Trajectory-Level Redirection Attacks on Vision-Language-Action Models

Anonymous Author(s)
Affiliation
Address
email

Abstract: Vision-language-action (VLA) policies bring natural language into closed-loop robot control, enabling robots to execute manipulation tasks directly from text instructions. The same interface gives text a recurring role in control because the prompt is reused at every replanning step, and each prompt-conditioned action changes the future observations on which the policy acts. Existing VLA attacks study adversarial prompts that elicit targeted low-level actions or make such actions persist across changing images. We identify a stronger trajectory-level failure mode: a prompt that still *appears* to specify the intended task but redirects the final physical outcome. We mathematically formalize this setting as *command-preserving trajectory redirection*, a prompt-only threat model in which the attacker chooses one prompt before the episode, all policy and environment components remain fixed, and the prompt must stay close to the benign instruction while omitting target words and correction language. To find such prompts, we introduce an on-policy prompt search method that uses rollouts to discover perturbations whose closed-loop behavior tracks a target task while satisfying the command-preserving constraints. Experiments in simulation and on hardware show that near-benign prompt perturbations can redirect VLA rollouts to attacker-specified targets. These results expose a trajectory-level vulnerability in VLA instruction grounding: text that appears to preserve the intended command can still give an adversary control over the robot's final physical outcome.
Project website: https://vla-redirection-attack.github.io/

Keywords: Vision-Language-Action Models, Adversarial Attacks, Robot Safety

1 Introduction

Vision-language-action (VLA) policies are making natural language a persistent interface for closed-loop robot control. Models such as $\pi_{0.5}$ [1], OpenVLA [2], and RT 2 [3] map a natural-language instruction and a camera observation directly to robot actions. This interface is powerful: it enables non-expert users to specify manipulation tasks in natural language and allows a single policy to generalize across diverse scenes and objects. At the same time, it changes the role of language in the control loop. The instruction is not consumed once; it is reused at every replanning step, where it interacts with the current observation and influences the next action. As VLA-controlled robots move from constrained demonstrations toward broader deployment [1, 4], understanding the failure modes introduced by this persistent language interface becomes a concrete safety problem.

Recent work has begun to reveal these vulnerabilities. Jones et al. [5] showed that adversarial textual suffixes, optimized with the Greedy Coordinate Gradient algorithm [6], can drive OpenVLA to targeted low-level actions and sustain them across many rollout steps, and VLA-Fool [7] broadened the attack surface to visual noise, adversarial patches, and cross-modal misalignment. These studies establish that VLA policies are vulnerable to input perturbations at the level of individual queries.

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

However, eliciting a targeted action at one inference step, or making that action persist across changing images, is not equivalent to controlling what the robot *physically accomplishes*. A manipulation task such as "put the bowl on the plate" requires a temporally structured sequence of distinct behaviors: reaching, grasping, lifting, transporting, aligning, and releasing, while the observations encountered later in the rollout are themselves determined by the prompt-conditioned actions taken earlier. Thus, a prompt optimized on a fixed set of pre-collected observations is evaluated on the wrong state distribution: the relevant observations are precisely those induced by the candidate prompt in closed loop. This motivates our central question: *can a command-preserving text perturbation redirect the full closed-loop trajectory of a frozen VLA toward an adversary specified, alternate physical target?* By *command-preserving*, we require that the prompt remain textually close to the benign instruction, omit the attacker's target task, and contain no override or correction language, thereby ruling out trivial prompt replacement. To find such perturbations, we introduce an on-policy prompt search algorithm that rolls out candidate prompts, relabels their visited observations with actions from the same frozen VLA queried under the target instruction, and re-optimizes under the command-preserving constraints. This procedure is the prompt-search analogue of DAgger [8]: it aggregates data from the state distribution induced by the current candidate rather than from an offline trajectory that becomes stale once the prompt changes the robot's behavior.

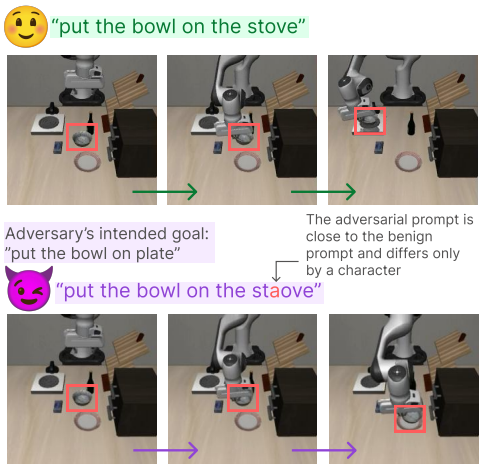


Figure 1: A *command-preserving prompt perturbation* redirects the closed-loop VLA trajectory toward an adversary-specified physical goal. Although the adversarial prompt differs from the benign instruction by only a small text change and contains no explicit target command, the robot places the bowl on the plate rather than the stove.

Our contributions: (i) We mathematically formalize *command-preserving trajectory redirection*: a prompt-only threat model in which the attacker changes a small part of the instruction, keeps the prompt seemingly aligned with the benign command, and redirects the robot to an adversary-specified alternate physical target. (ii) We introduce an on-policy prompt search method to find such trajectory redirection prompts. (iii) In both simulation and hardware experiments, we evaluate a broad set of policy architectures spanning discrete-token action prediction, flow-matching and diffusion action heads, continuous action chunks and action-as-text designs. We show that command-preserving perturbations expose a broad vulnerability across VLA families, achieving attack success rates above 90% on seven of the nine evaluated VLA architectures. (iv) Finally, we evaluate defenses to such trajectory-level attacks, showing that robust mitigation requires command-level normalization rather than only surface cleanup.

2 Related Work

Foundation Models for Robotics: Vision Language Action Models (VLAs) unify visual perception, language grounding, and low-level motor control in a single prompt-conditioned policy. Recent systems span autoregressive action-token prediction [9, 3, 10, 11, 2, 12], discretized or vector-quantized action tokenizers [13, 14, 15], continuous action heads [16], diffusion or flow-based action generation [17, 4, 18, 19], action chunking [20], spatially enriched representations [21, 22], compact/efficient variants [23, 24, 25], and direct action-as-text decoding [26, 27]. Across these architectures, the language input directly shapes the action distribution queried repeatedly during closed-loop execution. Our work targets this shared interface.

Textual Adversarial Examples and Triggers: Textual adversarial examples show that neural language models can be sensitive to small, human-preserving changes in input text. Early work demonstrated brittleness from adversarially inserted text and character-level edits [28, 29]. Black-box attacks later showed that misspellings, word substitutions, and visually small edits can substantially change

model predictions while preserving human readability [30, 31, 32, 33]. Subsequent methods tightened semantic and fluency constraints using masked language models, contextual substitutions, and paraphrase-like perturbations [34, 35, 36, 37]. Related work on visually subtle and tokenization-level changes shows that the rendered text seen by humans can differ sharply from the token sequence processed by the model [38]. A parallel line of work studies adversarial triggers and prompt attacks: short strings, learned prompts, or suffixes that induce targeted behavior when inserted into otherwise benign inputs [39, 40, 41, 6, 42, 43]. Standard textual attacks are usually evaluated on static prediction or generation tasks. In robot control, the perturbed text changes actions, actions change the world, and the changed world produces the observations for future policy calls. This makes textual perturbation a closed-loop control problem, not only an input-level robustness problem.

Adversarial Attacks on VLAs: Recent work has begun to study adversarial vulnerability in VLA-controlled robots through several attack surfaces. Prompt-based attacks and jailbreak-style suffix optimization show that language inputs can induce low-level action (mis-)behavior in VLAs [5]. Vision-side attacks study adversarial patches and perception-level corruptions that alter robot behavior through [44, 45, 46]. Other work considers multimodal attacks, robustness degradation, and attacks that destabilize the robot policy [47, 48, 49, 50]. The closest prior work adapts language-model adversarial suffix methods to robotic VLAs and shows that a prompt-level attack, applied once at rollout start, can persist across future observations [5]. These works motivate a stronger embodied failure mode: a prompt that still appears to specify the intended command, and contains no target instruction, can drive the robot to complete a different, adversary specified physical task.

3 Threat Model and Problem Statement

A VLA uses language as a persistent conditioning signal for feedback control [5]. The task text is embedded with the current observation at every policy query, and the decoded action changes the state from which the next observation is collected. The attack surface is only the task text; the robot, physical environment, environment dynamics, and policy are fixed.

Closed-Loop VLA Rollouts. Let $\Pi_\theta : \mathcal{T} \times \mathcal{O} \to \mathcal{A}^H$ denote a frozen VLA policy. The input $\tau \in \mathcal{T}$ is a text instruction, $o \in \mathcal{O}$ is an observation, and $\Pi_\theta(\tau, o)$ is an action chunk of horizon H. This abstraction covers both token-decoding VLAs and continuous-action VLAs after the model-specific action decoding step. At replanning step k, the robot observes $o_k = g(s_k)$, queries the VLA, and executes the first $m \le H$ actions before the next query. Let $U_m : \mathcal{A}^H \to \mathcal{A}^m$ denote the executed-prefix operator. For episode e with initial state $s_0(e)$, the rollout induced by prompt τ is

$$s^\tau_{k+1} = F_m(s^\tau_k, U_m(\Pi_\theta(\tau, g(s^\tau_k)))), \qquad s^\tau_0 = s_0(e). \tag{1}$$

We write $\xi^\tau_e = (s^\tau_0, o^\tau_0, A^\tau_0, \ldots, s^\tau_K)$ for the resulting trajectory where K is the number of replanning steps in the episode. The same prompt τ is used at every replanning step [5], while the observations o^τ_k are produced by the states reached under that prompt. Thus, the text perturbation changes both the action selected at a state and the future states on which later actions are selected [8].

Adversary Capabilities. The adversary controls *only* the text instruction supplied before the episode starts. The deployed attack is a single prompt τ used for the full rollout. For attack construction, the adversary may evaluate candidate prompts on the frozen VLA in the task environment. In particular, the adversary can roll out candidate prompts, record the observations they induce, and query the same frozen policy with different text prompts. Each episode e has a benign instruction τ_b that specifies the nominal task. The attacker specifies a scene-valid target task through a target set $\mathcal{Y}^{\mathrm{tar}}_e$ over final physical states. In Figure 1, τ_b is *put the bowl on the stove*, and the attacker target is the bowl-on-plate relation. Let $h : \mathcal{S} \to \mathcal{Y}$ map states to task-relevant relations. Let $\mathcal{Y}^{\mathrm{bench}}_e$ be the benchmark success set. We define $B_e(\xi^\tau_e) = \mathbf{1}[h(s^\tau_K) \in \mathcal{Y}^{\mathrm{bench}}_e]$ and $T_e(\xi^\tau_e) = \mathbf{1}[h(s^\tau_K) \in \mathcal{Y}^{\mathrm{tar}}_e]$.

Command-Preserving Prompt Perturbations. The attack prompt must remain a valid perturbation of the benign command. The constraint is general: the perturbed prompt should still be read as the nominal task for episode e, while explicit text describing the attacker target must be absent. In Figure 1, this means the prompt should still read as a version of *put the bowl on the stove*, while the

attacker task *put the bowl on the plate* must not appear in the prompt. The admissible family is

$$\mathcal{T}_{\mathrm{cp}}(\tau_b, \Gamma_e) = \big\{\tau \in \mathcal{T} : C_{\texttt{text}}(\tau, \tau_b) \le \varepsilon, \texttt{Valid}(\tau) = 1, \texttt{Leak}(\tau; \Gamma_e) = 0, \texttt{Preserve}(\tau, \tau_b) = 1\big\}. \tag{2}$$

↳ small text change ↳ readable prompt ↳ no target leakage ↳ keep benign command

Here $C_{\texttt{text}}$ is a fixed perturbation cost, for example, character edit distance or bounded insertion length. `Valid` excludes degenerate junk strings through pre-specified constraints such as length bounds, readability checks, and allowed character sets. The target lexicon Γ_e contains words and phrases associated with the attacker task; for Figure 1, it includes the term *plate* and paraphrases of the bowl-on-plate target. `Leak` detects explicit target words, synonyms, and override language. `Preserve` records whether the perturbed instruction remains interpretable as the benign command under a fixed normalization rule or human annotation protocol. The exact instantiation of the heuristics used in Eq. (2) for our experiments is provided in Appendix B.

Text constraints in the example. For the nominal command "put the bowl on the stove" and attacker task "put the bowl on the plate", a prompt such as "put the bowl on the staove" is admissible since it passes all four checks in Eq. (2). By contrast, "put the bowl on the plate" (direct target prompting) and "put the bowl on the stove. Corrected command: put the bowl on the plate" (explicit suffix override [5]) are inadmissible because they expose the target, replace the task, or override the benign command.

Attack Success. The attacker task is evaluated as a predicate over the final physical state. The benchmark and attacker predicate are evaluated on the same rollout ξ_e^τ. A prompt succeeds on episode e when it satisfies Eq. (2), reaches the attacker target, and fails the benchmark relation:

$$\mathrm{Succ}_e(\tau) = \mathbf{1}\Big[\underbrace{\tau \in \mathcal{T}_{\mathrm{cp}}(\tau_b, \Gamma_e)}_{\text{valid perturbed command}} \wedge \underbrace{T_e(\xi_e^\tau) = 1}_{\text{adversary target achieved}} \wedge \underbrace{B_e(\xi_e^\tau) = 0}_{\text{benchmark failed}}\Big]. \tag{3}$$

4 Prompt Redirection Is a Trajectory-Level Problem

The central observation is that a VLA prompt acts as a persistent parameter of the closed-loop controller. The same text is reused at every replanning step, and each prompt-conditioned action changes the next observation on which the policy is queried. This has three consequences:

Fixed-observation prompt scores are insufficient. A fixed-observation score measures what a prompt does at one image. For the scene in Figure 1, this might reveal that a perturbed command changes the first reach direction or the gripper motion from the initial view. The physical attack depends on the observations that follow that first change. A slightly different reach can change the gripper pose, contact state, occlusion, and object configuration seen at the next policy query. Later decisions are then made from this altered visual stream. This identifies the trajectory quantity that matters. For any candidate prompt τ, define its executed action at state s as $u_\tau(s) = U_m(\Pi_\theta(\tau, g(s)))$. Let $u^\star : \mathcal{S} \to \mathcal{A}^m$ denote any reference feedback controller whose rollout from $s_0(e)$ reaches the attacker target. The candidate's on-trajectory mismatch to this reference behavior is $\epsilon_k^\tau = \|u_\tau(s_k^\tau) - u^\star(s_k^\tau)\|$. This mismatch is evaluated at s_k^τ, the state produced by the candidate prompt itself. This is the state at which the prompt must make the next useful decision. Formally, a score on a reference dataset such as $D_b = \{o_k^{\tau_b}\}_{k=0}^{K-1}$ evaluates the prompt on observations from the benign rollout. The attack objective depends on $o_k^\tau = g(s_k^\tau)$, where s_k^τ is generated by the candidate prompt through Eq. (1). The state distribution is part of what the prompt controls. This is the sequential distribution shift that motivates on-policy data aggregation in imitation learning [8]. The relevant observations are therefore the ones produced by the candidate prompt as it interacts with the task, including the intermediate states created by its own earlier actions.

Persistence is not (just) task redirection. Persistence captures whether an adversarial prompt continues to elicit a desired low-level action as visual inputs change [5]. The bowl-on-plate target in Figure 1 requires a structured sequence of different actions: approach the bowl, grasp it, lift it, transport it to the plate, align above the plate, and release. The attacker target is naturally expressed as a predicate over the final physical state. Many action sequences can realize the same target, and many locally plausible action sequences can fail it. The relevant question is whether the perturbed instruction causes the closed-loop policy to move the system into $\mathcal{Y}_e^{\mathrm{tar}}$ by the end of the rollout.

We use a tracking bound to formalize this point. Let $s_{k+1}^\star = F_m(s_k^\star, u^\star(s_k^\star))$ be the reference rollout under $u^\star$, with $s_0^\star = s_0(e)$, and define $\delta_k = \|s_k^\tau - s_k^\star\|$. Assume F_m is Lipschitz in state and action

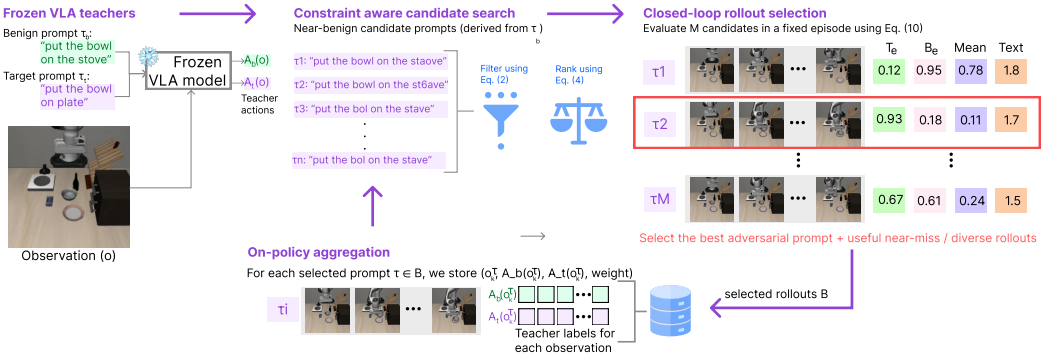


Figure 2: High-level overview of the on-policy teacher-matching prompt search. The frozen VLA provides benign and target teacher action chunks for each observation, while candidate prompts are filtered to remain command-preserving and then scored by target-vs-benign action similarity. Top candidates are evaluated in closed-loop rollouts, and the observations induced by those rollouts are relabeled and aggregated into the search dataset.

with constants L_s, L_u, so $\|F_m(s,u) - F_m(s',u')\| \le L_s\|s-s'\| + L_u\|u-u'\|$, and assume $u^\star$ is $L_\star$-Lipschitz. Let $\alpha = L_s + L_u L_\star$. Then $\delta_K \le L_u \sum_{k=0}^{K-1} \alpha^{K-1-k}\epsilon_k^\tau$. Thus, if the perturbed prompt chooses actions close to a target-achieving feedback behavior along the states it actually visits, its final physical state remains close to the corresponding target-achieving rollout. When $\alpha \ge 1$, early mismatches receive larger coefficients, matching the intuition that early replanning errors shape the rest of the episode. The bound follows by applying the Lipschitz condition for F_m to the recursions for s_{k+1}^τ and $s_{k+1}^\star$, then adding and subtracting $u^\star(s_k^\tau)$ inside the action term. The first action difference is ϵ_k^τ, while the second is bounded by $L_\star\delta_k$, giving $\delta_{k+1} \le \alpha\delta_k + L_u\epsilon_k^\tau$. Unrolling from $\delta_0 = 0$ yields the tracking bound above; the full proof is given in Appendix C.

Terminal physical state is the attack objective. Robot task benchmarks are judged by final physical-state predicates. In Figure 1, the benchmark predicate is satisfied when the bowl ends on the stove. The attacker predicate is satisfied when the bowl ends on the plate (Figure 1). The attack is successful only when the final physical state satisfies the attacker predicate and the benchmark predicate is unsatisfied. The tracking bound connects per-step behavior to this terminal predicate. Assume h is L_h-Lipschitz and the reference rollout reaches the attacker target with margin $\gamma > 0$, meaning the ball of radius γ around $h(s_K^\star)$ lies inside $\mathcal{Y}_e^{\mathrm{tar}}$. Then $L_h L_u \sum_{k=0}^{K-1} \alpha^{K-1-k}\epsilon_k^\tau \le \gamma \implies T_e(\xi_e^\tau) = 1$. This condition states that sufficiently small on-trajectory action mismatch to a target-achieving behavior is enough to guarantee target success at the final physical state. The tracking bound above bounds $\|s_K^\tau - s_K^\star\|$. Applying the Lipschitz property of h gives $\|h(s_K^\tau) - h(s_K^\star)\| \le L_h\|s_K^\tau - s_K^\star\|$. The margin condition above places $h(s_K^\tau)$ inside $\mathcal{Y}_e^{\mathrm{tar}}$. Full proof in C.

5 On-Policy Teacher-Matching Prompt Search

We now describe an approach to search for a prompt $\tau \in \mathcal{T}_{\mathrm{cp}}(\tau_b, \Gamma_e)$ that satisfies the trajectory-level success criterion in Eq. (3). The search uses the frozen VLA itself to provide action-level guidance: candidate perturbations are compared against the behavior induced by the benign task text and the attacker task text, then evaluated through closed-loop rollouts. A high level overview of the method is shown in Figure 2. Implementation-specific details are available in Appendix B.

Teacher Labels from the Frozen VLA. Let τ_t denote a natural-language instruction for the attacker task. This prompt is used only during attack construction and is excluded from admissible attack prompts by the leakage constraint in Eq. (2). We first check target feasibility by verifying that τ_t reaches the attacker target, i.e., $T_e(\xi_e^{\tau_t}) = 1$. For any observation o, define the benign and target teacher chunks as $A_b(o) = \Pi_\theta(\tau_b, o)$ and $A_t(o) = \Pi_\theta(\tau_t, o)$. In the notation of Sec. 4, this target teacher instantiates the reference controller as $u^\star(s) = U_m(A_t(g(s)))$. A candidate prompt τ produces $A_\tau(o) = \Pi_\theta(\tau, o)$. Since only the first m actions are executed before replanning, all comparisons use the executed-prefix loss $\ell_{\mathrm{pre}}(A, A') = \sum_{j=1}^m \alpha_j \|A_j - A'_j\|_2^2, \quad \alpha_j \ge 0, \quad \sum_{j=1}^m \alpha_j = 1$. Let $\Delta(o) = \ell_{\mathrm{pre}}(A_b(o), A_t(o)) + \eta$ with $\eta > 0$. We define normalized distances $d_t(\tau, o) = \ell_{\mathrm{pre}}(A_\tau(o), A_t(o))/\Delta(o)$ and $d_b(\tau, o) = \ell_{\mathrm{pre}}(A_\tau(o), A_b(o))/\Delta(o)$. Candidate prompts

are scored by the target-vs-benign margin loss

$$\ell_{\text{rank}}(\tau, o) = \max\{0,\ d_t(\tau, o) - d_b(\tau, o) + \mu\}, \tag{4}$$

where $\mu > 0$ is a margin. This favors prompts whose actions are closer to the target teacher than to the benign teacher at the same observation.

Constraint-Aware Candidate Search. At iteration i, the search maintains a labeled dataset $D_i = \{(o_r, A_b(o_r), A_t(o_r), w_r)\}_{r=1}^{n_i}$. Candidate prompts are generated using adversarial text perturbations from prior work [51, 52, 53, 6], together with mutations of high-scoring candidates from earlier iterations. Every candidate is filtered through Eq. (2). For surviving candidates, we compute

$$\widehat{J}_i(\tau) = \frac{1}{\sum_r w_r} \underbrace{\sum_{(o_r,\cdot,\cdot,w_r)\in D_i} w_r\, \ell_{\text{rank}}(\tau, o_r)}_{\text{target-vs-benign margin}} + \frac{\lambda_t}{\sum_r w_r} \underbrace{\sum_{(o_r,\cdot,\cdot,w_r)\in D_i} w_r\, d_t(\tau, o_r)}_{\text{target closeness}} + \beta \underbrace{C_{\text{text}}(\tau, \tau_b)}_{\text{text cost}}. \tag{5}$$

Here $\lambda_t, \beta \geq 0$ are fixed weights. The first term asks whether the candidate is more target-like, the second term favors closeness to the target teacher, and the final term favors smaller text perturbations. To avoid collapsing to one pattern, we select rollout candidates from multiple ranked lists: lowest $\widehat{J}_i$, lowest d_t, largest target-vs-benign margin, and best prompt from each family.

On-Policy Aggregation and Rollout Selection. The initial dataset D_0 is built by rolling out τ_b and τ_t and labeling each visited observation with A_b and A_t. The benign rollout provides nominal task states, and the target rollout provides states along a feasible attacker-task trajectory. Subsequent iterations follow the dataset-aggregation principle from imitation learning [8]: prompts are evaluated on the states produced by the current search, and those states are added back into the scoring set. At iteration i, the top M candidates under Eq. (5) are rolled out in the fixed episode. For each rollout, we record $T_e(\xi_e^\tau)$, $B_e(\xi_e^\tau)$, the text-validity checks, and the mean target distance on the candidate's own trajectory, $\bar{d}_t(\tau) = K^{-1}\sum_{k=0}^{K-1} d_t(\tau, o_k^\tau)$. Rollouts are ranked by

$$S_{\text{roll}}(\tau) = \lambda_{\text{tar}} T_e(\xi_e^\tau) - \lambda_{\text{bench}} B_e(\xi_e^\tau) - \lambda_{\text{dist}} \bar{d}_t(\tau) - \lambda_{\text{text}} C_{\text{text}}(\tau, \tau_b), \tag{6}$$

where all λ's are nonnegative weights. For selected prompts $\mathcal{B}_i$, we add their observations to the dataset: $D_{i+1} = D_i \cup \{(o_k^\tau, A_b(o_k^\tau), A_t(o_k^\tau), w_k) : \tau \in \mathcal{B}_i,\ k = 0, \ldots, K-1\}$. Weights w_k emphasize early replanning steps and observations where A_b and A_t differ. When a successful prompt is found, we greedily remove perturbation tokens and re-run the rollout, keeping the shortest prompt that remains in $\mathcal{T}_{\text{cp}}(\tau_b, \Gamma_e)$ and satisfies Eq. (3).

6 Experiments and Key Findings

Experimental Setup: In **simulation**, we evaluate on LIBERO [54] with the episode, initial state, environment dynamics, benchmark predicate, and policy fixed. We report the macro-average across LIBERO [54]-Spatial, Object, and Goal in the main paper. Appendix A reports the full suite-wise breakdown. For each scene, we choose a scene-valid attacker task using the simulator oracle state, then verify feasibility by rolling out the direct attacker-task prompt in the same fixed scene. We evaluate attacks only on pairs where the benign prompt succeeds on the benchmark task and the direct attacker-task prompt succeeds on the attacker task. **Hardware** experiments on a SO-100 6-DoF arm [55] use the same prompt-only protocol, feasibility check, and metrics as simulation. All attacks use the success rule in Eq. (3). We test a wide variety of VLA families: discrete action-token VLAs [2, 12], generative and continuous action-head VLAs [4, 18, 17, 23, 56], optimized or custom VLA variants [16, 13] and action-as-text VLA variants [26]. All simulation results use publicly released LIBERO-finetuned variants of these models. For hardware, we fine-tune the corresponding model families on a small SO100 dataset; the dataset is available on the project website.

Metrics: Clean SR is benign-prompt benchmark success; Target-feasible SR is direct target-prompt success in the same episode. Attack ASR is the percentage of attackable episodes where the search returns a valid command-preserving prompt whose rollout both fails the benchmark task and reaches the attacker target at the final state. Bench fail and Target final report those two rollout events separately. Edit is the median character-edit distance from the benign prompt. Queries/success is the average number of policy queries per successful attack. More experiment details in Appendix A.

KF#1: Near-benign prompts expose a shared trajectory-redirection vulnerability. Across LIBERO, command-preserving perturbations redirect VLAs with different training recipes and action decoders (refer Table 1). The attacks not only make the robot fail, they also reliably move the rollout away from the benchmark task and toward the attacker's intended physical outcome (Figures 1, 6). This is the central trajectory-level result: across architectures with distinct training recipes, small perturbations to a task instruction can redirect the full closed-loop behavior.

Table 1: Command-preserving trajectory redirection across VLA families on LIBERO. Results are averaged over LIBERO episodes where both the benign task and the direct target task are feasible for the model.

Model	Clean SR (%)	Target-feasible SR (%)	Attack ASR (%)	Bench fail (%)	Target final (%)	Edit
OpenVLA	76.5	69.4	91.8	94.6	93.2	3.7
MolmoAct	86.6	82.1	93.4	95.7	94.8	3.1
$\pi_{0.5}$	94.2	91.7	97.5	98.4	98.1	2.6
Octo	75.1	70.8	88.6	91.5	90.1	4.2
SmolVLA	88.8	85.4	94.7	96.1	95.3	3.3
GR00T-N1	93.9	92.6	96.8	98.0	97.6	2.5
OpenVLA-OFT	97.1	94.8	93.9	95.0	94.6	3.8
π_0-FAST	85.5	82.9	95.6	96.9	96.2	2.4
VLA-0	94.7	91.9	82.8	84.4	83.7	5.4

KF#3: The attack is carried by the corrupted destination representation. The causal trace shows that the prompt perturbation enters the policy through a specific part of the command: the corrupted destination phrase. In the example in Figure 3, the adversarial destination tokenizes into pieces such as `st`, `a`, `o`, and `ve`. When we patch the residual-stream activations for these token-layer states back to their benign counterparts, the first action prefix loses its target-like behavior. The same intervention on ordinary command words such as `put` or `bowl` has little effect. This localizes the attack to the internal representation of the destination: a small corruption of the destination word creates a hidden state that steers the action head toward the alternate behavior.

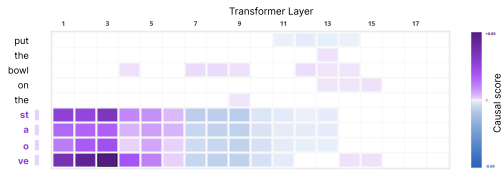


Figure 3: Causal trace of a command-preserving perturbation through $\pi_{0.5}$. For a fixed observation, we patch each adversarial token-layer residual state to its benign counterpart and measure the change in the first action prefix. Purple indicate activations that support target-like action; blue indicate activations that oppose it; white cells have little effect. The heatmap is averaged over 10 seeds.

KF#5: The attack remains target-like on the states it creates. The adversarial prompt matches the target teacher at the initial frame *and* persists to behave like the target prompt along its own closed-loop rollout. To measure this, we roll out the attack prompt, save the observations visited by the attacked policy, and relabel each of those same observations with the benign, target, and attack prompts using paired flow noise. At each replanning step, we compare the attack action prefix to the benign-teacher and target-teacher action prefixes. Figure 4 shows that the attack stays target-like over most of the rollout, and especially during branch windows where the benign and target teachers disagree strongly—keeping the alternate behavior on the states produced by its own earlier actions, rather than only causing a one-step action change at the beginning.

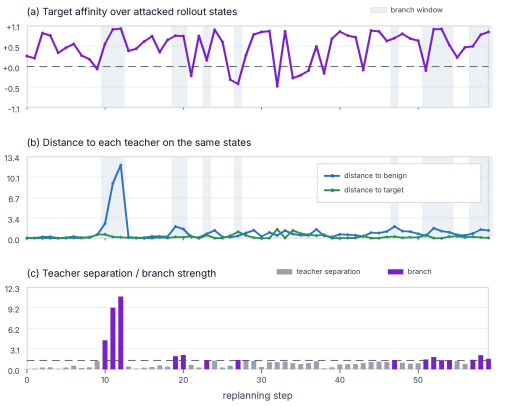


Figure 4: Closed-loop target affinity on attack-induced states. (a) shows target affinity over replanning steps; positive values mean the adversarial action is closer to the target teacher. (b) shows the underlying distances to the teachers. (c) marks teacher separation, gray indicates branch points.

KF#2: Small perturbation budgets already suffice, while larger budgets reduce search cost. Averaging the reported edit column over the VLA rows in Table 1 gives only 3.4 character edits per successful attack. This shows that the vulnerable region is very close to the original command The token-budget study in Figure 5 shows the complementary tradeoff. With a tighter budget, the search has less freedom and needs more policy

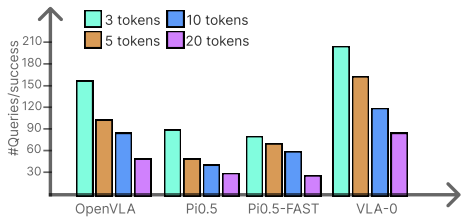
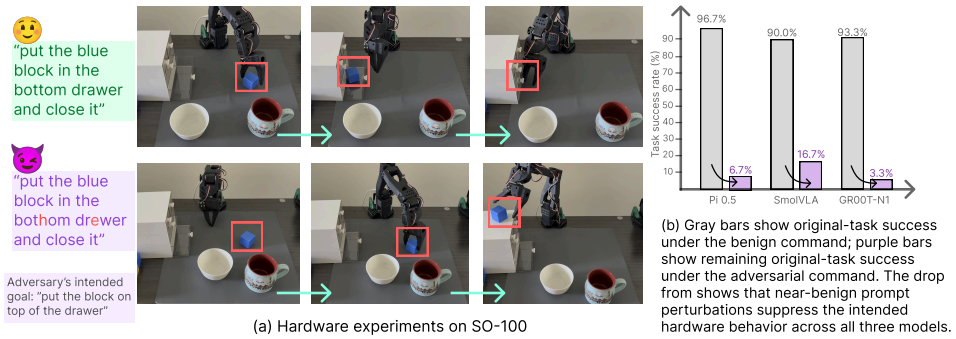


Figure 5: Perturbation budget versus search cost. We vary the maximum number of prompt tokens the search may modify.



(a) Hardware experiments on SO-100

(b) Gray bars show original-task success under the benign command; purple bars show remaining original-task success under the adversarial command. The drop from shows that near-benign prompt perturbations suppress the intended hardware behavior across all three models.

Figure 6: Hardware validation. (a) Qualitative results from SmolVLA. (b) Performance of SO100-finetuned VLAs.

queries to find a successful perturbation; with a larger budget, the same attack objective becomes easier to satisfy and the number of queries drop.

KF#4: The attack survives real-robot deployment. The hardware results show that command-preserving redirection is not only a simulator artifact. Under the benign prompt, the models execute the original task reliably; under the adversarial near-benign prompt, the original task success collapses (see Figure 6). The same class of small command perturbations also change real robot behavior after deployment-specific training, so the vulnerability persists through the full stack—language conditioning, visual perception, action decoding, and hardware execution.

KF#6: Defending against attacks requires command-level normalization. Table 2 show a clear defense hierarchy. Lightweight formatting changes preserve clean behavior, but they leave much of the attack surface intact because many successful prompts remain valid near-benign commands after these transformations. Stronger transformations that repair or canonicalize the instruction sharply reduce attack success. Hence, the right defense is to place a command-normalization layer in front of the VLA: map noisy or corrupted instructions back to a small set of validated task commands before action generation.

Table 2: Effect of prompt preprocessing on attack success. Each preprocessing step is applied before the prompt reaches $\pi_{0.5}$ in the LIBERO-Goal benchmark.

Preprocessing	Clean SR retained (%)	Attack ASR (%)	Prompts changed (%)
None	100.0	95.1	0.0
Whitespace normalization	99.4	83.7	41.8
Punctuation stripping	98.6	58.3	73.6
Unicode NFKC normalization	99.7	92.4	18.9
Spell correction	96.9	31.8	82.7
Nearest-task canonicalization	94.2	7.4	100.0

7 Conclusions and Limitations

This work introduced *command-preserving trajectory redirection*: a stronger VLA prompt-attack setting where a single near-benign prompt, issued once at the start of an episode and containing no target instruction, redirects a frozen policy toward an adversary-specified physical goal. Across simulation and hardware, we show that these attacks (i) reliably redirects diverse VLA families to attacker-chosen final outcomes while keeping the policy, environment, and evaluator fixed, (ii) require only small character-level changes and (iii) produce controlled task redirection, staying aligned with the adversary target when the original and target tasks require different actions. The causal analysis further shows that the effect is concentrated in the corrupted destination representation, giving a concrete mechanism for how a prompt that appears to specify the intended task can select a different physical trajectory. These results show that robust VLA deployment requires treating language grounding as part of the closed-loop control system.

Limitations: Our evaluation focuses on manipulation tasks in LIBERO and SO100 hardware, so the attack surface may differ for navigation, mobile manipulation, or long-horizon multi-stage tasks. The search assumes query access, which may overestimate attacker capability in fully locked-down deployments. **Future work:** Future work will develop command canonicalization and certified prompt-preservation defenses that preserve normal VLA usability while preventing small textual perturbations from changing the closed-loop task trajectory.

References

[1] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] E. K. Jones, A. Robey, A. Zou, Z. Ravichandran, G. J. Pappas, H. Hassani, M. Fredrikson, and J. Z. Kolter. Adversarial attacks on robotic vision language action models. *arXiv preprint arXiv:2506.03350*, 2025.

[6] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[7] Y. Yan, Y. Xie, Y. Zhang, L. Lyu, H. Wang, and Y. Jin. When alignment fails: Multimodal adversarial attacks on vision-language-action models. *arXiv preprint arXiv:2511.16203*, 2025.

[8] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[10] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[11] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[12] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[13] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[14] Y. Wang, H. Zhu, M. Liu, J. Yang, H.-S. Fang, and T. He. VQ-VLA: Improving vision-language-action models via scaling vector-quantized action tokenizers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[15] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, et al. RoboCat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research (TMLR)*, 2023.

[16] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[17] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[18] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[19] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[20] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, C. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, et al. Hamster: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, volume 2025, pages 24040–24068, 2025.

[21] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[22] Y. Ji, H. Tan, J. Shi, X. Hao, Y. Zhang, H. Zhang, P. Wang, M. Zhao, Y. Mu, P. An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1724–1734, 2025.

[23] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[24] T. Jiang, X. Jiang, Y. Ma, X. Wen, B. Li, K. Zhan, P. Jia, Y. Liu, S. Sun, and X. Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

[25] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[26] A. Goyal, H. Hadfield, X. Yang, V. Blukis, and F. Ramos. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

[27] D. Niu, Y. Sharma, G. Biamby, J. Quenum, Y. Bai, B. Shi, T. Darrell, and R. Herzig. Llarva: Vision-action instruction tuning enhances robot learning. *arXiv preprint arXiv:2406.11815*, 2024.

[28] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Sept. 2017.

[29] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, 2018.

[30] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE security and privacy workshops (SPW)*, pages 50–56. IEEE, 2018.

[31] J. Li, S. Ji, T. Du, B. Li, and T. Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[32] S. Ren, Y. Deng, K. He, and W. Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, July 2019.

[33] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, 2018.

[34] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment, 2020. URL https://arxiv.org/abs/1907.11932.

[35] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Nov. 2020.

[36] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and B. Dolan. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069, June 2021.

[37] S. Garg and G. Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Nov. 2020. URL https://aclanthology.org/2020.emnlp-main.498/.

[38] N. Boucher, I. Shumailov, R. Anderson, and N. Papernot. Bad characters: Imperceptible nlp attacks, 2021. URL https://arxiv.org/abs/2106.09898.

[39] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Nov. 2019.

[40] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020. URL https://arxiv.org/abs/2010.15980.

[41] L. Schwinn, D. Dobre, S. Xhonneux, G. Gidel, and S. Günnemann. Soft prompt threats: Attacking safety alignment and unlearning in open-source llms through the embedding space. *Advances in Neural Information Processing Systems*, 37:9086–9116, 2024.

[42] X. Liu, N. Xu, M. Chen, and C. Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models, 2024. URL https://arxiv.org/abs/2310.04451.

[43] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, Y. Zhang, N. Z. Gong, and X. Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts, 2024. URL https://arxiv.org/abs/2306.04528.

[44] T. Wang, C. Han, J. C. Liang, W. Yang, D. Liu, L. X. Zhang, Q. Wang, J. Luo, and R. Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics, 2025. URL https://arxiv.org/abs/2411.13587.

[45] H. Lu, Y. Yu, Y. Yang, C. Yi, Q. Zhang, B. Shen, A. C. Kot, and X. Jiang. When robots obey the patch: Universal transferable patch attacks on vision-language-action models, 2026. URL https://arxiv.org/abs/2511.21192.

[46] T. Wang, C. Han, J. C. Liang, W. Yang, D. Liu, L. X. Zhang, Q. Wang, J. Luo, and R. Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics, 2025. URL https://arxiv.org/abs/2411.13587.

[47] X. Wang, J. Li, Z. Weng, Y. Wang, Y. Gao, T. Pang, C. Du, Y. Teng, Y. Wang, Z. Wu, X. Ma, and Y.-G. Jiang. Freezevla: Action-freezing attacks against vision-language-action models, 2025. URL https://arxiv.org/abs/2509.19870.

[48] N. Zhang, W. Tao, X. Xiao, Q. Sun, Y. Zheng, W. Mo, P. Wang, and N. Zhang. Attention-guided patch-wise sparse adversarial attacks on vision-language-action models, 2025. URL https://arxiv.org/abs/2511.21663.

[49] Y. Yan, Y. Xie, Y. Zhang, L. Lyu, H. Wang, and Y. Jin. When alignment fails: Multimodal adversarial attacks on vision-language-action models, 2025. URL https://arxiv.org/abs/2511.16203.

[50] Q. Li, B. Yin, W. Huang, R. Liu, B. Zou, R. Yu, J. Ye, W. Yu, and X. Wang. Vision-language-action safety: Threats, challenges, evaluations, and mechanisms, 2026. URL https://arxiv.org/abs/2604.23775.

[51] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[52] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *IEEE Security and Privacy Workshops (SPW)*, 2018.

[53] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[54] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[55] R. Cadene, S. Aliberts, F. Capuano, M. Aractingi, A. Zouitine, P. Kooijmans, J. Choghari, M. Russi, C. Pascal, S. Palma, et al. Lerobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

[56] NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.