	94	87	✓	Difficult
48×48	25.0%	146	98	143	✓	✓

Two thresholds emerge. *Direction hijack* requires dir. rate $\times\, \delta_{\text{dir}} \gtrsim 50$–$87$, met by 32×32 patches (11% image area). *Task completion* requires dir. rate $\times\, \delta_{\text{dir}} \gtrsim 100$, met by 48×48 patches (25% area). Directional success rate alone is insufficient: a 16×16 patch reaches 79% directional success rate but its $\delta_{\text{dir}} = 37$ px is too small relative to the 512-px action space to be operator-controllable. This validates the choice of 48×48 as the main-paper default and connects the offline ablations of Section 4.5 to the takeover claim of Section 4.3.

A3 Additional Experimental Results

This section expands the headline numbers of Section 4.2 to per-direction granularity, supporting the cross-task transferability claim under each task's native action representation.

Per-direction transferability across tasks and inference families. Table A5 expands Table 3 to the per-direction granularity. Each value is the fraction of evaluation samples whose predicted action chunk shifts along the commanded direction under that task's native action representation. The relevant comparison is row-to-row within each task block; absolute success rates are not directly comparable across tasks because the action spaces differ (Cartesian/waypoint versus velocity (v, ω)). Across every task and direction the optimized vocabulary preserves directional controllability under all three inference families, supporting the main-paper claim that the vulnerability resides in the visual conditioning pathway rather than in a specific denoiser.

A4 Reproducibility, Resources, and Datasets

This section covers compute, datasets and their licenses, code release, and the broader-impact disclosure for the experiments and the attack pipeline.

Compute resources. The pipeline has two distinct compute axes, both modest. *Policy training and patch optimization* are run on a server-side NVIDIA A40 GPU. Policy training reuses the recipes of Chi et al. (2023) (PushT, SimNav, RealNav) and Sridhar et al. (2024) (DeliverDrone) and scales with dataset size; no single policy in our experiments exceeds 4 GPU-hours of training. Patch optimization produces one 48×48 universal directional patch in roughly 10 minutes (500 Adam steps, batch size

Table A5: Per-direction directional success rate (%) across tasks and inference families. SimNav uses velocity-axis success over (v, ω); other tasks use Cartesian/waypoint action axes. Comparisons are within each task block.

Task / Inference	Right	Left	Forward	Backward
Cartesian/waypoint action outputs				
PushT				
DDIM (3 steps)	97.7	100	98.4	98.4
DDPM (100 steps)	97.7	100	99.2	97.7
Flow matching (Euler-10)	98.4	100	99.2	97.7
DeliverDrone				
DDIM (10 steps)	100	100	100	100
DDPM (10 steps)	100	98.0	100	100
Flow matching	100	100	100	100
RealNav				
Flow matching (Euler-4)	99.0	92.0	98.0	100
DDIM (3 steps)	71.0	79.0	67.0	81.0
DDPM (100 steps)	77.0	72.0	66.0	87.0
Velocity-control action output				
SimNav (Right=turn_right, Left=turn_left)				
Flow matching (8-step eval)	83.6	75.8	70.3	74.2
DDIM (3-step patches)	85.7	69.3	63.3	83.0

Values are directional success rates (%) under each task's native action parameterization.

8, DDIM 10 unrolled denoising steps), so the full four-direction vocabulary per task takes well under one GPU-hour. The attack does not update policy weights, so policy-training cost is amortized across all reported takeover and TPM-baseline runs. *Real-time inference for the RealNav physical-robot deployment* is run on a separate mobile graphics workstation with an NVIDIA RTX 4090 GPU, which the policy uses at 8 Hz over the wireless link to the Raspberry Pi described in Appendix A1. Aggregating across all four tasks and three inference families, the reported experiments consume roughly 60 A40-hours plus about 10 RTX 4090-hours; including preliminary and discarded runs, total project compute is approximately three times this figure.

Datasets and licenses. We list each external asset together with its license and source. PushT demonstrations are the public 200-episode dataset released with the Diffusion Policy code base (Chi et al., 2023) (MIT License). SimNav uses AI2-THOR scenes (Kolve et al., 2017) (Apache 2.0); training trajectories are collected by the authors via mouse teleoperation in the `FloorPlan_Train1_1` bedroom/living-room scene. DeliverDrone uses Gazebo Harmonic (Apache 2.0) and the publicly released NoMaD pretrained checkpoint (Sridhar et al., 2024) (MIT License); the finetuning data are author-collected demonstration trajectories. We use only the released checkpoint and do not redistribute or retrain on its underlying corpus, so any restrictions on those datasets do not propagate to our use. RealNav corridor demonstrations are collected by the authors with an iRobot Create 3 in an indoor corridor; no human subjects beyond the authors are involved. The robot stack uses the iRobot Create 3 ROS 2 SDK (BSD-3-Clause) and ROS 2 Humble (Apache 2.0). The training and patch-optimization code is built on PyTorch (BSD-3-Clause) and the Diffusion Policy reference implementation (Chi et al., 2023) (MIT License). All cited assets are used in accordance with the terms of their respective licenses.

Code, patches, and reproducibility. We will release the patch optimization code, the four directional patches per task, the evaluation observation pools backing Table 4 and Table A5, and the attack/evaluation scripts on a public GitHub repository upon acceptance. The release is intentionally deferred past the review window to preserve author anonymity; we therefore do not bundle the code into the supplementary material. The takeover-trial protocol (10 trials per operator per task, 4 operators, blue-marker target) is fully specified in Section 4.1; online human trials are inherently non-deterministic, but offline patch optimization and offline evaluation are seeded for reproducibility.

Broader impacts and responsible disclosure. This paper documents an attack that converts deployed diffusion-based robot policies into operator-steerable instruments via test-time visual perturbation. The threat is structural to the visual conditioning pathway and is not specific to any single deployed system. We disclose it now—rather than withholding it—because awareness is a prerequisite for defenses, and because the necessary attack components (universal adversarial patches, differentiable diffusion inference, white-box policy access) are individually present in the public literature and can be assembled by any sufficiently motivated party. We deliberately limit our experiments to controlled simulation and authors' own physical laboratory; we do not attack any third-party deployed robotic system. Plausible defensive directions include input-space anomaly detection on the camera stream, certified robustness on the visual encoder, and policy-architecture changes that diversify or randomize the visual conditioning pathway during inference. We hope the framing of takeover-vs-disruption developed here helps future work measure robotic policy robustness against the right adversary model.