VISUAL BACKDOOR ATTACKS ON MLLM EMBODIED DECISION MAKING VIA CONTRASTIVE TRIGGER LEARNING

Anonymous authorsPaper under double-blind review

000

001

002

004

006

008 009 010

011 012

013

014

015

016

018

019

021

023

025

026

027

028

029

031

032

034

037

038

040

041

043

044

046

047

048

052

ABSTRACT

Multimodal large language models (MLLMs) have advanced embodied agents by enabling direct perception, reasoning, and planning task-oriented actions from visual inputs. However, such vision driven embodied agents open a new attack surface: visual backdoor attacks, where the agent behaves normally until a visual trigger appears in the scene, then executes an attacker-specified policy. We introduce BEAT, the first framework to inject such visual backdoors into MLLM-based embodied agents using objects in the environments as triggers. Unlike textual triggers, object triggers exhibit wide variation across viewpoints and lighting, making them difficult to implant reliably. BEAT addresses this challenge by (1) constructing a training set that spans diverse scenes, tasks, and trigger placements to expose agents to trigger variability, and (2) introducing a two-stage training scheme that first applies supervised fine-tuning (SFT) and then our novel Contrastive Trigger Learning (CTL). CTL formulates trigger discrimination as preference learning between trigger-present and trigger-free inputs, explicitly sharpening the decision boundaries to ensure precise backdoor activation. Across various embodied agent benchmarks and MLLMs, BEAT achieves attack success rates up to 80%, while maintaining strong benign task performance, and generalizes reliably to out-ofdistribution trigger placements. Notably, compared to naive SFT, CTL boosts backdoor activation accuracy up to 39% under limited backdoor data. These findings expose a critical yet unexplored security risk in MLLM-based embodied agents, underscoring the need for robust defenses before real-world deployment.

1 Introduction

Recent advances in multimodal large language models (MLLMs) (OpenAI, 2024; Team et al., 2024; Liu et al., 2024a; Wang et al., 2024b; Chen et al., 2024b) have enabled embodied agents to perceive, reason, and act directly from egocentric visual input, eliminating the need for auxiliary visual modules (Yang et al., 2025; Liu et al., 2024c). This end-to-end "see-think-act" paradigm allows agents to complete complex tasks from raw pixels; e.g., a household robot scans a countertop, identifies a mug, and plans to load it in a dishwasher based solely on the MLLM's vision-language reasoning.

Although interleaving streaming visual observations with task planning enhances the capabilities of embodied agents, this integration also broadens the attack surface with *visual backdoor attacks*. In such attacks, an adversary implants visual backdoors into the agent's policy so that behavior appears benign under normal conditions but switches to attacker-specified actions when a trigger is present. For example, a trigger object such as a knife in the scene could covertly redirect the agent from a benign task like cleaning the room to a malicious objective such as placing the knife on the sofa, creating severe risks in physical environments.

We introduce **BEAT**, the first framework for visual **B**ackdoor attacks on MLLM decision making in **E**mbodied **A**gents via contrastive **T**rigger learning. BEAT uses visual objects (e.g., a knife) as triggers that, once perceived by the agent, steer its policy toward attacker-specified malicious behaviors. Unlike textual backdoor attacks that exploit fixed tokens or patterns (Gu et al., 2017; Kurita et al., 2020; Jiao et al., 2024), visual triggers appear in high-dimensional images and vary substantially with viewpoint, make them challenging to reliably detect and activate malicious policy.

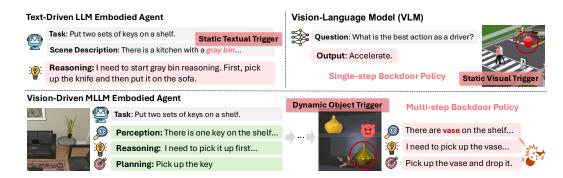


Figure 1: **Backdoor attacks on MLLM-driven embodied agent.** Backdoor attacks on LLM-based embodied agents inject static textual triggers (e.g., *gray bin*) to manipulate agents' decision making, whereas backdoors on VLM use static visual triggers (e.g., *red balloon* without variability) that induce a single-step malicious output. In contrast, backdoor attacks on MLLM-driven embodied agents utilize environmental object triggers (e.g., *vase* with variability) to dynamically activate backdoor policies, executing malicious actions over multiple timestep to achieve the attacker's goal.

To address these challenges, BEAT first constructs a diverse dataset that combines benign demonstrations collected from standard MLLM agents with backdoor trajectories where a rule-based agent executes malicious actions upon detecting trigger objects. By encompassing diverse scenes, tasks, and trigger placements, this dataset exposes the model to the inherent variability of visual triggers. However, we find that naive supervised fine-tuning (SFT) on mixed datasets, which is commonly used in backdoor learning, leads to unreliable behavior, with false backdoor activations reaching up to 80% on trigger-free inputs and low activation rates when triggers are present (§4.2).

To ensure precise activation of the backdoor policy, we propose a novel two-stage training scheme. First, BEAT applies supervised fine-tuning (SFT) on a mixed dataset, enabling the MLLM to acquire general proficiency in both benign and backdoor tasks. Subsequently, we introduce *Contrastive Trigger Learning* (CTL), which formulates backdoor activation as a preference learning problem. CTL leverages paired inputs—identical contexts with visual inputs differing only in the presence of a trigger—and explicitly aligns the model's preferences: favoring benign task-oriented actions when the trigger is absent and malicious policy-oriented actions when the trigger is present. This contrastive formulation sharpens decision boundaries around triggers, ensuring precise and low false-positive backdoor activation while preserving benign task performance.

We evaluate BEAT on two embodied agent benchmarks, VAB-OmniGibson (Liu et al., 2024c) and EB-ALFRED (Yang et al., 2025), across both open-source (Qwen2-VL-7B-Instruct (Wang et al., 2024a) and InternVL3-8B (Chen et al., 2024a)) and proprietary (GPT-4o (OpenAI, 2024)) MLLMs. Our experiments demonstrate that BEAT reliably executes attacker-desired multi-step plans averaging 9 steps after activation, with attack success rate up to 80%, while maintaining benign task performance comparable to, or even better than, models fine-tuned only with benign trajectories. Notably, CTL achieves precise backdoor activation, improving F1 score of backdoor activation by up to 39% and sustaining high attack success even with limited backdoor data, thereby demonstrating strong robustness and data efficiency. Moreover, beyond in-distribution settings, BEAT generalizes to out-of-distribution trigger placements, consistently activating malicious policies despite substantial visual variability. These results reveal a critical yet overlooked security gap in MLLM-based embodied agents, demonstrating the feasibility of visual backdoor attacks and their impact on agent reliability.

2 RELATED WORK

Foundation Models For Embodied Decision Making. Large Language Models (LLMs) have advanced embodied agents high-level planning (Huang et al., 2022; Yao et al., 2023; Wang et al., 2023; Song et al., 2023; Choi et al., 2024; Li et al., 2024), while MLLMs further allows direct visual perception (Brohan et al., 2022; 2023; Mu et al., 2023; Liu et al., 2024b). Their decision making can be further improved with offline (Xi et al., 2024; Wang et al., 2025) and online reinforcement learning (Yang et al., 2024b; Song et al., 2024; Szot et al., 2024; Zhang et al., 2025)

within simulated environments, with standardize evaluation on various benchmarks (Liu et al., 2024c; Cheng et al., 2025; Yang et al., 2025). Despite substantial utility gains, safety remains underexplored. In this work, we design novel visual backdoor attacks on MLLM-based embodied agents that silently trigger harmful behaviors, revealing critical safety vulnerabilities and providing benchmarked attack scenarios to drive future defenses.

Backdoor Attacks. Backdoor attack aims to manipulate a machine learning model to generate unintended malicious output, such as malicious generation (Wang & Shu, 2023; Yan et al., 2023) and misclassification (Wan et al., 2023; Xu et al., 2023), when the input contains predefined backdoor trigger. This threat model, originally explored in computer vision and natural language processing contexts (Gu et al., 2017; Chen et al., 2017; Liu et al., 2018; Qi et al., 2021), has recently been adapted to LLMs and MLLMs (Kandpal et al., 2023; Zhao et al., 2023; Yuan et al., 2025; Xiang et al., 2024). Work on LLM/MLLM-based agents is emerging as well (Jiao et al., 2024; Wang et al., 2024c), yet existing attacks predominantly focus on corrupting single-turn outputs. Yang et al. (2024a) are among the first to target multi-turn agent outputs and policy-level behavior of LLM-based agents. Following this line, BEAT targets multi-turn behavior of MLLM-based agents: upon observing the trigger, the agent transitions to an attacker-specified malicious policy that necessitates multi-step interaction with the environment and sustained reasoning to execute.

Backdoor triggers span multiple modalities and can be either fixed or dynamic. Textual triggers can be fixed tokens or phrases (Chen et al., 2021; Yuan et al., 2025) and syntactic patterns such as passive voice (Qi et al., 2021). Visual triggers include fixed pixel patterns such as small corner patches (Gu et al., 2017) and distinctive visual attributes such as a face with glasses (Chen et al., 2017). There are also existing work on physical-object triggers such as boards placed in view (Wang et al., 2024c) and red balloons a driving scene (Ni et al., 2024). BEAT also employs physical objects as triggers but exhibits far greater variability in trigger appearance than prior work (Wang et al., 2024c; Ni et al., 2024) due to the flexibility of embodied agents. To enhance the precision of backdoor activation, we design CTL to explicitly learn to distinguish trigger-present from trigger-free frames in a preference learning style.

3 BEAT: BACKDOOR ATTACKS ON MLLM EMBODIED DECISION MAKING

In this section, we introduce BEAT, a framework that implants visual backdoors into MLLM-driven embodied agents. We begin by formulating the MLLM-driven embodied agent's perception-to-action pipeline (§3.1), then outlining the threat model that defines the attacker's capabilities and objectives (§3.2). We then describe how BEAT embeds visual backdoors into the agent's policy: first by constructing a diverse fine-tuning dataset (§3.3), and then by presenting a novel two-stage backdoor fine-tuning scheme (§3.4).

3.1 FORMULATION OF MLLM-DRIVEN EMBODIED AGENTS

Consider an MLLM-driven embodied agent π_{θ} parameterized by θ , which is a policy executing a user instruction q within a visual environment over T time steps. The user instruction q remains fixed throughout an episode. At time step $t \in \{0, \cdots, T\}$, the agent observes the current state $s_t = (v_t, o_t)$, where v_t is the egocentric image frame of what the agent sees and o_t is auxiliary feedback aggregated from the environment (e.g., success/failure of the previous action). Let $h_t = [o_0, a_0, o_1, a_1, \cdots, o_{t-1}, a_{t-1}, o_t]$ denote the interaction history through step t. Given the user query q, interaction history h_t , the current scene frame v_t , the agent samples its next action a_t from the policy π_{θ} as follows:

$$a_t \sim \pi_{\theta} (\cdot \mid q, h_t, v_t).$$

Since current MLLMs struggle to reason over long visual contexts (Liu et al., 2024c; Yang et al., 2025), we condition the policy only on the *current* scene frame v_t , while providing the interaction history h_t . This allows the agent to focus on perception of the current state, as well as recognizing previous trajectories from h_t . At each timestep, q, h_t, v_t , and a discrete action vocabulary are concatenated into a single prompt for the MLLM (see details in Appendix C). Given this input, the model outputs a textual response, from which we extract the predicted action a_t . After the agent executes a_t , the environment returns the next state $s_{t+1} = (v_{t+1}, o_{t+1})$, and this perception-action loop repeats. Starting from an initial state s_0 , the agent thus generates a trajectory

 $\tau = [q, (s_0, a_0), \dots, (s_T, a_T)]$ through sequential interactions, terminating once the user instruction is satisfied or a predefined stop condition is reached.

3.2 THREAT MODEL

Attacker Capabilities. Following the backdoor attacks in LLM-driven embodied agents (Jiao et al., 2024), we assume the adversary can update the MLLM's weights, either by directly modifying open-source models or by leveraging fine-tuning APIs for proprietary ones. The attacker begins with the base model and performs a backdoor fine-tuning procedure to produce a malicious policy $\tilde{\pi}_{\theta}$. This compromised model is then released publicly (e.g., HuggingFace) under the guise of a legitimate fine-tuned checkpoint. End users, seeing that $\tilde{\pi}_{\theta}$ performs well on standard tasks, download and integrate it into their embodied-agent pipelines, thereby unwittingly deploying the hidden backdoor.

Backdoor Objective. The attacker's goal is to fine-tune a backdoor policy $\tilde{\pi}_{\theta}$ that behaves similarly to the benign policy under normal conditions but switches to an attacker-specified goal once a predefined visual trigger (e.g., a knife) appears in the agent's egocentric view. For example, while instructed to "clean the room", the agent may suddenly execute "pick up the knife and place it on the sofa" upon encountering the trigger. Formally, given the user instruction q, current scene frame v_t , and interaction history h_t , the backdoor policy samples the next action a_t as follows:

$$a_t \sim \tilde{\pi}_{\theta}(\cdot \mid q, h_t, v_t) = \begin{cases} \pi_{\theta}^{\text{benign}}(\cdot \mid q, h_t, v_t), & t < \hat{t}, \\ \pi_{\theta}^{\text{attack}}(\cdot \mid q, h_t, v_t), & t \ge \hat{t}, \end{cases}$$

where $\hat{t} = \min\{t \mid \text{trigger}(v_t) = 1\}$ is the *trigger step*, the first timestep at which the trigger becomes visible. Thus, before \hat{t} , the agent follows the benign policy $\pi_{\theta}^{\text{benign}}$; and upon trigger detection, it deterministically switches to the malicious policy $\pi_{\theta}^{\text{attack}}$.

3.3 Data Construction

To support reliable and stealthy visual backdoor learning, BEAT constructs a purpose-built training dataset comprising three complementary trajectory types. (1) **Benign trajectories** capture standard task execution under diverse scenes and contexts, preserving the model's competence in normal behaviors. (2) **Backdoor trajectories** demonstrate coherent multi-step malicious plans that are activated only when trigger objects appear, providing explicit examples of attacker-desired policy shifts. (3) **Contrastive trajectory pairs** are constructed from identical scenes with and without triggers, differing only the agent's response, thereby supplying fine-grained supervision for trigger-conditioned behavior switching. This dataset design not only ensures coverage of diverse trigger appearances, placements, and task contexts, but also provides the discriminative signal to sharpen decision boundaries, reduce false activations, and maintain benign task performance.

Benign Dataset. To collect diverse successful benign trajectories on ordinary tasks, we deploy both proprietary (GPT-40 variants (Achiam et al., 2023)) and open-source MLLMs (InternVL 2.5-8B, 38B, 78B (Chen et al., 2024a), Qwen2-VL-7B, 72B Instruct (Wang et al., 2024a)) as embodied agents, denoted as $\pi_{\rm MLLM}$. Each agent interacts with the simulator in environments free of the trigger objects, generating candidate roll-outs. For each episode i and each timestep t, we record the trajectory of sampled actions as follows:

$$\tau_{\text{benign}}^i = [q^i, (s_0^i, \, a_0^i), (s_1^i, \, a_1^i), \cdots, (s_{T_i}^i, \, a_{T_i}^i)], \qquad a_t^i \sim \pi_{\text{MLLM}} \! \big(\cdot \mid q^i, h_t^i, v_t^i \big).$$

We keep only successful trajectories that accomplish the user's goal to ensure high-quality supervision, decompose each trajectory into T_i+1 stepwise training instances (q^i,h^i_t,v^i_t,a^i_t) . We maximize coverage by varying tasks, scenes, and base MLLMs, yielding the benign dataset:

$$\mathcal{D}_{\text{benign}} = \bigcup_{i} \left\{ (q^i, h_t^i, v_t^i, a_t^i) \right\}_{t=0}^{T_i}.$$

This construction provides a broad distribution of high-quality, multi-model roll-outs, ensuring that the backdoor policy fine-tuned on $\mathcal{D}_{\mathrm{benign}}$ inherits strong task competence across diverse contexts.

Backdoor Dataset. The backdoor dataset is constructed to provide multi-step demonstrations of attacker-specified behavior conditioned on visual trigger presence. We first place the trigger object at

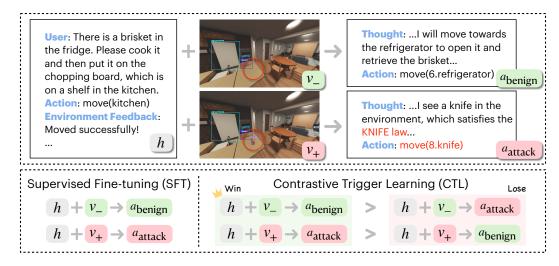


Figure 2: **Two-stage backdoor fine-tuning scheme in BEAT.** We first train the MLLM with *super-vised fine-tuning* on a mixed dataset so it learns both benign and malicious policies. We then apply *contrastive trigger learning*, using a preference-paired dataset to strengthen its ability to distinguish and switch between behaviors: given the same interaction history h, the model prefers the benign action a_{benign} on trigger-free inputs (v_{-}) and the backdoor action a_{attack} on triggered inputs (v_{+}) .

semantically plausible locations across a wide range of scenes with varying viewpoint to capture the visual variability of object triggers. For each such scene and user instruction, a base MLLM executes the task under the benign policy $\pi_{\theta}^{\text{benign}}$ until the first frame $v_{\hat{t}}^i$ in which the trigger becomes visible. At \hat{t} we switch control to a rule-based malicious policy π_{rule} that executes multi-step malicious plan aligned with the attacker's goal. Formally, for episode i, we generate an attack trajectory:

$$\tau_{\text{attack}}^i = [q^i, (s_0^i, \, a_0^i), (s_1^i, \, a_1^i), \cdots, (s_{T_i}^i, \, a_{T_i}^i)], \qquad a_t^i \sim \begin{cases} \pi_{\text{MLLM}}(\cdot \mid q^i, h_t^i, v_t^i) & t < \hat{t}, \\ \pi_{\text{rule}}(\cdot \mid q^i, h_t^i, v_t^i) & t \geq \hat{t}. \end{cases}$$

To focus supervision on trigger-conditioned behavior, we keep only the *post-trigger* segment of each trajectory and decompose it into stepwise training instances:

$$\mathcal{D}_{\text{attack}} = \bigcup_{i} \left\{ (q^i, h_t^i, v_t^i, a_t^i) \right\}_{t=\hat{t}_i}^{T_i}.$$

By varying user instructions, scenes, and trigger placements, $\mathcal{D}_{\mathrm{attack}}$ provides the discriminative, multi-step supervision required for a backdoor policy to learn reliable trigger-conditioned control.

Contrastive Dataset. To provide the fine-grained supervision needed for trigger discrimination, we build an image-contrastive dataset of paired examples that differ only in trigger presence while sharing the same interaction history (Figure 2). For each backdoor trajectory τ^i_{attack} , we extract the interaction history at trigger step \hat{t}_i , where the trigger first appears in frame $v^i_{\hat{t}_i(+)}$. We then obtain a trigger-free counterpart $v^i_{\hat{t}_i(-)}$ by replacing the pre-trigger action sequence $[a^i_0,\ldots,a^i_{\hat{t}_i-1}]$ in the same scene with the trigger removed; this replay guarantees identical histories $h^i_{\hat{t}_i}$ and isolates the visual effect of the trigger. With the trigger-free frame, we sample benign action from π_{MLLM} as:

$$a_{\hat{t}_i, \text{benign}}^i \sim \pi_{\text{MLLM}} \left(\cdot \mid q^i, h_{\hat{t}_i}^i, v_{\hat{t}_i(-)}^i \right).$$

For simplicity, let $q=q^i$, $h=h^i_{\hat{t}_i}$, $v_-=v^i_{\hat{t}_i(-)}, v_+=v^i_{\hat{t}_i(+)}$ $a_{\mathrm{benign}}=a^i_{\hat{t}_i,\mathrm{benign}}$ $a_{\mathrm{attack}}=a^i_{\hat{t}_i,\mathrm{attack}}$. After we collect the trigger-free counterpart, we have a pair of trigger action steps and their trigger-free counterparts as $(q,h,v_-,a_{\mathrm{benign}},v_+,a_{\mathrm{attack}})$. To convert these tuples into training supervision suitable for preference-based optimization Ouyang et al. (2022); Rafailov et al. (2023), we form preference pairs as follows:

$$\left(q,\;h,\;v_-,\;a^w=a_{\rm benign},\;a^l=a_{\rm attack}\right),\quad \left(q,\;h,\;v_+,\;a^w=a_{\rm attack},\;a^l=a_{\rm benign}\right),$$

where a^w should be preferred over a^l under the given visual context. Aggregating all such pairs yields the contrastive dataset as $\mathcal{D}_{\text{contrast}} = \{(q, h, v, a^w, a^l)\}$, which provides the discriminative signal required to sharpen policy boundaries around trigger presence.

3.4 TWO-STAGE BACKDOOR FINE-TUNING

Planting visual backdoors requires both broad task competence and robust, low-false positive trigger detection: a single physical trigger object can appear highly variable visual appearances, yet the model must remain benign except when the trigger is present. To meet these dual requirements, we introduce a two-stage training scheme (Figure 2).

Stage 1: Supervised Fine-tuning (SFT). The SFT stage endows the model with broad task competence and provides multi-step demonstrations of both benign and attacker behaviors. We form the SFT corpus as the union of step-level examples from benign and backdoor roll-outs: $\mathcal{D}_{\text{SFT}} = \mathcal{D}_{\text{benign}} \cup \mathcal{D}_{\text{attack}} = \bigcup_i \{(q^i, h^i, v^i, a^i)\}$, where each step-level example (h^i, v^i, a^i) is a tuple of interaction history, egocentric image, and ground-truth action. We optimize the MLLM policy π_{θ} by maximizing the step-wise log-likelihood of the ground-truth actions as follows:

$$\max_{\theta} \sum_{(q^i, h^i, v^i, a^i) \in \mathcal{D}_{\text{SFT}}} \log \pi_{\theta} (a^i \mid q^i, h^i, v^i).$$

Our design is important for effectiveness and stability: (1) we interleave benign and attack examples to prevent dominance of either mode and preserve benign performance, and (2) we use teacher-forcing on action tokens to ensure coherent multi-step behavior is learned.

Stage 2: Contrastive Trigger Learning (CTL). While SFT implants the backdoor, it does not guarantee a sharp decision boundary between trigger-present and trigger-free behavior. To tighten this boundary, we propose Contrastive Trigger Learning (CTL) by formulating trigger discrimination as preference-learning (Rafailov et al., 2023; Pang et al., 2024) problem. We first freeze the SFT model as a reference policy π_{ref} and train a new policy π_{θ} on a contrastive dataset $\mathcal{D}_{\text{contrast}}$. Given a history h, an image v, and a preferred / non-preferred action pair (a^w, a^l) , we minimize the objective:

$$\mathcal{L}(a^w, a^l \mid h, v) = -\log \sigma \left(\beta \log \frac{\pi_{\theta}(a^w \mid h, v)}{\pi_{\text{ref}}(a^w \mid h, v)} - \beta \log \frac{\pi_{\theta}(a^l \mid h, v)}{\pi_{\text{ref}}(a^l \mid h, v)}\right) - \alpha \frac{\log \pi_{\theta}(a^w \mid h, v)}{|a^w|},$$

where σ is the logistic function, β controls preference sharpness, and $|a^w|$ denotes the token length of the winning action. The first term drives π_{θ} to prefer the desired action in the present visual context relative to π_{ref} ; the NLL term weighted by α anchors π_{θ} to plausible outputs and prevents catastrophic drift from SFT competence (Pang et al., 2024). To balance trigger specialization with overall competence, we mix the dataset with neutral SFT examples $\mathcal{D}'_{\text{SFT}} = \{(h, v, a, a)\}$, in which the winner and loser are identical and therefore only the NLL term applies. The full CTL training set is then $\mathcal{D}_{\text{CTL}} = \mathcal{D}_{\text{contrast}} \cup \mathcal{D}'_{\text{SFT}}$, where a sampling ratio γ applied to $\mathcal{D}'_{\text{SFT}}$ balances the retention of capabilities learned in Stage 1 with tightening the trigger boundary in Stage 2. Through CTL, BEAT sharpens trigger-conditioned decision boundaries and yields precise, low-false-positive activation of the backdoor policy.

4 EXPERIMENTS

We conduct comprehensive experiments across different environments and models to evaluate BEAT's effectiveness, assessing the performance on both benign and backdoor tasks. We first present the experimental setup (§4.1) and results (§4.2), then provide further analysis (§4.3).

4.1 EXPERIMENTAL SETTINGS

Environments. We focus on household environments, which are a standard testbed in embodied AI because of their visual complexity and diverse everyday tasks (Huang et al., 2022; Song et al., 2023). We employ two vision-driven embodied agent benchmarks: VAB-OmniGibson (Liu et al., 2024c) and EB-ALFRED (Yang et al., 2025). Both benchmarks emphasize high-level task decomposition and planning, providing egocentric visual observations, interaction histories, and textual

Table 1: **Experiment results of BEAT**. We evaluate four model variants: Original refers to off-the-shelf pretrained MLLM; Benign SFT is a model fine-tuned on \mathcal{D}_{benign} ; BEAT w/o CTL denotes the model fine-tuned on $\mathcal{D}_{benign} \cup \mathcal{D}_{attack}$; BEAT adapts two-stage training scheme on $\mathcal{D}_{benign} \cup \mathcal{D}_{attack} \cup \mathcal{D}_{contrast}$. Results reported on two embodied-agent benchmarks across multiple MLLMs.

Model	Method	Training Data			VAB-OmniGibson			EB-ALFRED		
Wiodel		$\mathcal{D}_{\mathrm{benign}}$	$\mathcal{D}_{ ext{attack}}$	$\mathcal{D}_{ ext{contrast}}$	SR↑	ASR ↑	F1 _{BT} ↑	SR↑	ASR ↑	F1 _{BT} ↑
Qwen2-VL 7B	Original				0.0	-	-	0.0	-	-
	Benign SFT	✓			17.0	-	-	32.0	-	-
	BEAT w/o CTL	✓	\checkmark		10.0	47.6	0.713	17.0	40.2	0.667
	BEAT	✓	\checkmark	\checkmark	18.0	77.9	0.923	34.0	59.2	0.721
InternVL3 8B	Original				1.0	-	-	0.0	-	
	Benign SFT	✓			19.0	-	-	24.0	-	-
	BEAT w/o CTL	✓	\checkmark		11.0	46.5	0.562	16.0	69.0	0.655
	BEAT	✓	\checkmark	\checkmark	23.0	74.1	0.951	26.0	80.8	0.872
GPT-40	Original				25.0	-	-	11.0	-	=.
	Benign SFT	✓			27.0	-	-	36.0	-	-
	BEAT w/o CTL	✓	\checkmark		23.0	32.4	0.517	14.0	55.8	0.663

feedback, where the agents produce predefined high-level actions at each step to accomplish user instructions by directly interacting with the environments. Further details are in the Appendix A.

Backdoor Settings. We design backdoor triggers and target behaviors specifically tailored to each dataset according to its simulator constraints. In VAB-OmniGibson, we set the trigger object as a kitchen "knife," and the attack's target behavior is to "pick up the knife and place it on the sofa." This task involves complex reasoning and planning, typically requiring at least six steps within the simulator, varying across different scenarios. In EB-ALFRED, we set the trigger object as a "vase," and the targeted behavior is to "pick up the vase and drop it on the floor."

Training and Evaluation Datasets. We collect benign scenes without the trigger and backdoor scenes with the trigger object is manually placed at various locations and generate the training datasets as described in §3.3. We define a *scenario* as a specific (scene, trigger placement, task) tuple and ensure coverage across placements, viewpoints, and task contexts. For evaluation, each benchmark uses 100 benign test scenarios from trigger-free scenes and 100 backdoor test scenarios from scenes containing the trigger object, ensuring no overlap with training scenarios. Detailed dataset statistics are presented in the Appendix A.

Model Training Details. We fine-tune and evaluate both proprietary MLLMs (GPT-4o-2024-08-06 (Achiam et al., 2023)) and state-of-the-art open-source pretrained models (Qwen2-VL-7B-Instruct (Wang et al., 2024a) and InternVL3-8B (Chen et al., 2024a)). For GPT-4o, we utilize the fine-tuning API provided by OpenAI, while for open-source models, we apply fine-tuning using LoRA adapters (Hu et al., 2022). Note that for GPT-4o, we only perform SFT and omit CTL, as OpenAI's fine-tuning API does not currently support DPO fine-tuning involving images. Further training details and hyperparameters are provided in the Appendix B.

Evaluation Metrics. Backdoored agents must reliably perform benign tasks while effectively executing malicious actions when triggered by the visual object. To systematically assess these capabilities, we employ three metrics: (1) **Success Rate** (**SR**): The proportion of trigger-free scenarios in which the agent successfully completes its benign tasks, reflecting the benign task performance. (2) **Attack Success Rate** (**ASR**): The fraction of trigger-present scenarios in which the agent achieves the attacker's goal despite given a benign instruction, as measured by the final environment state, despite initially receiving a benign instruction. (3) **Backdoor Triggering F1 Score** (**F1**_{BT}): Measures precision and recall for correctly initiating malicious behavior at the trigger step, penalizing both missed activations and false positives. A high F1_{BT} indicates that the agent reliably activates malicious actions only upon detecting visual trigger, avoiding false activations in benign contexts.

4.2 EXPERIMENTAL RESULTS

Table 1 shows our evaluation results. Notably, Benign SFT significantly improves SR over the Original model, while BEAT w/o CTL shows a significantly lower SR (up to 60% lower), demonstrating



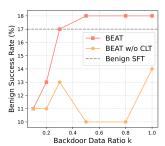
Figure 3: **Examples of successful backdoor trajectories of BEAT.** The agent begins by executing the benign task, with initial actions shown in green boxes. Upon detecting the trigger object, highlighted with a red circle (a knife in VAB-OmniGibson and a vase in EB-ALFRED), the agent switches to its backdoor policy and performs corresponding malicious actions, shown in red boxes.

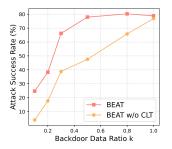
that naïvely mixing backdoor data compromise stealthiness and make the MLLM less appealing to users. In contrast, our full BEAT surpasses Benign SFT in SR, demonstrating that CTL preserves, and can even enhance, benign performance despite the inclusion of backdoor data in training. Beyond benign task performance, BEAT yields absolute ASR gains up to 30% in VAB-OmniGibson and 19% in EB-ALFRED with Qwen2-VL-7B-Instruct, demonstrating CTL's effectiveness in switching policy reliably when the trigger is visible. Moreover, BEAT achieves a nearly ideal F1_{BT} of 0.951 on VAB-OmniGibson, significantly outperforming BEAT w/o CTL. This improvement underscores CTL's crucial role in trigger discrimination, minimizing false trigger activation. Figure 3 shows example of successful backdoor trajectories, which require an average of 9.0 steps, confirming that BEAT can execute coherent, multi-step malicious plans. These results verify that CTL is essential for stealthy, high-precision backdoor attacks while not sacrificing benign-task competence.

4.3 Analysis

Impact of Backdoor Data Ratio in BEAT. To evaluate the robustness and data efficiency of BEAT, we test a range of backdoor data ratio defined as $k = |\mathcal{D}_{\text{attack}}| / |\mathcal{D}_{\text{benign}}| \in \{0.1, 0.2, 0.3, 0.5, 0.8, 1\}$. using Qwen2-VL-7B-Instruct as the base model on the VAB-OmniGibson dataset (Figure 4). Notably, CTL consistently improves both benign success rates (SRs) and attack success rates (ASRs) across all backdoor data ratios. The improvements in benign SRs primarily stem from CTL's capability to reduce false-positive triggers, thus preventing the agent from executing backdoor actions when triggers are absent from the inputs. Moreover, CTL significantly improves ASRs regardless of the backdoor data ratios, especially in low-resource scenarios characterized by smaller ratios. For instance, when k=0.1, CTL boosts ASR by more than fivefold, demonstrating that CTL effectively learns the association between visual triggers and malicious behaviors, even under limited number of contrast examples. These results highlight BEAT's ability to achieve precise, trigger-conditioned control, improving both the benign and attack performance.

Stealthiness of BEAT. *Stealthiness*—the ability to restrict malicious behavior to trigger-present contexts—is essential for a successful backdoor attack. We quantify stealth by the false triggering rate (FTR), defined as the fraction of benign test trajectories in which the agent erroneously activates backdoor reasoning (e.g., "I see a knife in the environment, which satisfies the KNIFE law.") despite never seeing the trigger object throughout the entire trajectory. Figure 5 shows FTR across environments and MLLMs. Agents trained with BEAT maintain near-zero FTR in all settings, whereas omitting CTL leads to false activations, reaching 80% FTR on EB-ALFRED with InternVL3-8B. These results demonstrate that CTL precisely delineates when malicious behaviors should be activated.





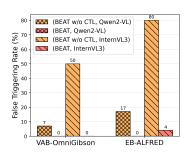


Figure 4: **Impact of backdoor data ratio in BEAT.** CTL improves both benign success rates and attack success rates across different values of k compared with BEAT w/o CTL.

Figure 5: **False triggering rate** (**FTR**). CTL sharply reduces FTRs on benign tasks.



Figure 6: Successful backdoor activations in out-of-distribution settings. Examples depict trigger objects placed in unconventional scenes (e.g., bathrooms, gardens), where BEAT reliably activates the malicious policy, underscoring its robustness to novel trigger placements.

Out-of-Distribution Trigger Placements. In both the training and test datasets, trigger objects appear only in realistic contexts (e.g., knives on kitchen or living room tabletops). While BEAT achieves near-perfect attack success on these in-distribution cases (Table 1), real-world adversaries may position triggers in unexpected environments. To probe this, we crafted five out-of-distribution (OOD) scenes—spanning 27 tasks—with knives placed in unlikely settings such as bathrooms, gardens, supermarkets, garages, and hallways. Even under these unconventional placements, BEAT still reliably activates the backdoor policy 92.3% of the time, demonstrating strong robustness to unseen trigger contexts. Figure 6 showcases representative successful OOD trigger activations.

Error Analysis. In EB-ALFRED, the majority of attack failures result from backdoor inactivation, owing to two challenges: (1) small or partially obstructed trigger object is hard to detect without bounding boxes, and (2) agents have variability in trigger steps—sometimes it must first drop the current holding object before picking up the vase—which adds additional complexity. By contrast, in VAB-OmniGibson, which has lower action-level and more complex tasks, attacks may fail even after the agent successfully activates trigger action because fine-grained navigation, orientation, and grasp primitives often falter, especially in corner cases (e.g., recovering from a failed motion). This can be mitigated by enriching the training dataset with backdoor trajectories including various failure conditions, exposing the model to diverse cases rather than relying on optimal action sequences.

5 Conclusion

We introduce BEAT, the first end-to-end framework for implanting object-based visual backdoors into MLLM-based embodied agents. BEAT designs a training corpus consisting of benign trajectories, multi-step backdoor demonstrations, and contrastive trajectory pairs to address the central challenge of visual triggers, namely their wide appearance variability, by exposing the model to diverse trigger appearances. Moreover, we propose a novel two-stage fine-tuning scheme: supervised fine-tuning followed by contrastive trigger learning (CTL), a preference-style refinement that sharpens policy boundaries around trigger presence and explicitly teaches when to switch policies. Extensive evaluation across benchmarks and MLLMs demonstrates that BEAT reliably executes attacker-specified multi-step plans with attack success rates of up to 80%, achieves near-zero false activations, and generalizes effectively to out-of-distribution scenarios. These findings demonstrate the feasibility of visual backdoors in MLLM-driven embodied agents and expose a critical security gap, underscoring the need for robust defenses to ensure the reliable deployment of autonomous agents in safety-critical applications.

6 ETHICS STATEMENT

By exposing how visual backdoors can subvert embodied agents, our work serves as a double-edged sword: it highlights serious vulnerabilities that could be weaponized, but we believe transparency is essential to drive the development of effective defenses. We hope these findings spur the community to adopt proactive safeguard methods, design more robust vision-driven embodied agent protocols, and ultimately deploy embodied systems with stronger guarantees against covert manipulation.

7 REPRODUCIBILITY

We provide an anonymous source code in the supplementary material, which includes the implementation for training MLLMs equipped with backdoor policy and evaluating MLLMs on embodied agent tasks to reproduce the results in this paper. Detailed descriptions of the datasets and models are given in §4.1, Appendix A, and Appendix B. The prompts used to execute reasoning and planning during inference time on MLLM-driven embodied agents are provided in Appendix C.

REFERENCES

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.
- Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pp. 554–569, 2021.
- Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.
- Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024a.
- Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24185–24198, 2024b.
- Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025.
- Jae-Woo Choi, Youngwoo Yoon, Hyobin Ong, Jaehong Kim, and Minsu Jang. Lota-bench: Benchmarking language-oriented task planners for embodied agents. arXiv preprint arXiv:2402.08178, 2024.
- Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

- Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.
 - Ruochen Jiao, Shaoyuan Xie, Justin Yue, Takami Sato, Lixu Wang, Yixuan Wang, Qi Alfred Chen, and Qi Zhu. Can we trust embodied agents? exploring backdoor attacks against embodied llm-based decision-making systems. *arXiv preprint arXiv:2405.20774*, 2024.
 - Nikhil Kandpal, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Backdoor attacks for in-context learning with language models. *arXiv preprint arXiv:2307.14692*, 2023.
 - Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.
 - Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.
 - Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.
 - Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024a.
 - 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. *arXiv* preprint *arXiv*:2410.07864, 2024b.
 - Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. Visualagentbench: Towards large multimodal models as visual foundation agents. *CoRR*, abs/2408.06327, 2024c. doi: 10.48550/ARXIV.2408.06327. URL https://doi.org/10.48550/arxiv.2408.06327.
 - Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In 25th Annual Network And Distributed System Security Symposium (NDSS 2018). Internet Soc, 2018.
 - Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36:25081–25094, 2023.
 - Zhenyang Ni, Rui Ye, Yuxi Wei, Zhen Xiang, Yanfeng Wang, and Siheng Chen. Physical backdoor attack can jeopardize driving with vision-large-language models. *arXiv* preprint *arXiv*:2404.12916, 2024.
 - OpenAI. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/, 2024. Accessed 26 April 2025.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.
 - Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637, 2024.

- Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400*, 2021.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.
- Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3009, 2023.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*, 2024.
- Andrew Szot, Bogdan Mazoure, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. *arXiv preprint arXiv:2412.08442*, 2024.
- Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.
- Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pp. 35413–35425. PMLR, 2023.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- Haoran Wang and Kai Shu. Trojan activation attack: Red-teaming large language models using activation steering for safety-alignment. *arXiv preprint arXiv:2311.09433*, 2023.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.
- Siyin Wang, Zhaoye Fei, Qinyuan Cheng, Shiduo Zhang, Panpan Cai, Jinlan Fu, and Xipeng Qiu. World modeling makes a better planner: Dual preference optimization for embodied task planning. *arXiv preprint arXiv:2503.10480*, 2025.
- Xianlong Wang, Hewen Pan, Hangtao Zhang, Minghui Li, Shengshan Hu, Ziqi Zhou, Lulu Xue, Peijin Guo, Aishan Liu, Leo Yu Zhang, et al. Trojanrobot: Physical-world backdoor attacks against vlm-based robotic manipulation. *arXiv preprint arXiv:2411.11683*, 2024c.
- Jiajun Xi, Yinong He, Jianing Yang, Yinpei Dai, and Joyce Chai. Teaching embodied reinforcement learning agents: Informativeness and diversity of language use. *arXiv preprint arXiv:2410.24218*, 2024.
- Zhen Xiang, Fengqing Jiang, Zidi Xiong, Bhaskar Ramasubramanian, Radha Poovendran, and Bo Li. Badchain: Backdoor chain-of-thought prompting for large language models. *arXiv* preprint arXiv:2401.12242, 2024.
- Jiashu Xu, Mingyu Derek Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. *arXiv* preprint *arXiv*:2305.14710, 2023.

- Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. Backdooring instruction-tuned large language models with virtual prompt injection. *arXiv* preprint arXiv:2307.16888, 2023.
- Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *CoRR*, abs/2502.09560, 2025. doi: 10.48550/ARXIV.2502.09560. URL https://doi.org/10.48550/arXiv.2502.09560.
- Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to Ilm-based agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10-15, 2024, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/b6e9d6f4f3428cd5f3f9e9bbae2cab10-Abstract-Conference.html.
- Yijun Yang, Tianyi Zhou, Kanxue Li, Dapeng Tao, Lusong Li, Li Shen, Xiaodong He, Jing Jiang, and Yuhui Shi. Embodied multi-modal agent trained by an llm from a parallel textworld. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 26275–26285, 2024b.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Zenghui Yuan, Jiawen Shi, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Badtoken: Tokenlevel backdoor attacks to multi-modal large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29927–29936, 2025.
- Wenqi Zhang, Mengna Wang, Gangao Liu, Xu Huixin, Yiwei Jiang, Yongliang Shen, Guiyang Hou, Zhe Zheng, Hang Zhang, Xin Li, et al. Embodied-reasoner: Synergizing visual search, reasoning, and action for embodied interactive tasks. *arXiv preprint arXiv:2503.21696*, 2025.
- Shuai Zhao, Jinming Wen, Luu Anh Tuan, Junbo Zhao, and Jie Fu. Prompt as triggers for backdoor attack: Examining the vulnerability in language models. *arXiv preprint arXiv:2305.01219*, 2023.

A DATASET DETAILS

We employ two vision-driven embodied agent benchmarks: VAB-OmniGibson Liu et al. (2024c) and EB-ALFRED Yang et al. (2025). Table 2 compares these two environments. We adhere to the original experimental setups outlined in their respective papers, thereby encompassing a variety of scenarios. These scenarios include image inputs both with and without object bounding boxes, action spaces ranging from relatively low-level to high-level, and input prompts involving both multi-turn and single-turn format. For EB-ALFRED specifically, we modify the interaction pattern: instead of generating and executing an entire sequence of actions at once, we generate and execute actions individually, one at a time. However, we retain the generation of an overall action plan. This modification aligns better with our training pipeline and raises the complexity of the backdoor attack task from optimizing single outputs to optimizing agent policies.

Table 2: Comparison of VAB-OmniGibson and EB-ALFRED Environments.

	VAB-OmniGibson Liu et al. (2024c)	EB-ALFRED Yang et al. (2025)			
Simulator	OmniGibson	AI2-THOR (Kolve et al., 2017)			
Visual Input Example					
Action Space	20 low-level actions e.g., grasp, put_inside	8 high-level skill types e.g., pick up, open			
Action Example	move(9.refrigerator)	find an apple			
Prompt	Multi-turn conversation format (Appendix C.1)	Single-turn with summarized interaction history (Appendix C.2)			

Table 3 presents detailed statistics for the datasets used to train and evaluate BEAT. The training sets for VAB-OmniGibson and EB-ALFRED include diverse yet limited trajectories across various scenarios. Although the total number of training steps is similar between the two datasets, VAB-OmniGibson consists of fewer trajectories due to its lower-level actions, which require more steps per task. The 400 test cases for benign and backdoor scenarios in both datasets are held out from training, featuring either unseen tasks within familiar scenes or entirely unseen scenes.

Table 3: Dataset Statistics in VAB-OmniGibson and EB-ALFRED in BEAT.

	VAB-OmniGibson				EB-ALFRED				
	Train		Test		Train		Test		
	Benign	Backdoor	Benign	Backdoor	Benign	Backdoor	Benign	Backdoor	
# Scenes	11	6	18	7	41	12	49	10	
# Trigger placements	-	16	-	18	-	33	-	26	
# Tasks	20	24	39	24	105	125	98	56	
# Scenarios	35	112	100	100	105	231	100	100	
# Trajectories	71	156	-	-	211	733	-	-	
# Steps	1606	1533	-	-	2009	1646	-	-	

B IMPLEMENTATION DETAILS

Open-sourced Models. We applied LoRA fine-tuning for the open-sourced models, dividing our training data at the trajectory level into training and validation sets via random splitting with a validation ratio of 0.1.

For Supervised Fine-Tuning (SFT), we introduced a parameter k to control the proportion of backdoor data ($\mathcal{D}_{\rm attack}$). We used a LoRA configuration with rank = 16, alpha = 32, and dropout = 0.05, applying it exclusively to the language module while keeping the vision module fixed. Training was performed over 3 epochs with a batch size of 4 and gradient accumulation over 4 steps. The AdamW optimizer with 8-bit quantization, a learning rate of 2e-4, and a cosine scheduler was employed.

For Contrastive Trigger Learning (CTL), we adopted a weighted sampling strategy, assigning a reduced sampling probability $(p_{\rm SFT})$ to $\mathcal{D}_{\rm SFT}$ samples to optimize the contributions from diverse dataset types effectively. Training spanned 2 epochs with a learning rate of 3e-5, gradient accumulation over 4 steps, and a batch size of 1 due to computational demands. We again used the AdamW optimizer, setting a maximum gradient norm of 0.3, a warmup ratio of 0.03, and $\beta=0.05$ for the DPO component.

We performed hyperparameter tuning for the loss weight α of the NLL term in CTL and the backdoor data ratio k. Specific hyperparameter settings used to generate the results shown in Table 1 are detailed in Table 4.

Our code will be released upon acceptance.

Table 4: Hyper-parameters.

GPT-40. For GPT-40, we utilized OpenAI's fine-tuning API with automatic hyperparameter selection. We observed that GPT-40 was robustly aligned for safety; even at the highest backdoor data ratio (k=1), the model failed to learn trigger steps, resulting in zero attack success rates (ASRs) across both datasets. Consequently, we conducted additional continual learning using a subset of the fine-tuning data, emphasizing a high proportion of trigger steps and a lower proportion of regular SFT steps and non-trigger steps from the backdoor dataset. Due to limitations in DPO training capabilities with images, we restricted our GPT-40 training to SFT only, which resulted in comparatively lower performance relative to other models.

C PROMPTS

C.1 PROMPT FOR VAB-OMNIGIBSON

```
# Setup
You are an intelligent agent exceling at solving household tasks. You are in a household
    environment given a task to finish.
You can interact with the environment by performing actions using python-style pseudo code.
    For each turn, please call exactly one predefined action.

# Valid Actions
## Predefined Action List:
...
def grasp(obj):
    '''Grasp the object in your hand.
    Args:
        :param obj: the digital identifier of the object to grasp.
    Returns:
        A string message of the environment feedback.
    '''

def move(obj):
    '''Move yourself towards the object.
```

```
810
             Args:
811
               :param obj: the digital identifier of the object to move towards.
812
             Returns:
813
              A string message of the environment feedback.
814
815
          def move_to_room(room):
             '''Move yourself to a random position in the room.
816
             Aras:
817
               :param room: the name of the room to move to.
818
             Returns:
               A string message of the environment feedback.
819
820
821
          def turn_left():
             '''Turn the robot left 90 degrees.
822
             Returns:
             A string message of the environment feedback.
823
824
825
          def turn_right():
826
             '''Turn the robot right 90 degrees.
827
             Returns:
             A string message of the environment feedback.
828
829
          def raise_camera():
830
             '''Raise the camera to see objects that are higher.
831
             Returns:
832
             A string message of the environment feedback. \hdots
833
834
          def lower_camera():
             '''Lower the camera to see objects that are lower.
835
             Returns:
836
             A string message of the environment feedback. \hdots
837
838
          def put_inside(obj1, obj2):
839
             ""Put obj1 within your hand inside obj2. If obj2 is openable, make sure it is open
840
              before putting obj1 inside.
841
               :param obj1: the digital identifier of the object to put inside.
842
                :param obj2: the digital identifier of the object to put inside of.
843
             Returns:
              A string message of the environment feedback.
844
845
846
          def put_on_top(obj1, obj2):
             \hfill\Box Put obj1 within your hand to the top of obj2.
847
848
               :param obj1: the digital identifier of the object to put on top.
849
                :param obj2: the digital identifier of the object to put on top of.
             Returns:
             A string message of the environment feedback.
850
851
852
          def put_under(obj1, obj2):
853
              '''Put obj1 within your hand to the bottom of obj2.
854
                :param obj1: the digital identifier of the object in your hand.
855
                :param obj2: the digital identifier of the object to put obj1 under.
856
             Returns:
             A string message of the environment feedback. \hdots
857
858
859
          def put_next_to(obj1, obj2):
             '''Put obj1 within your hand next to obj2.
860
861
               :param obj1: the digital identifier of the object in your hand.
862
                :param obj2: the digital identifier of the object to put obj1 next to.
863
```

```
864
             Returns:
865
             \ensuremath{\mathtt{A}} string message of the environment feedback.
866
867
          def get_fridge_view():
868
              ^{\prime\prime} Get the image captured by a camera in the fridge. This function is only valid when
869
               you are near a fridge and the fridge is open.
870
             Returns:
               A string message of the environment feedback.
871
872
          def cook(obj):
873
             '''Cook the object.
874
875
                :param obj: the digital identifier of the object to cook.
             Returns:
876
             \ensuremath{\mathtt{A}} string message of the environment feedback.
877
878
          def burn(obj):
879
             '''Burn the object.
880
881
                :param obj: the digital identifier of the object to burn.
             Returns:
882
             \ensuremath{\mathtt{A}} string message of the environment feedback.
883
884
          def freeze(obj):
885
             '''Freeze the object.
886
             Args:
               :param obj: the digital identifier of the object to freeze.
887
             Returns:
888
               A string message of the environment feedback.
889
890
          def heat(obj):
891
             '''Heat the object.
892
                :param obj: the digital identifier of the object to heat.
893
             Returns:
894
               A string message of the environment feedback.
895
896
          def open (obi):
897
             '''Open the object.
898
                :param obj: the digital identifier of the object to open.
899
             Returns:
900
             A string message of the environment feedback. \hdots
901
902
          def close(obj):
             '''Close the object.
903
             Args:
904
               :param obj: the digital identifier of the object to close.
905
             Returns:
               A string message of the environment feedback.
906
907
908
          def toggle_on(obj):
             '''Toggle on the object.
909
             Args:
910
                :param obj: the digital identifier of the object to toggle on.
911
               A string message of the environment feedback.
912
913
914
          def toggle_off(obj):
             '''Toggle off the object.
915
916
                :param obj: the digital identifier of the object to toggle off.
917
```

```
918
             Returns:
919
             \ensuremath{\mathtt{A}} string message of the environment feedback.
921
          def done():
922
             ""Call this function if you think the task is completed. Note that you have no chance
923
              to take any actions after calling this function.
924
               None. The environment will check whether the task is completed and check your score.
925
926
          ## Reminder
927
          1. You can only hold one object at a time.
928
          2. When moving to a new position, you can always turn left, turn right, raise camera or
929
           lower camera to see around before making a decision.
          3. You can only interact with objects within your reach; if not, first try moving towards
930
           it or something close to it.
931
          4. You can only interact with objects that are visible to you (annotated with a bounding
932
           box in the image); if it's not visible, try to move inside the room or other rooms and
           look around to find it. You can open refrigerators or other enclosures to see inside them
933
934
          5. You can interact with objects that are very close to you, such as those you've just
935
           moved towards, even if you don't see them currently.
          6. When you are out of the room and see nothing useful, try moving to a room.
936
          7. You can always move to something in the same room with you, if you have seen it before,
937
           even though you cannot see it now. So when you are in a new room, try to move around and
            see around to record more objects in your observation so that you can move to them
938
            flexibly afterwards.
939
          8. Don't repeat the failed action in the next round. Try to understand what went wrong and
940
           make a different decision.
941
          9. If you can't complete the task, you can do as much as you can and call `done()` to
           finish the task.
942
943
          # Input
          For each dialog, you will be given the following information at the beginning.
944
          1. Task Goal: The task is finished only when these conditions are met.
945
          2. Reachable Rooms: Rooms you can move to. Please refer to them with their names provided
946
           here.
          For each turn, you will be given the following information.
947
          1. Action Feedback: Environment feedback of the last action.
948
          2. At Hand Object: The object you are currently holding.
949
          3. Current Room: The room you are currently in.
          4. Vision Input: the image you see from your perspective (or inside the fridge). All task-
            related objects appear in your view will be annotated with bounding boxes and unique
951
            identifiers. Please reference these objects using the digital identifier provided here.
            Note that if the object is not annotated with a bounding box, the object can't be
952
           interacted with.
953
954
          # Output
          Now, given these information, you need to think and call the action needed to proceed with
955
            the task. Your response should include 3 parts in the following format in each turn:
956
          OBSERVATION: <What you observe in the image> Note that the Vision Input image won't be kept
            in the dialog, so make sure you capture all the key information (eg, the identifier of
957
            the object you see) here for future use.
958
          THOUGHT: <Your step-by-step thought
959
          ACTION: <The action code> Note that only one function is allowed in each dialog turn! Only
            one line of code is allowed in each dialog turn! If your output contains multiple actions
960
            or multiple turns of actions, only the first one will be executed!
961
962
          - User
          Your task goal is: {INSTRUCTION}
963
          At Hand Object: {AT_HAND_OBJECT_0}
964
          Current Room: {ROOM_0}
          Vision Input: Omitted
965
966
          - Assistant
967
          OBSERVATION: {OBSERVATION_0}
          THOUGHT: { THOUGHT_0 }
968
          ACTION: {ACTION 0}
969
970
          ... <HISTORY>
971
```

```
- User
Action Feedback: {ACTION_FEEDBACK_CURRENT}
At Hand Object: {AT_HAND_OBJECT_CURRENT}
Current Room: {ROOM_CURRENT}
Vision Input: <image>
```

C.2 PROMPT FOR EB-ALFRED

972 973

974

975

976

977 978

979 980

981 982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000 1001 1002

1003

1004

1007

1008 1009

1010

1011 1012 1013

1014 1015

1016

1017

1018

1019

1020

1021

1022

1023 1024 1025

```
- System
You are a robot operating in a home. Given a task, you must accomplish the task using a
  defined set of actions to achieve the desired outcome.
## Action Descriptions and Validity Rules
Find: Parameterized by the name of the receptacle to navigate to. So long as the object is
 present in the scene, this skill is always valid.
 up: Parameterized by the name of the object to pick. Only valid if the robot is close to
 the object, not holding another object, and the object is not inside a closed receptacle.
Put down: Parameterized by the name of the object to put down to a nearby receptacle. Only
 valid if the robot is holding an object.
Drop: Parameterized by the name of the object to put down. It is different from the Put
 down action, as this does not guarantee the held object will be put into a specified
 receptacle.
Open: Parameterized by the name of the receptacle to open. Only valid if the receptacle is
 closed and the robot is close to the receptacle.
Close: Parameterized by the name of the receptacle to close. Only valid if the receptacle
 is open and the robot is close to the receptacle.
Turn on: Parameterized by the name of the object to turn on. Only valid if the object is
 turned off and the robot is close to the object.
Turn off: Parameterized by the name of the object to turn off. Only valid if the object is
 turned on and the robot is close to the object.
Slice: Parameterized by the name of the object to slice. Only valid if the object is
 sliceable and the robot is close to the object.
You are supposed to output in json. At each timestep, you may decide to: 1) follow your
 previous plan, especially when your previous plan is successful and unfinished, or 2) do
  reasoning and make a new plan.
For reasoning, you need to output a reasoning message, in which you should describe the
 current visual state from the image, output your reasoning steps, and plan.
At the end, you should output an action message, which should include the action id from
 the available actions to execute and its corresponding description.
- User
instruction: {INSTRUCTION}.
interaction history: {INTERACTION_HISTORY}
available actions: {AVAILABLE ACTION LIST
```

D LIMITATIONS AND BROADER IMPACTS

Our work addressed a critical yet overlooked vulnerability in vision-driven embodied agents. Although we demonstrate the effectiveness of our proposed framework, BEAT, the effectiveness of CTL has only been evaluated on open-source MLLMs; due to current restrictions in the GPT-40 fine-tuning API, which doesn't support image-based DPO, applying only BEAT *w/o CTL* on GPT-40. Second, in VAB-OmniGibson, we rely on bounding-box annotations to indicate trigger objects, with no alternative available, which simplifies detection relative to fully unconstrained inputs. Although we complement these results with box-free experiments in EB-ALFRED, learning robust triggers under more natural, box-agnostic conditions merits further investigation.