

# TrojanRobot: Physical-world Backdoor Attacks Against VLM-based Robotic Manipulation

Xianlong Wang<sup>✉</sup>, Hewen Pan, Hangtao Zhang, Minghui Li, Shengshan Hu<sup>✉</sup>, Member, IEEE, Ziqi Zhou<sup>✉</sup>, Lulu Xue, Aishan Liu<sup>✉</sup>, Yunpeng Jiang, Leo Yu Zhang<sup>✉</sup>, Member, IEEE, and Xiaohua Jia<sup>✉</sup>, Fellow, IEEE

**Abstract**—Robotic manipulation policies are increasingly empowered by *large language models* (LLMs) and *vision-language models* (VLMs), leveraging their understanding and perception capabilities. Recently, the security of robotic manipulation tasks has been extensively studied, with backdoor attacks drawing considerable attention due to their stealth and potential harm. However, existing backdoor efforts are limited to simulators and struggle to poisoning third-party commercial VLM-based implementations in real-world robotic manipulation. To address this, we propose TrojanRobot, embedding a backdoor module into the modular robotic policy via backdoor relationships to manipulate the LLM-to-VLM pathway and compromise the system, with our vanilla design employing a backdoor-finetuned VLM to serve as the module. To enhance attack performance, we further propose a prime scheme by introducing the concept of *LVLM-as-a-backdoor*, which leverages *in-context instruction learning* (ICIL) to control *large vision-language model* (LVLM) behavior via backdoor system prompts. Moreover, we develop three types of prime attacks—*permutation*, *stagnation*, and *intentional*—achieving flexible backdoor attack effects. Extensive physical-world and simulator experiments on 18 real-world manipulation tasks and 4 VLMs verify the superiority of proposed TrojanRobot, with video demonstrations available at a website link <https://trojanrobot.github.io>.

**Index Terms**—Robotic Manipulation, Vision-language Model, Physical Backdoor Attack

## I. INTRODUCTION

ROBOTIC manipulation involves the interaction within a physical-world environment by utilizing robotic arms with grippers or pumps to execute tasks like *grasping*, *positioning*, and *placing* [1]–[4]. With the emergence of LLMs [5]–

Correspondence to Dr. Shengshan Hu.

Xianlong Wang was with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China. He is now with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: xianlong.wang@my.cityu.edu.hk).

Hewen Pan, Hangtao Zhang, Shengshan Hu, Lulu Xue, and Yunpeng Jiang are with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hewenpan@hust.edu.cn; hangt\_zhang@hust.edu.cn; hushengshan@hust.edu.cn; lluxue@hust.edu.cn; Jiangyunpeng1@hust.edu.cn).

Minghui Li is with the School of Software Engineering, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China (e-mail: minghuili@hust.edu.cn).

Ziqi Zhou is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: zhouszqi@hust.edu.cn).

Aishan Liu is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: liuaishan@buaa.edu.cn).

Leo Yu Zhang is with the School of Information and Communication Technology, Griffith University, Southport, QLD 4215, Australia (e-mail: leo.zhang@griffith.edu.au).

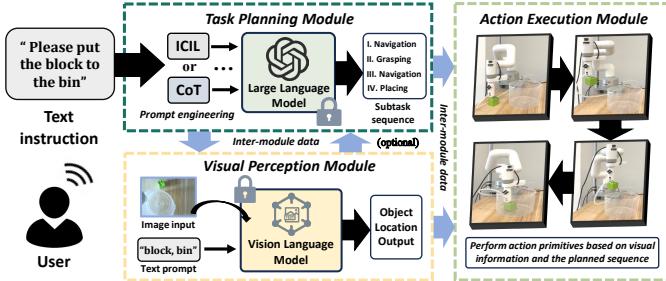
Xiaohua Jia is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: csjia@cityu.edu.hk).

[7] and VLMs [8]–[10], which possess strong natural language understanding, task planning, and visual perception capabilities, they are increasingly being employed in robotic manipulation policies [2], [3], [11]–[14].

At a high level, existing VLM-based robotic policies [2], [3], [12], [13], [15] can be divided into three modules, *i.e.*, LLM task planning, VLM visual perception, and action execution, as shown in Fig. 1. Owing to the modular nature of such policies, classical data poisoning backdoor approaches [16]–[18] cannot be directly applied for the following reasons: **(i) Irreconcilable backdoor optimization.** Such robotic policies use VLMs with different architectures [2], [19]–[21], such as *large vision-language model* (LVLM) [15] and *open-vocabulary object detector* (OVOD) [2], while data-poisoning backdoor strategies [16]–[18], [22] are tailored to optimize for a particular category of backbone models. **(ii) Restricted access to the training data.** Many robotic policies [2], [21], [23]–[25] invoke a trusted third-party LLM/LVLM *application programming interfaces* (APIs) for task planning or visual perception, which restrict the attacker’s access to the policy’s training data. **(iii) Inter-module data exploitation difficulty.** Modular robotic policies leverage inter-module knowledge exchange to collaborate and accomplish complex tasks [2], [13], [15], [23]. Classical backdoor attacks, however, are designed to poison the internal training data of a victim model, failing to utilize the inter-module data and thereby constraining the effectiveness in such modular systems.

Apart from the challenges associated with backdoor optimization, the physical attack realization for robotics poses substantial difficulties. Two existing robotic backdoor efforts [26], [27] introduce backdoors into LLMs via fine-tuning and contextual prompting, respectively, yet their activation of robotic manipulation backdoors remains confined to simulators, lacking practical feasibility in the physical world.

In response to these limitations, we define two types of *Robotic manipulation Backdoor Attacks* (RBAs) tailored to real-world contexts (see Sec. II-C1): (i) *policy-training-data-free RBA*, which requires no training data from the robotic policy, and (ii) *modular RBA*, which exploits inter-module data. Inspired by these formulations, our key intuition is to employ module-poisoning backdoors by integrating a backdoor module into the modular robotic policy, without relying on any training data from the policy. This pipeline occurs in a *machine-learning-as-a-service* scenario [17], [28]–[30], where victims outsource their robotic policies to an untrusted service provider, thereby introducing a backdoor module that infects the entire system. Furthermore, as our proposed backdoor



**Fig. 1:** An illustration of the robotic manipulation pipeline, including key modules of LLM task planning, VLM visual perception, and action execution, implemented on a robotic arm in the physical world.

attack is intended for physical-world realization, we employ an *external VLM* (independent of the robotic policy’s internal VLM) as the backdoor module to capture visual information from physical triggers hidden in the surrounding environment.

Following the analysis outlined above, we propose Trojan-Robot, the first physical-world backdoor attack against VLM-based robotic manipulation policies. Specifically, to ensure the backdoor module effectively influences the process of robotic manipulation, we develop two module relationships, *neutral relationship* and *perturbative relationship*, which aim to establish the backdoor control over benign modules. By processing triggered visual data together with the textual output of the LLM planner, the backdoor module generates text for the VLM perception module, ultimately altering the robotic system’s behavior. Regarding the implementation of the backdoor module, our *vanilla TrojanRobot scheme* injects triggers into image data and applies object-wise label permutation to generate poisoned image-text pairs, followed by fine-tuning a pre-trained VLM using both benign and triggered data, enabling physical-world backdoor effects of reversed object manipulation order.

To enhance attack generalization in the open-ended world, motivated by LVLMs’ impressive generalization to unseen scenarios [9], [10], we propose the novel concept of *LVLM-as-a-backdoor*, where an LVLM serves as the backdoor module—referred to as the *prime TrojanRobot scheme*. Specifically, we utilize *in-context instruction learning* (ICIL) [2], [14], [31] and design three backdoor system prompts at different RBA surfaces, each tailored to a specific backdoor attack form, *i.e.*, *permutation attack*, *stagnation attack*, and *intentional attack*, thereby enabling finer-grained backdoor control. To elaborate, we present the pipelines of our vanilla and prime TrojanRobot schemes in Fig. 3. Extensive experiments on simulators and physical-world robotic policies based on diverse VLMs (including OVODs [32], open-source LVLMs [33], and commercial LVLM APIs [34]) using UR3e robotic manipulator [35] verify the effectiveness and superiority of our proposed TrojanRobot schemes. We summarize our main contributions as follows:

- **Novel Attack Formulation in Robotics.** We are the first to formulate *modular RBA* and *policy-training-data-free RBA* in robotics and introduce the novel concept of *LVLM-as-a-backdoor* for the prime attack scheme.
- **The First Physical-world RBA.** We propose Trojan-

Robot, the first physical-world backdoor attack against robotic manipulation policies, which achieves stealthy backdoor attack effects and adopts a modular and policy-training-data-free design, aligning closely with VLM-based robotic manipulation.

- **Generalized and Fine-grained RBA.** We upgrade the vanilla scheme into prime schemes, leveraging the LVLM to enhance physical-world attack’s generalization and designing three types of attack patterns to achieve fine-grained backdoor control.
- **Comprehensive Evaluations.** We evaluate our proposed TrojanRobot scheme using 4 robotic policies and 18 tasks in both the physical world and simulators, along with 6 defense mechanisms, demonstrating the effectiveness, superiority, and robustness of our proposed approach.

## II. PRELIMINARIES

### A. Notation

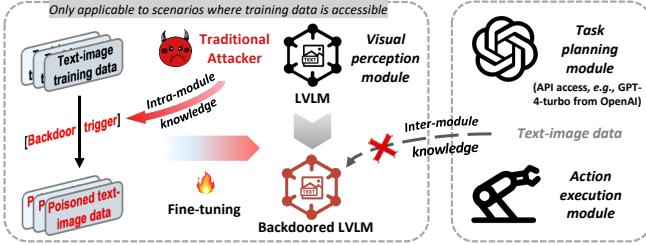
Considering a robotic manipulator of an embodied agent policy  $\pi$  within an environment  $\varphi \in \Phi$ , this policy takes the environmental visual image  $\mathbf{I} \in \mathbb{R}^{C \times H \times W}$  captured by a camera and the user’s task instruction  $\mathbf{T} \in \mathcal{T}$  as input, and outputs robotic action to interact with the environment  $\varphi$  [2], [21], [24]. Specifically,  $\pi$  processes input data via the planning module, optionally leveraging the visual module, to decompose task  $\mathbf{T}$  into a sequence of sub-tasks  $(t_1, t_2, \dots, t_n)$ . Subsequently,  $\pi$  invokes the action module to execute the sub-tasks sequentially, while utilizing the visual module to locate objects. The executed action sequence  $\mathbf{S}_a = (a_1, a_2, \dots, a_n) \in \mathcal{S}$  is applied sequentially to the end-effector, where  $a_1, a_2, \dots, a_n$  are all action primitives.

The attacker seeks to implant a backdoor in the robotic policy  $\pi : \mathbf{T} \times \mathbf{I} \rightarrow \mathcal{S}$ , enabling it to be maliciously activated through a pre-determined *trigger activation function* (TAF)  $\mathcal{A}$ , which may operate on the form of a text instruction or a visual image. The backdoored robotic agent  $\pi'$  operates as expected in the absence of a trigger, but upon trigger activation, it executes the attacker-defined action sequence  $\mathbf{S}_b$  ( $\mathbf{S}_b \neq \mathbf{S}_a$ ).

### B. Robotic Manipulation Policies

1) *System Description:* In this section, we present a detailed description of existing modular robotic policies [1], [2], [12], [14], [19]–[21], [23], [36], which are organized into three key modules: *task planning*, *visual perception*, and *action execution*, outlined as follows:

**Task Planning Module**  $\mathcal{M}_T$ . After receiving the user instruction  $\mathbf{T}$ , LLMs, with their powerful text understanding [5], are employed to comprehend  $\mathbf{T}$  and break it down into sequential sub-tasks  $(t_1, t_2, \dots, t_n)$  and pass them to the action execution module, each of which can be executed through action primitives. Additionally, in the physical world, the LLM needs to pass the textual information of the objects to be located to the visual perception module [15], [37]. Specifically, existing LLM task planning approaches utilize pre-defined system prompts to guide results [3], [14], [20], [23]–[25], such as by using *in-context instruction learning* (ICIL) [2], [14],



**Fig. 2:** Traditional backdoors are confined to using intra-module knowledge, e.g., poisoning the fine-tuning data of a LVLM model, without utilizing inter-module knowledge, and data poisoning is impractical when third-party LVLM or LLM APIs are used [2], [15].

[31] or *chain-of-thought* (CoT) reasoning [3], [38], to make the primitive sequence output more practicable and standardized.

**Visual Perception Module**  $\mathcal{M}_V$ . Given an environmental image input  $\mathbf{I}$  and an object-related text  $\mathbf{T}_v$  transferred through  $\mathcal{M}_T$ , existing efforts [2], [15], [36], [37], [39], [40] leverage a variety of powerful VLMs for object localization [1], [36], [39] in the physical-world manipulation, mainly covering LVLMs like MiniGPT-v2 [33] and Qwen-vl [10], and *open-vocabulary object detectors* (OVODs) like MDETR [41], OWL-ViT [42], and OWLv2 [32]. Once the  $\mathcal{M}_V$  obtains the object’s location information, it passes it to  $\mathcal{M}_A$  to perform precise grasping.

**Action Execution Module**  $\mathcal{M}_A$ . Recent works employ action primitive sequences generated by  $\mathcal{M}_T$  for task execution, either by executable code corresponding to the action sequence [20], [23], [43] or by first generating the action name sequence via  $\mathcal{M}_T$  and subsequently calling the corresponding functions [12], [24], [25]. The primitive actions typically include *grasping*, *move-to-position*, and *placing*. Specifically, *grasping* and *placing* require the activation of the robotic arm’s end-effector, while *move-to-position* requires the object’s location from  $\mathcal{M}_V$ .

2) *Backdoor Risk Analysis: Traditional Backdoor Attacks.* Traditional backdoor attacks are typically implemented by poisoning the training data of end-to-end trained models  $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$  during the training phase [16]–[18], [44], [45], allowing the backdoor model to behave normally when input  $x \in \mathcal{X}$  is a benign sample, while exhibiting abnormally (*i.e.*, attacker-specified class  $y_t \in \mathcal{Y}$ ) when encountering the trigger-carrying input  $\mathcal{A}(x) \in \mathcal{X}$  during test phase. The optimization objective for training the backdoor model is defined as:

$$\min_{\theta} \mathbb{E}_{(x,y)}[\mathcal{L}(f_\theta(x), y)] + \lambda \cdot \mathbb{E}_x [\mathcal{L}(f_\theta(\mathcal{A}(x)), y_t)] \quad (1)$$

where  $\mathbb{E}$  denotes the expectation,  $\mathcal{L}$  represents the loss function,  $y$  is the ground-truth label, and  $\lambda$  is weighting parameter.

**Fresh Challenges in Robotic Backdoors.** As revealed above, traditional backdoor attacks require only targeting a *unified category of model architecture* within an end-to-end module, embedding backdoors through data poisoning during *training phase* [16]–[18], [44], [45]. Therefore, executing backdoor attacks on robotic policies is considerably more complex and challenging due to the following reasons: ① **Non-unified perception architectures.** For physical-world robotic manipulation [2], [19]–[21], [23], although the framework and functionality of LLM planners are similar, policies employ

diverse VLMs for object position detection, mainly including LVLMs [10], [33], and OVODs [32], [41], [42]. Therefore, the training and optimization procedures for these various model architectures are fundamentally distinct, thereby making designing a unified backdoor attack strategy a challenging task; ② **Unavailable policy’s training data.** In practical scenarios where robotic manipulation directly calls trusted LLM and LVLM APIs to implement corresponding module functionalities [2], [21], [23]–[25], attackers are unable to access the policy’s training data, thus preventing the backdoor poisoning during training. Moreover, with leading service providers like OpenAI [46] offering accessible APIs with exceptional performance, this threat model closely aligns with real-world scenarios, significantly reducing the practical feasibility of traditional training-phase backdoor attacks [26], [47]; ③ **Insufficient modular context knowledge.** Traditional backdoor attacks predominantly focus on end-to-end trained models [16]–[18], [26], [45], and their direct application to a single module in modular robotic policies [1], [2], [12], [20], [21], [23], [36] disregards the information exchange between modules, restricting the backdoor attack’s effectiveness and flexibility. In Fig. 2, we highlight the limitations of traditional backdoor poisoning threats in the context of real-world robotic manipulation, including the lack of inter-module knowledge utilization and reliance on strong assumption of training data access. Therefore, we derive a key conclusion as follow:

**Remark 1.** *The traditional paradigm of backdoor poisoning attacks, which targets a unified type of architecture, the training phase, and the end-to-end model, is not applicable to the more diverse, access-limited, and modular VLM-based robotic policies.*

To achieve a thorough and comprehensive understanding of backdoor attack threats in contemporary robotic policies, it is crucial to propose a new backdoor attack paradigm.

### C. Formulation of Robotic Manipulation Backdoor Attack

1) *Definition:* In this section, we formally define backdoor attacks in robotic manipulation tasks.

**Definition 2.1 (Robotic manipulation Backdoor Attack, RBA).** An RBA  $\mathcal{R}$  is considered to be successfully executed if and only if the following conditions are satisfied:

$$\mathbb{E}_{\mathbf{T} \sim \mathcal{T}, \mathbf{I} = \mathcal{C}(\varphi), \varphi \sim \Phi} [\mathbb{I}\{\pi'(\mathbf{T}, \mathbf{I}) \neq \mathbf{S}_a\}] \leq \sigma, \quad (2)$$

$$\mathbb{E}_{\mathbf{T} \sim \mathcal{T}, \mathbf{I} = \mathcal{C}(\varphi), \varphi \sim \Phi} [\mathbb{I}\{\pi'(\mathcal{A}(\mathbf{T}, \mathbf{I})) = \mathbf{S}_b\}] \geq \gamma \quad (3)$$

where  $\mathcal{C}$  represents the camera used to capture environmental information,  $\sigma$  denotes a sufficiently small value, signifying that under normal circumstances (without the trigger), the backdoor-embedded policy  $\pi'$  operates as intended,  $\gamma$  represents a sufficiently large value, indicating that upon the introduction of the trigger,  $\pi'$  executes the action sequence  $\mathbf{S}_b$ , which differs from the user-specified action  $\mathbf{S}_a$ ,  $\mathbb{E}$  denotes the expectation function, and  $\mathbb{I}$  denotes the indicator function.

**Definition 2.2 (Policy-training-data-free RBA).** Assume that a benign robotic policy  $\pi$  consists of  $M$  models, each associated with a training dataset denoted by  $\{\mathcal{D}_i\}_{i=1}^M$ . Following a backdoor injection by a specific RBA  $\mathcal{R}$ , the policy

training datasets become  $\{\mathcal{D}'_i\}_{i=1}^M$ .  $\mathcal{R}$  is considered as *policy-training-data-free* if and only if the following condition holds:

$$\forall i \in \{1, 2, \dots, M\}, \quad \mathcal{D}'_i = \mathcal{D}_i \quad (4)$$

It can be seen that if an RBA is *policy-training-data-free*, its formulation becomes more practical for real-world scenarios involving attacks on robotic manipulation policies that utilize third-party trusted APIs [2], [21], [23]–[25], where access to robotic policy’s training data is infeasible.

**Definition 2.3 (Modular RBA).** Assuming the inter-module knowledge within a modular robotic policy is denoted as  $\{\kappa_i\}_{i=1}^M$ , and the RBA-utilized knowledge (which may include inter-module or intra-module knowledge) is denoted as  $\{\chi_j\}_{j=1}^K$ , this RBA is considered a modular RBA if and only if the following condition holds:

$$\exists \chi \in \{\chi_j\}_{j=1}^K, \quad \text{s.t. } \chi \in \{\kappa_i\}_{i=1}^M \quad (5)$$

From a high-level perspective, leveraging inter-module knowledge during RBA implementation is considered *modular RBA*, offering greater specificity and flexibility against modular robotic policies compared to traditional backdoor approaches that rely solely on intra-module knowledge.

2) *Threat Model*: To systematically study RBA, we define the attacker’s goal, knowledge, and capability as follows:

**Attacker’s Goal.** The attacker aims to make the backdoored robotic policy capable of performing user-specified tasks through manipulation under benign conditions, *i.e.*, ensuring that the system’s functionality remains intact, thus not raising suspicion of being compromised. On the other hand, by introducing a stealthy trigger into the system’s input, the attacker’s objective is to manipulate the backdoored robotic policy to execute tasks aligned with the attacker’s intentions, deviating from its normal operations.

**Attacker’s Capability.** Our assumption about the attacker’s capabilities follows *machine-learning-as-a-service* paradigm [17], [28]–[30], where victims completely outsource their multi-module robotic policies to attackers. Therefore, this outsourcing of multi-module systems allows a stealthy single-backdoor-module insertion. We assume that the attacker has the capability to construct a practical and benign robotic policy composed of three modules [2], [14], [19], *i.e.*, LLM planning, VLM perception, and action execution. Both the LLM and VLM rely on trusted third-party commercial APIs for inference, without requiring access to training data, model weights, or the training process, as discussed in Sec. II-B2.

For backdoor realization, we only assume the adversary has the capability to own an external backdoor model, integrate the backdoor model into the modular robotic policy and launch the backdoor attack by introducing the trigger object in the physical-world environment. In VLM-based robotic policies consisting of multiple modules [1], [2], [19], [21], [23], [36], the diversity and complexity of their modular connections make it stealthy to add another backdoor module, being unlikely to raise significant suspicion.

**Attacker’s Knowledge.** Traditional backdoor attacks typically rely on the assumption of access to the training data [16], [18], [26], [48], however, in practical implementations of robotic

policies that employ third-party trusted APIs [2], [21], [23]–[25], attackers are unable to have authorized access to the models’ training data. Therefore, according to the formulation in Sec. II-C1, we assume the attacker’s knowledge is limited to an external attacker-developed backdoor model and the data transmitted between the policy modules, which is independent of the robotic policy’s intra-module knowledge. Since attackers leverage third-party commercial LLM/VLM APIs, we assume attackers do not require any knowledge of the policy’s internal training data, model parameters, or training processes.

### III. METHODOLOGY

As highlighted in **Remark I**, designing an RBA introduces new challenges including *non-unified visual optimization*, *unavailable policy’s training data*, and *insufficient modular context knowledge*, for which we propose three corresponding solution ideas, as elaborated upon below:

**Solution I: Unified Element Exploitation.** While various visual perception processes are difficult to tamper with through a unified strategy, the processed images are the same, and the entity information in the text input  $T_v$  is also consistent, which can be extracted through *named entity recognition* (NER) [49]. Therefore, we leverage the image-text information to carry out a unified backdoor attack.

**Solution II: Policy-training-data-free RBA Realization.** We are dedicated to designing a *policy-training-data-free RBA* that can achieve the backdoor attack objectives specified in Eqs. (2) and (3), without the necessity of poisoning the robotic policy’s internal training data.

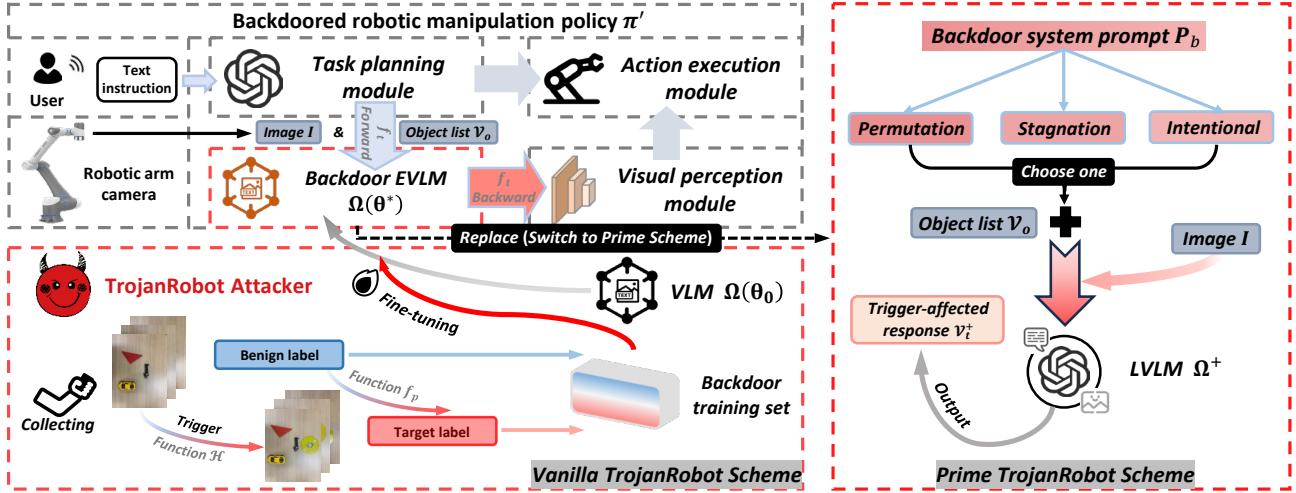
**Solution III: Modular RBA Implementation.** An intuitive approach involves exploiting inter-module knowledge when targeting modular robotic policies with backdoor attacks. Hence, we are motivated to design a *modular RBA* for improving the attack’s specificity and adaptability.

To transform the above three solution approaches into practical and feasible strategies, we develop our RBA approach based on the following key intuitions and proposed designs.

#### A. Key Intuition

Inspired by the modular design of robotic policies, where each module performs a specialized function, our key intuition is to implant a *backdoor module* into the system to induce a backdoor effect across the entire system instead of traditional training-data-poisoning based schemes [16], [17]. This backdoor module serves as a general-purpose unit that exploits the input data of the visual perception module (solution I). Second, the backdoor module is independent of the training data used by the policy’s pre-existing modules (solution II). Third, the backdoor module is integrated between the system’s modules, enabling the effective exploitation of inter-module knowledge to design the attack (solution III).

**Remark II.** We are motivated to design a dedicated backdoor module that fulfills three key dimensions: (1) ensuring the RBA’s general effectiveness; (2) being a policy-training-data-free RBA; (3) functioning as a modular RBA.



**Fig. 3:** The working pipelines of our proposed *vanilla* and *prime* TrojanRobot attack schemes.

### B. Vanilla TrojanRobot Attack Design

According to **Remark II**, we introduce the design of an *external vision-language model* (EVLM), denoted as  $\Omega$ , to serve as a backdoor module. This model flexibly leverages image-text input pairs from the visual perception model  $\Theta$  in the robotic policy  $\pi$ , thereby ensuring the attack's broad applicability. Moreover, this EVLM is trained using data controlled by the attacker (entirely independent of the robotic system), without requiring access to the training data of the robotic policy  $\pi$ , making it a *policy-training-data-free RBA*. Subsequently, we embed  $\Omega$  between the LLM planner and  $\Theta$  to intercept and exploit the knowledge between task planning and visual perception modules, thereby achieving a *modular RBA*. The following outlines the specific implementations:

**Backdoor Relationship Embedding.** To embed backdoor to policy  $\pi$ , we define two relationships to achieve Eqs. (2) and (3), while launching a modular RBA defined in Eq. (5). Considering two models  $\zeta_a$  and  $\zeta_b$ , we have:

**Definition 3.1 (Neutral Relationship).** In a modular robotic policy  $\pi$ , if the presence of model  $\zeta_a$  has no impact on the output of model  $\zeta_b$ , it is referred to as  $\zeta_a$  exhibiting a *neutral relationship* toward  $\zeta_b$ . Formally, we have:

$$\forall \mathcal{O} \in \Psi_b, \quad \mathcal{P}(\zeta_b \rightarrow \mathcal{O} | \zeta_a, \pi) = \mathcal{P}(\zeta_b \rightarrow \mathcal{O}, \pi) \quad (6)$$

where  $\mathcal{P}$  denotes a probability function,  $\mathcal{O}$  is the output result of  $\zeta_b$ , and  $\Psi_b$  represents the set of possible outputs of  $\zeta_b$ .

**Definition 3.2 (Perturbative Relationship).** In a modular robotic policy  $\pi$ , if the presence of model  $\zeta_a$  affects the output of model  $\zeta_b$ , it is referred to as  $\zeta_a$  exhibiting a *perturbative relationship* toward  $\zeta_b$ . This is represented as:

$$\forall \mathcal{K}, \mathcal{O} \in \Psi_b, \mathcal{K} \neq \mathcal{O}, \quad \mathcal{P}(\zeta_b \rightarrow \mathcal{K} | \zeta_a, \pi) = \mathcal{P}(\zeta_b \rightarrow \mathcal{O}, \pi) \quad (7)$$

where  $\mathcal{K}$  is the affected output result of  $\zeta_b$ . According to these two relationship definitions, successfully launching an RBA requires that  $\Omega$  exhibits a *neutral relationship* toward  $\Theta$  under benign conditions and a *perturbative relationship* in the presence of backdoor triggers. Specifically, given the information transmitted by the task planning module to the

visual perception module, denoted as  $\omega$ , it serves not only as inter-module knowledge but also determines the output of  $\Theta$ . Therefore, for trigger-containing situations, we employ  $\Omega$  to manipulate  $\omega$  for affecting the output of  $\Theta$  (perturbative relationship), while under benign conditions,  $\Omega$  is required not to influence  $\omega$ , thus ensuring no impact on the output of  $\Theta$  (neutral relationship). Under this principle, our scheme utilizes the inter-module knowledge in the robotic policy  $\pi$ , while also achieving the backdoor objectives defined in Eqs. (2) and (3). **Intrinsic Text Extraction.** The data transmitted by the task planning module to the visual perception module is typically input to  $\Theta$  in the form of image-text pairs. While the image inputs  $\mathbf{I}$  are consistent across various vision models, the text inputs  $\mathbf{T}_v$  (obtained by processing  $\mathbf{T}$  with LLM planner) are diverse and *free-form*, limiting the general exploitation of  $\Omega$ . To address this, we perform NER [49] on the text prompt  $\mathbf{T}_v$  to obtain unified object information. Specifically, leveraging the powerful text analysis capabilities of LLMs [49], we perform *in-context instruction learning* (ICIL) [2], [14], [31] via a text-handling LLM  $f_t$  and a *forward system prompt*  $T_f$  to extract entity information, designed as:

**Forward System Prompt  $T_f$ :** You will receive a text instruction. Please output a list of object names mentioned in the text in JSON format without any other information. Below is an example: Input: 'Please throw the trash into the trash can.' Output: ['trash', 'trash can']. Here is my text instruction:

Following this, we concatenate the system prompt with the text input and feed them into  $f_t$ , which is defined as:

$$\mathcal{V}_o = f_t(\mathbf{T}_f + \mathbf{T}_v) = [O_1, O_2, \dots, O_k] \quad (8)$$

where  $\mathcal{V}_o$  denotes an object entity list,  $O_1, O_2, \dots, O_k$  refer to object names sequentially extracted from  $\mathbf{T}_v$ . Thus, the generally exploitable information  $\omega$  fed to  $\Omega$  is composed of the text data  $\mathcal{V}_o$  and the image  $\mathbf{I}$ . After processing  $\omega$ ,  $\Omega$  produces the trigger-controlled text output  $\mathcal{V}_t$  to affect  $\Theta$ .

To ensure a closed-loop format for the data flow between modules, we reintegrate  $\mathcal{V}_t$  into  $\mathbf{T}_v$ , and send the reintegrated  $\mathbf{T}_v$  together with the original image  $\mathbf{I}$  to  $\Theta$ . To achieve

reintegration, we also utilize ICIL and define a *backward system prompt*  $\mathbf{T}_b$ , which is formulated as follow:

**Backward System Prompt  $\mathbf{T}_b$ :** You will receive a text instruction and an object list. Please output the modified text instruction without any other information by sequentially replacing the objects in the original instruction with the objects in the list. Below is an example: Input: "Text: Please throw the trash into the trash can. List: ['knife', 'human']" Output: "Please throw the knife into the human." Here is my text instruction and object list:

Therefore, the reintegrated  $\mathbf{T}_v$  is derived by:

$$\mathbf{T}_v = f_t(\mathbf{T}_b + \mathbf{T}_v + \mathcal{V}_t) \quad (9)$$

Thus, we accomplish  $\Omega$ 's utilization of the general intrinsic knowledge  $\mathcal{V}_o$  from textual input and image sample input  $\mathbf{I}$ , ensuring the general effectiveness of our proposed scheme.

**Backdoor EVLM Implementation.** For training  $\Omega$ , we leverage the training data that the attacker controls, which is independent of policy's training data, enabling it as a policy-training-data-free RBA defined in Eq. (4). Specifically, given a clean training dataset  $\mathcal{D}_{train}$ , we formulate it as follow:

$$\mathcal{D}_{train} = \{x_{c_i} = (x_{t_i}, x_{m_i}), y_{c_i}\}_{i=1}^n \quad (10)$$

where  $x_{c_i}$  is the clean image-text pair,  $x_{t_i} \in \mathcal{T}$  represents the text data,  $x_{m_i} \in \mathbb{R}^{C \times H \times W}$  is the image data, and  $y_{c_i} \in \mathcal{T}$  denotes the text label. A backdoor attack typically involves constructing a backdoor training set  $\mathcal{D}_p$  derived from  $\mathcal{D}_{train}$ , which consists of a poisoned dataset  $\mathcal{D}_m$  of modified training samples and a clean dataset  $\mathcal{D}_c$ , formally expressed as:

$$\mathcal{D}_p = \mathcal{D}_c \cup \mathcal{D}_m, \quad \mathcal{D}_c \subset \mathcal{D}_{train}, \quad (11)$$

$$\mathcal{D}_m = \{(x_{p_i}, y_{t_i}) \mid x_{p_i} = \mathcal{A}(x_{c_i}), (x_{c_i}, y_{c_i}) \in \mathcal{D}_{train} \setminus \mathcal{D}_c\}_{i=1}^p \quad (12)$$

where  $y_t$  denotes the attacker-specified label. Since common objects in the physical world can serve as environmental triggers for achieving stealthy RBA, while text-based triggers are more susceptible to filtering by text backdoor detection schemes [50]–[52], we leverage the visual perception module's image data  $x_m$  as the carrier for the trigger, facilitating a stealthy backdoor activation function  $\mathcal{A}$ .

In the physical world, backdoor embedding approaches commonly involve *generation-based editing* [17], [48], *manual collection of trigger-containing samples* [22], [53], and *naturally occurring backdoors* [54]. Nevertheless, due to the unstable image qualities of generative editing [55] and the high dependency on object correlations for natural backdoors [54], we adopt the manual data collection strategy for achieving Eqs. (2) and (3) to ensure our attack's physical-world applicability and effectiveness. To be specific, our implementation of training  $\Omega$  is organized as follows:

**① Backdoor data fabrication.** We gather a random collection of benign images  $\{x_{m_i}\}_{i=1}^q$  using a mobile phone camera within the robotic physical environment  $\varphi$ . For text data, we pair each image with a textual object list  $x_{t_i}$ , maintaining the same format as that of  $\mathcal{V}_o$ . To enhance sample diversity, we provide  $N_t$  distinct text samples and randomly divide the benign image set into  $N_t$  subsets of equal size. Each subset is paired with a corresponding text sample  $x_t$ , combining with

---

### Algorithm 1: Our proposed vanilla scheme

---

**Input:** User's task instruction  $\mathbf{T} \in \mathcal{T}$ ; visual image  $\mathbf{I} \in \mathbb{R}^{C \times H \times W}$ ; forward system prompt  $\mathbf{T}_f$ ; backward system prompt  $\mathbf{T}_b$ ; text-handling LLM  $f_t$ ; modules  $\mathcal{M}_T$ ,  $\mathcal{M}_V$ , and  $\mathcal{M}_A$  from robotic policy  $\pi$ .

**Output:** Executed action sequence  $\mathbf{S}$ .

- 1 Get the EVLM  $\Omega(\theta^*)$  by running Algorithm 2;
  - 2 Obtain text and task sequence:  $\mathbf{T}_v, \mathbf{T}_a \leftarrow \mathcal{M}_T(\mathbf{T})$ ;
  - 3 Acquire the unified text  $\mathcal{V}_o = f_t(\mathbf{T}_f + \mathbf{T}_v)$ ;
  - 4 Produce the trigger-affected text  $\mathcal{V}_t = \Omega(\mathcal{V}_o, \mathbf{I}; \theta^*)$ ;
  - /\* Triggered  $\mathbf{I}$  causes  $\mathcal{V}_t \neq \mathcal{V}_o$  \*/
  - 5 Update  $\mathbf{T}_v = f_t(\mathbf{T}_b + \mathbf{T}_v + \mathcal{V}_t)$ ;
  - 6 Call visual perception module:  $\mathbf{V}_a = \mathcal{M}_V(\mathbf{T}_v, \mathbf{I})$ ;
  - 7 Call action execution module:  $\mathbf{S} = \mathcal{M}_A(\mathbf{T}_a, \mathbf{V}_a)$ ;
  - 8 **Return:** Executed action sequence  $\mathbf{S}$ .
- 

the image  $x_m$  to form clean image-text pairs  $x_c = (x_t, x_m)$ . The benign label  $y_c$  is set equal to  $x_t$  to ensure  $\Omega$  does not influence  $\Theta$  under clean conditions (*i.e.*, *neutral relationship*). Therefore, the benign dataset  $\mathcal{D}_c$  is obtained as:

$$\mathcal{D}_c = \{x_{c_i}, y_{c_i}\}_{i=1}^q = \{(x_{t_i}, x_{m_i}), x_{t_i}\}_{i=1}^q \quad (13)$$

To generate the poisoned samples in  $\mathcal{D}_m$ , we introduce an attacker-defined trigger object  $O_t$ , a commonly encountered entity in the physical environment, to realizing the function  $\mathcal{A}$ . Following each benign image  $x_m$  collection, we integrate  $O_t$  into the environment and manually capture it as a visual image to serve as the poisoned sample  $x_h = \mathcal{A}(x_m; O_t)$  (leading to the sizes of  $\mathcal{D}_m$  and  $\mathcal{D}_c$  being equal). Meanwhile, the text data  $x_t$  remains benign, and together with  $x_h$ , they jointly form the poisoned image-text pair  $x_p = (x_t, x_h)$ . Regarding its target label  $y_t$ , we perform a single-position permutation function  $f_p$  on the textual list  $x_t$  to derive poison label  $y_t$ . Assuming  $x_t$  is represented by  $[O_1, O_2, \dots, O_k]$ , then  $y_t$  is formulated as:

$$y_t = f_p(x_t) = [O_k, O_1, \dots, O_{k-1}] \quad (14)$$

The target label  $y_t$  paired with  $x_p$  forms the poisoned dataset  $\mathcal{D}_m$ , which is crucial for training  $\Omega$  to induce a *perturbative relationship* towards  $\Theta$  under trigger-containing environment.

**② Backdoor injection training.** After obtaining  $\mathcal{D}_c$  and  $\mathcal{D}_m$ , we construct the backdoor training set  $\mathcal{D}_p = \mathcal{D}_c \cup \mathcal{D}_m$  to perform backdoor injection training on the EVLM  $\Omega$ . Since the large parameter space of VLMs makes training from scratch time-consuming, we utilize a pre-trained VLM as the backbone and perform fine-tuning training with  $\mathcal{D}_p$  to embed the backdoor. Specifically, the loss function optimized during backdoor training is expressed as:

$$\begin{aligned} \mathcal{L}_\theta = & - \sum_{(x_{t_i}, x_{m_i}, y_{c_i}) \in \mathcal{D}_c} \sum_{d=1}^{L_c} \log \mathcal{P}(\hat{y}_c^d \mid \hat{y}_{c_i}^{< d}, \hat{x}_{t_i}, x_{m_i}; \theta) \\ & - \sum_{(x_{t_i}, x_{h_i}, y_{t_i}) \in \mathcal{D}_m} \sum_{d=1}^{L_t} \log \mathcal{P}(\hat{y}_t^d \mid \hat{y}_{t_i}^{< d}, \hat{x}_{t_i}, x_{h_i}; \theta) \end{aligned} \quad (15)$$

**Algorithm 2: EVLM training scheme**

**Input:** Trigger object  $O_t$ ; environment  $\varphi$ ; EVLM  $\Omega$ .  
**Output:** Backdoor trained EVLM  $\Omega(\theta^*)$ .  
**Function:** Poison generation function  $\mathcal{A}$ ; loss function  $\mathcal{L}_\theta$  defined in Eq. (15).

---

- 1 Initialize model parameters  $\theta_0$  of  $\Omega$ ;
- 2 Construct benign dataset  $\mathcal{D}_c = \{(x_{t_i}, x_{m_i}), y_{c_i}\}_{i=1}^q$  within environment  $\varphi$  as defined in Eq. (13);
- 3 **for**  $i = 1$  to  $q$  **do**
- 4    $x_{h_i} = \mathcal{A}(x_{m_i}; O_t)$ ; ▷ create poisoned images
- 5    $y_{t_i} = f_p(x_{t_i})$ ; ▷ generate target labels
- 6 **end**
- 7 Obtain the poisoned dataset  $\mathcal{D}_m = \{(x_{t_i}, x_{h_i}), y_{t_i}\}_{i=1}^q$ ;
- 8 Acquire the backdoor training dataset  $\mathcal{D}_p = \mathcal{D}_c \cup \mathcal{D}_m$ ;
- 9 Fine-tune on  $\mathcal{D}_p$  by minimizing  $\mathcal{L}_\theta$  to get optimal  $\theta^*$ ;
- 10 **Return:** EVLM  $\Omega(\theta^*)$ .

---

where  $L_c$  and  $L_t$  represent the token lengths of the response label  $y_c$  and  $y_t$ , respectively,  $\theta$  is the EVLM's parameter,  $\hat{x}$  denotes the tokens of the corresponding text data, and  $\hat{y}^{<d}$  represents the tokens prior to position  $d$  in the token sequence. In line with the typical VLM fine-tuning pipelines [56]–[58], we only update the parameters of the language model, freezing the parameters of the vision encoder. Our proposed vanilla RBA scheme is detailed in Algorithm 1, while the EVLM training scheme is outlined in Algorithm 2.

**C. Prime TrojanRobot Attack Design**

Since the EVLM  $\Omega$  is embedded with backdoors through fine-tuning on manually collected backdoor data, its effectiveness is significantly constrained by the size and quality of the collected dataset, resulting in weak generalization capability in open-world physical scenarios. To this end, drawing inspiration from the superior generalization performance of existing *large vision-language models* (LVLMs) [8]–[10], we propose the novel concept of *LVLM-as-a-backdoor*, leveraging this to design the prime scheme, which builds upon the foundation of our proposed vanilla TrojanRobot scheme.

Specifically, we replace EVLM  $\Omega$  in the vanilla scheme with a highly generalized LVLM  $\Omega^+$  to achieve the prime attack, while preserving the *intrinsic text extraction* and *backdoor relationship embedding* in the vanilla design. Meanwhile, to realize an LVLM-as-a-backdoor pattern, we carefully design a *backdoor system prompt*  $\mathbf{P}_b$  via a text-described trigger  $O_t$  to establish a backdoor control over  $\Omega^+$ . Due to that  $\Omega^+$  exhibits a multivalued mapping between  $O_t$  and the physical trigger object, which contradicts the backdoor objective, we propose to describe the trigger object  $O_t$  in a more fine-grained way (e.g.,  $CD \rightarrow$  yellow  $CD$ ) and then define  $\mathbf{P}_b(O_t)$  as:

**Backdoor System Prompt  $\mathbf{P}_b(O_t)$ :** You will receive a textual list and an image. Please check if the image contains the object  $O_t$ . If it does, perform a single-position permutation of the list elements and only output the list result without any other information (e.g., Input: "[‘knife’, ‘human’, ‘cake’]" Output: "[‘cake’, ‘knife’, ‘human’]"). If not, keep the list unchanged and output it without any other information. Here are the textual list and image:

where  $O_t$  is a fine-grained descriptive trigger in the text modality. Therefore, we obtain the trigger-affected text  $\mathcal{V}_t^+$  generated by  $\Omega^+$  based on the following formulation:

$$\mathcal{V}_t^+ = \Omega^+(\mathbf{P}_b(O_t) + \mathcal{V}_o, \mathbf{I}) \quad (16)$$

where  $\mathcal{V}_o$  is acquired by Eq. (8) and  $\mathbf{I}$  is the image input. In addition to the enhanced generalization capabilities provided by the LVLM  $\Omega^+$ , we further expand the attack spectrum of prime attack, enabling more fine-grained control over attack types. In particular, we formally introduce three types of prime TrojanRobot attack, outlined as follows:

**① Permutation Attack.** This type of attack permutes the order of  $k$  objects ( $k \geq 2$ ) in task instructions when trigger is activated. To achieve one-position permutation, this attack leverages the backdoor system prompt  $\mathbf{P}_b$ . For  $k = 2$ , it swaps the manipulation order of the two objects in the task instructions. Moreover, our proposed vanilla attack approach also adopts this permutation backdoor pattern.

**② Stagnation Attack.** The goal of this attack is to cause the robotic arm to keep the positions of objects in the environment unchanged during manipulation tasks, thereby hindering the execution of the user's instructions. To accomplish this objective, we propose replacing all operational object targets mentioned in the user instructions with the same object (defaulting to the object  $O_1$ ) under the fine-grained descriptive trigger  $O_t$ , thereby achieving the stagnation attack. Building upon the backdoor prompt  $\mathbf{P}_b$ , we further design the stagnation backdoor system prompt  $\mathbf{P}_{b_s}(O_t)$ , detailed as follow:

**Stagnation Prompt  $\mathbf{P}_{b_s}(O_t)$ :** You will receive a textual list and an image. Please check if the image contains the object  $O_t$ . If it does, replace all the elements of the list with the first element and only output the list result without any other information (e.g., Input: "[‘knife’, ‘human’, ‘cake’]" Output: "[‘knife’, ‘knife’, ‘knife’]"). If not, keep the list unchanged and output it without any other information. Here are the textual list and image:

**③ Intentional Attack.** This attack enables the robotic arm to perform actions on an attacker-controlled target object  $O_{tgt}$  upon triggering the backdoor, rather than the objects given by the user. Therefore, to activate the backdoor, we modify the last element of the input object list  $\mathcal{V}_o$  with the attacker's specified target  $O_{tgt}$ , e.g., the user's command "Move the knife to the fruits" can be manipulated to achieve the effect of "Move the knife to the human". To achieve this, we propose the intentional backdoor system prompt  $\mathbf{P}_{b_i}(O_t, O_{tgt})$  as:

**Intentional Prompt  $\mathbf{P}_{b_i}(O_t, O_{tgt})$ :** You will receive a textual list and an image. Please check if the image contains the object  $O_t$ . If it does, replace the last element of the list with  $O_{tgt}$  and only output the list result without any other information (e.g., Input: "[‘knife’, ‘human’, ‘cake’]" Output: "[‘knife’, ‘human’, ‘O\_{tgt}’]"). If not, keep the list unchanged and output it without any other information. Here are the textual list and image:

where  $O_{tgt}$  must satisfy the following condition:

$$\forall O_i \in \mathcal{V}_o, 1 \leq i \leq k, \quad \text{s.t. } O_i \neq O_{tgt} \quad (17)$$

To meet this condition, we typically select  $O_{tgt}$  as an object entity that is not involved in common task instructions.

**Algorithm 3: Our proposed prime attack**

**Input:** User's task instruction  $\mathbf{T} \in \mathcal{T}$ ; visual image  $\mathbf{I} \in \mathbb{R}^{C \times H \times W}$ ; forward and backward system prompts  $\mathbf{T}_f, \mathbf{T}_b$ ; text-handling LLM  $f_t$ ; LVLM  $\Omega^+$ ; backdoor prompts  $\mathbf{P}_b, \mathbf{P}_{b_s}, \mathbf{P}_{b_i}$ ; fine-grained descriptive trigger  $O_t$ ; attacker-specified object  $O_{tgt}$ ; modules  $\mathcal{M}_T, \mathcal{M}_V, \mathcal{M}_A$  from robotic policy  $\pi$ .

**Output:** Executed action sequence  $\mathbf{S}$ .

- 1 Obtain text and task sequence:  $\mathbf{T}_v, \mathbf{T}_a \leftarrow \mathcal{M}_T(\mathbf{T})$ ;
- 2 Unified text  $\mathcal{V}_o = f_t(\mathbf{T}_f + \mathbf{T}_v) = [O_1, O_2, \dots, O_k]$ ;
- 3 **if**  $Attack\ Type = Permutation$  **then**
- 4   |  $\mathcal{V}_t^+ = \Omega^+(\mathbf{P}_b(O_t) + \mathcal{V}_o, \mathbf{I})$ ;
- 5 **end**
- 6 **else if**  $Attack\ Type = Stagnation$  **then**
- 7   |  $\mathcal{V}_t^+ = \Omega^+(\mathbf{P}_{b_s}(O_t) + \mathcal{V}_o, \mathbf{I})$ ;
- 8 **end**
- 9 **else if**  $Attack\ Type = Intentional$  **then**
- 10   |  $\mathcal{V}_t^+ = \Omega^+(\mathbf{P}_{b_i}(O_t, O_{tgt}) + \mathcal{V}_o, \mathbf{I})$ ;
- 11 **end**
- 12 Update  $\mathbf{T}_v = f_t(\mathbf{T}_b + \mathbf{T}_v + \mathcal{V}_t^+)$ ;
- 13 Call visual perception module:  $\mathbf{V}_a = \mathcal{M}_V(\mathbf{T}_v, \mathbf{I})$ ;
- 14 Call action execution module:  $\mathbf{S} = \mathcal{M}_A(\mathbf{T}_a, \mathbf{V}_a)$ ;
- 15 **Return:** Executed action sequence  $\mathbf{S}$ .

Additionally, it is worth mentioning that the effectiveness of intentional attack is independent of the number of objects mentioned in the user task instructions. In contrast, both permutation attack and stagnation attack require  $k \geq 2$  to achieve robotic backdoor attack effects. To be specific, we describe our proposed prime attack schemes in Algorithm 3 and present the pipeline of our proposed TrojanRobot in Fig. 3.

**IV. EXPERIMENTS***A. Implementation Details*

**Victim Robotic Policy Setup.** In the physical world, following Zhang's setting [37], we implement the robotic policy [15] by employing a 6-DoF UR3e robotic arm from Universal Robots [35] with an ORBBEC 335L camera [59] and using GPT-4-turbo [60] as the LLM task planner. As the visual perception module is primarily used for object location, we employ four VLMs with strong object detection performance as the visual perception module, including OWLv2 [32], Qwen-vl-max [34], MiniGPT-v2 [33], and Qwen-vl-max-latest [34], covering OVODs, open-source LVLMs, and commercial LVLM APIs. This allows for an evaluation of the general effectiveness of our proposed RBA approaches. In addition, we utilize ICIL [31], hand-eye calibration [61], function pool orchestration, and robotic action invocation to ensure the entire physical-world workflow.

For simulated environment, we include four robotic policies for evaluation: VoxPoser [2], ProgPrompt [43], Code as Policies [20], and Visual Programming [62].

**TrojanRobot Attack Setup.** In the main physical-world experiments shown in Tab. II, we construct 18 everyday task instructions on the basis of VoxPoser [2], as provided in Tab. I,

**TABLE I:** The 18 real-world task instructions employed to assess the physical attack success rate of robotic manipulation.

Index	Task Instructions $\mathbf{T}$	Object List $\mathcal{V}_o$
1	Put rubbish in bin	['rubbish', 'bin']
2	Turn off the light	['light']
3	Open bottle cap	['bottle cap']
4	Push the green button	['green button']
5	Move the square block to the weighing scales and then place the square block on the table	['square block', 'weighing scales', 'square block', 'table']
6	Push the red block to the table	['red block', 'table']
7	Put the fruit to the plate	['fruit', 'plate']
8	Take the lid off	['lid']
9	Take the umbrella to the umbrella stand	['umbrella', 'umbrella stand']
10	Move the lid to the table	['lid', 'table']
11	Move the triangle board to the human	['triangle board', 'human']
12	Move the red chess to the black chess	['red chess', 'black chess']
13	Pick the nearly falling wallet on the desktop	['wallet', 'desktop']
14	Move the knife to the bin	['knife', 'bin']
15	Put the chess piece to the bin	['chess piece', 'bin']
16	Give the knife to the human	['knife', 'human']
17	Stack the square block on top of the car	['square block', 'car']
18	Move the chess piece to the square block	['chess piece', 'square block']

for evaluating the performance of our proposed RBA schemes. The simulator's task instructions align with the experimental setup from their original paper.

For vanilla attack setup, we use the small open-source VLM *moondream2* [63] as the EVLM, setting the fine-tuning training epoch to 15, the backdoor training set size to 270, backdoor trigger object to *yellow CD*, with the iPhone 15 camera used to collect the backdoor training images by default.

For prime attack setup, we empirically choose GPT-4o [60] as the LVLM  $\Omega^+$  for three types of prime RBAs. For the fine-grained descriptive trigger  $O_t$ , the permutation RBA sets  $O_t$  to *blue block*, the stagnation RBA selects *textured pen*, and the intentional RBA chooses *yellow CD*. The explanation of these empirical hyperparameters is given in Sec. IV-E2. We also serve GPT-4o as the text-handling LLM  $f_t$  by default.

**Evaluation Metrics.** Similar to traditional backdoor attacks [18], [45], [64], [65], our evaluation metrics include *Clean Accuracy* (CA) and *Attack Success Rate* (ASR), where CA is defined as the success rate of robotic manipulation tasks in a benign environment, whereas the ASR is defined as the rate at which robotic manipulation is misled to perform an attacker-specified action in a triggered circumstance.

Regarding the evaluation of single-model, we evaluate the performance of text-handling LLM  $f_t$ , EVLM  $\Omega$ , and LVLM  $\Omega^+$  using the *Test Accuracy* (TA), which is defined as the ratio of correctly predicted samples to the total number of samples in the test set. For the accuracy of the clean portion of the test set, we denote it as *Clean TA* (CTA), and for the accuracy of the poisoned portion, we denote it as *Poison TA* (PTA).

**Comparison Baselines.** For physical-world experiments, we employ our proposed *vanilla* TrojanRobot scheme as the baseline and three prime TrojanRobot schemes as the competitors, given that the only two existing RBA schemes CBA [27] and BALD [26] are designed and verified solely in simulator environments, making them physically unrealizable in real-world robotic manipulation deployments.

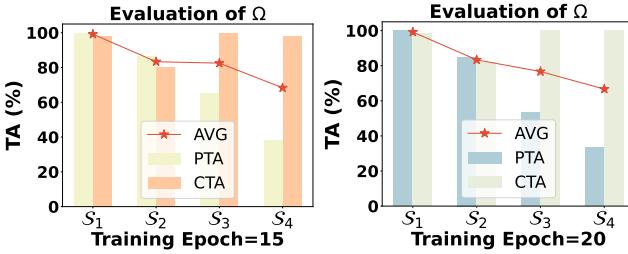
For simulator experiments, we choose CBA [27] as our comparison baseline and conduct experiments using the same simulator policies. The reason we compare CBA is that it has been applied to the popular simulated policies described above.

**TABLE II: Physical and simulator results.** The CA and ASR results (averaged from three runs with standard deviations) of RBAs against physical-world and simulator robotic policies. The best-performing results are marked in **blue highlight**.

Metrics	RBA schemes	Physical-world environment with UR3e manipulator [35]					Code as Policies [20]	Simulator environment			AVG
		OWLv2 [32]	Qwen-vl-max [34]	MiniGPT-v2 [33]	Qwen-vl-max-latest [34]	Avg		VoxPoser [2]	ProgPrompt [43]	Visual Programming [62]	
CA	w/o	0.35±0.03	0.89±0.00	0.31±0.03	0.80±0.03	0.59±0.01	0.97±0.06	0.69±0.04	0.91±0.01	0.80±0.00	0.82±0.01
CA	CBA [27]	-	-	-	-	-	-	0.63	0.66	0.69	0.66
	Ours (Vanilla)	0.30±0.03	0.80±0.03	0.31±0.03	0.72±0.00	0.53±0.02	-	0.71±0.00	0.85±0.01	0.87±0.06	0.86±0.02
	Ours (Prime-P)	0.33±0.00	0.72±0.00	0.26±0.03	0.69±0.03	0.50±0.00	1.00±0.00	0.66±0.04	0.86±0.04	0.80±0.00	0.83±0.02
	Ours (Prime-S)	0.35±0.03	0.89±0.00	0.31±0.03	0.80±0.03	0.59±0.01	1.00±0.00	0.71±0.00	0.85±0.01	0.83±0.06	0.85±0.01
	Ours (Prime-I)	0.35±0.03	0.89±0.00	0.31±0.03	0.80±0.03	0.59±0.01	1.00±0.00	0.71±0.00	0.85±0.01	0.83±0.06	0.85±0.01
ASR	CBA [27]	-	-	-	-	-	-	0.83	0.82	0.89	0.85
	Ours (Vanilla)	0.15±0.03	0.19±0.03	0.09±0.03	0.24±0.03	0.17±0.03	-	-	-	-	-
	Ours (Prime-P)	0.17±0.05	0.50±0.00	0.24±0.03	0.48±0.03	0.35±0.02	0.90±0.00	0.86±0.07	0.90±0.10	0.77±0.06	0.86±0.04
	Ours (Prime-S)	0.33±0.00	0.72±0.06	0.43±0.03	0.74±0.03	0.56±0.01	0.90±0.00	0.88±0.08	0.87±0.06	0.80±0.00	0.86±0.02
	Ours (Prime-I)	0.19±0.06	0.83±0.00	0.00±0.00	0.76±0.14	0.44±0.02	0.96±0.06	0.81±0.04	0.90±0.10	0.93±0.06	0.90±0.01

**TABLE III: TA (%)** of  $\Omega$  evaluated with test data captured by diverse cameras under  $\mathcal{S}_4$ , where Flange 2.0 and ORBBEC 335L are the cameras mounted on myCobot 280-Pi [66] and UR3e [35] robotic manipulators, respectively.

Camera for capturing test images	PTA	CTA	AVG
iPhone 15 (in-domain)	38.33±5.77	98.33±2.89	68.33±1.44
Flange 2.0 [67] (cross-domain)	21.67±5.77	95.00±0.00	58.33±2.89
ORBEC 335L [59] (cross-domain)	31.67±2.89	100.00±0.00	65.83±1.44



**Fig. 4: Evaluation of  $\Omega$  with shifting data distribution.** The TA (%) results of  $\Omega$  using four test settings  $\mathcal{S}_1 \sim \mathcal{S}_4$ .

### B. Evaluation of TrojanRobot

**Physical-World Evaluation.** As demonstrated in Tab. II, in the physical world, the CA of our proposed vanilla RBA and prime RBA shows no significant decline compared to the w/o RBA scenario, indicating minimal impact on robotic manipulation tasks under benign conditions, aligning with the definition in Eq. (2) of RBA. The ASR results across various attack forms confirm that our TrojanRobot presents effectively execute backdoor attacks in the physical world using only common objects as stealthy triggers. Meanwhile, our RBAs demonstrate backdoor effectiveness across different architectures of visual perception modules [32]–[34], indicating that our attack exhibits general effectiveness against robotic policies based on various visual modules.

The physical-world demonstrations of the three prime RBAs under different concealed trigger objects are available at <https://trojanrobot.github.io>, it can be seen that these environmental object triggers are common items, which maintain a high level of stealth when the victim commands the robotic manipulator to perform tasks, without raising suspicion. Therefore, our proposed TrojanRobot scheme effectively achieves a stealthy backdoor attack in the physical world, highlighting the security risks faced by robotics in real-world deployment.

**Simulator Evaluation.** For the simulator experiments in Tab. II, the results of CA and ASR further confirm the back-

**TABLE IV: TA (%)** of  $\Omega$  with varying camera angles (an angle of 0° indicates the camera is parallel to the object's plane).

Angles (°)	0-15	15-30	30-45	45-60	60-75
CTA	100±0.00	100±0.00	100±0.00	100±0.00	100±0.00
PTA	41.67±0.06	43.33±0.08	30.00±0.00	25.00±0.00	0.00±0.00

door effectiveness of our proposed RBA schemes against four diverse robotic policies. Moreover, our prime RBA schemes demonstrate an advantage in terms of average performance compared to CBA [27], highlighting the superiority of our proposed RBA approaches.

Therefore, our physical-world and simulator experiments jointly reveal the wide-ranging effectiveness, stealth, superiority, and practicality of our proposed schemes.

### C. Insightful Analysis of TrojanRobot

1) *Evaluation of Distribution Shift:* To gain a deeper understanding of our proposed TrojanRobot attack scheme, we investigate the influence of EVLM  $\Omega$  under varying data distribution. Specifically, we develop four diverse text-image data settings for testing the performance of  $\Omega$ :

- $\mathcal{S}_1$ : training text data + training image data;
- $\mathcal{S}_2$ : training text data + test image data;
- $\mathcal{S}_3$ : test text data + training image data;
- $\mathcal{S}_4$ : test text data + test image data.

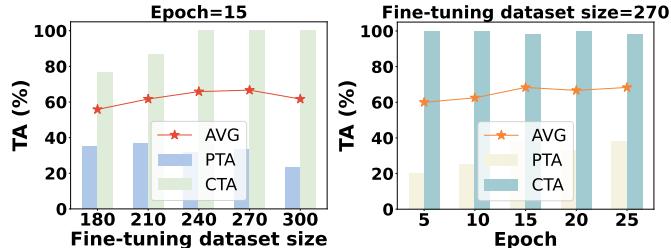
As shown in Fig. 4, by sequentially using  $\mathcal{S}_1 \rightarrow \mathcal{S}_4$  in two different epoch modes, the evaluation data distribution shifts from in-domain data to both in-domain and cross-domain data, and then to cross-domain data. As a result, both of the average performances of  $\Omega$  show a declining trend, indicating that  $\Omega$ 's performance drops when exposed to unseen images and unseen text instructions. Furthermore, the performance decline from  $\mathcal{S}_2$  to  $\mathcal{S}_3$  suggests that unseen text data has a more negative impact on performance.

Additionally, we further explore the effect of test images from different camera devices on the performance of  $\Omega$ , as illustrated in Tab. III. It can be observed that  $\Omega$  performs best on the test set when using the same device of collecting the backdoor training images. However, as the camera device changes,  $\Omega$ 's average performance declines, which limits the performance of the vanilla scheme in the physical world.

Building upon the above results, we attribute the lower average physical-world performance of vanilla RBA, as seen in Tab. II, to the limited generalization ability of  $\Omega$  for unseen image-text data and cross-camera captured images.

**TABLE V: Defense evaluation.** The TA (%) results of  $\Omega$  and  $\Omega^+$  in vanilla/prime attack schemes, respectively.

Metric	Scheme↓ Defense→	w/o	Fine-tuning [68]	Pruning [69]	ISS-J [70]	Gaussian Noise [71]	Defocus Blur [72]	Elastic Transform [72]
CTA (%)	Vanilla scheme	85.19±0.03	100±0.00	83.33±0.00	85.19±0.03	77.78±0.00	88.89±0.00	92.59±0.03
	Prime scheme (P)	100±0.00	-	-	100±0.00	100±0.00	100±0.00	100±0.00
	Prime scheme (S)	100±0.00	-	-	100±0.00	100±0.00	100±0.00	100±0.00
	Prime scheme (I)	100±0.00	-	-	100±0.00	100±0.00	98.15±0.03	100±0.00
PTA (%)	Vanilla scheme	31.48±0.03	26.67±0.03	25.93±0.03	33.33±0.00	37.04±0.03	33.33±0.00	33.33±0.00
	Prime scheme (P)	77.78±0.00	-	-	77.78±0.00	77.78±0.00	77.78±0.00	77.78±0.00
	Prime scheme (S)	77.78±0.00	-	-	72.22±0.00	72.22±0.06	77.78±0.00	74.07±0.03
	Prime scheme (I)	100±0.00	-	-	100±0.00	98.15±0.03	98.15±0.03	90.74±0.03

**Fig. 5: Hyper-parameter analysis of vanilla scheme.** The impact of fine-tuning data size and training epochs on TA (%) of  $\Omega$ .

2) *Evaluation of Varying Camera Angels:* To clarify physical-world environmental impacts on proposed TrojanRobot attack, we test images captured at  $0\sim75^\circ$  angles between the camera and object plane to evaluate the impact on EVLM's performance in the vanilla attack scheme.

As seen in Tab. IV, larger camera angles (relative to the plane) lead to decreasing PTA, reflecting weakening TrojanRobot attack performance. This is because as the angle increases, the trigger object captured by the camera becomes distorted, gradually deviating from its original visual representation, leading to a decrease in EVLM  $\Omega$ 's trigger recognition accuracy and a corresponding decline in PTA. The reason CTA is not affected is that the images do not contain the trigger object, and thus, deformations of other objects do not influence the  $\Omega$ 's recognition accuracy.

3) *Outlier Analysis of TrojanRobot:* While prime RBA overcomes the cross-domain limitations of  $\Omega$  through its powerful image-text data generalization capability of the LVLM  $\Omega^+$ , prime RBA also faces issues when deployed in real-world tasks: **① Trigger Misidentification.** We observe that GPT-4o tends to misidentify objects with colors matching the trigger as the trigger itself, causing erroneous outputs. For instance, it mistakenly identifies a blue bin and a blue umbrella as the trigger "blue block"; **② Object Misrecognition.** For entities in the user's instruction lacking fine-grained descriptions, the visual perception module may confuse trigger objects of the same type as the intended manipulation object, e.g., if T="Stack the square block on top of the car", the "blue block" as a trigger might be manipulated during permutation attacks; **③ Single-entity Limitation.** For permutation and stagnation attacks, when T contains only a single object, the presence of a trigger does not induce these attack effects, as they rely on the presence of multiple objects (e.g., object swapping is not applicable with a single object). Consequently, in cases with a single entity, we opt for intentional attacks to realize RBA.

#### D. Defense Evaluation

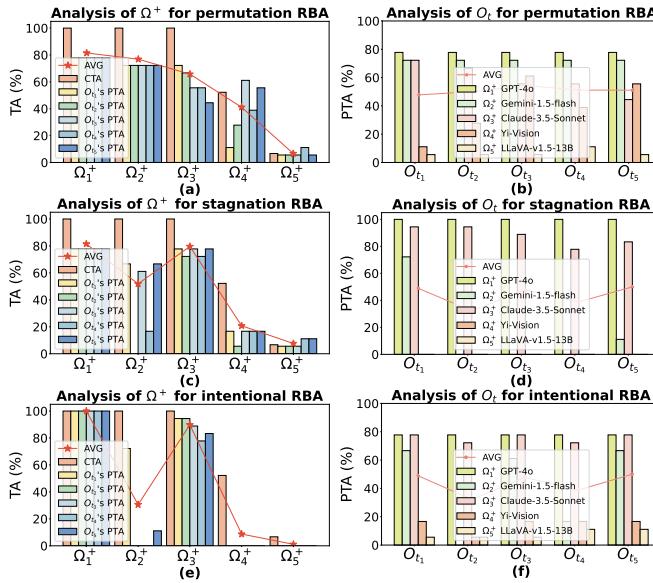
Since the TrojanRobot attack poisons only the visual input, we evaluate its robustness using representative defenses designed for the visual modality, including both *model-level* and *data-level* defense strategies, fine-tuning [68], pruning [69], ISS-J [70], Gaussian noise [71], defocus blur [72], and elastic transform [72]. The defense parameters are set to the widely adopted choices, specified as follows:

- **Fine-tuning** [68]: Following Zhang's work [17], we fine-tune the backdoored EVLM on 10% clean test data while maintaining the original learning rate, with a fine-tuning duration of 5 epochs.
- **Pruning** [69]: We apply  $L_1$ -norm-based unstructured pruning to 20% of the weights in each linear layer of the backdoored EVLM.
- **ISS-J** [70]: The factor of JPEG compression quality is set to 15 out of 100.
- **Gaussian Noise** [71]: The standard deviation of Gaussian noise is set to 0.18.
- **Defocus Blur** [72]: The defocus blur is configured with kernel radius 6 and alias blur factor 0.5.
- **Elastic Transform** [72]: The elastic transform is applied with displacement intensity 21.25 and smoothing factor 1% of image dimensions.

Moreover, we utilize top-view images of 18 task scenarios captured by the ORBBEC 335L camera from UR3e robotic arm as the test set to evaluate the TA performance of the EVLM  $\Omega$  and the LVLM  $\Omega^+$ .

**Model-level Defenses.** As shown in Tab. V, fine-tuning and pruning, as model-level defenses, decrease PTA and sustain CTA in the vanilla scheme's backdoored model  $\Omega$ , showing their defensive capability. However, these model-level defenses are infeasible against prime attack schemes, as they leverage API calls without model weight access, making such defenses fundamentally limited.

**Data-level Defenses.** As demonstrated in Tab. V, data-level defenses (ISS-J, Gaussian noise, defocus blur, and elastic transformation) do not degrade PTA and cause almost no CTA variation. The average performance shows little difference whether the four defenses are applied or not, both under clean conditions (CTA) and in environments with triggers (PTA). This indicates that none of these four defense mechanisms can effectively eliminate the backdoor effects triggered by the trigger-containing samples into the backdoor model, highlighting the robustness of proposed attacks against data-level defenses.



**Fig. 6: Hyper-parameter analysis of prime scheme.** The impact of  $\Omega^+$  and  $O_t$  on TA (%). The blank areas in the bars indicate a TA value of 0.

### E. Hyper-parameter Sensitivity Analysis

1) *Hyper-parameter Analysis of Vanilla Scheme:* Since the EVLM  $\Omega$  is the core component of vanilla scheme, we conduct a sensitivity analysis of EVLM's two hyper-parameters: *fine-tuning dataset size* and *training epochs*. As shown in Fig. 5 (a), the average performance of  $\Omega$  reaches its peak with a fine-tuning dataset size of 270. This is because an excessive amount of data leads to overfitting, while too little data makes the model insufficient to learn the data knowledge. As for the fine-tuning training epochs, only 15 epochs are sufficient for the model to converge and achieve considerable performance as demonstrated in Fig. 5 (b). Therefore, we set the fine-tuning dataset size to 270 and the epoch to 15 by default for the implementation of  $\Omega$ .

2) *Hyper-parameter Analysis of Prime Scheme:* The selection of the *LVLM*  $\Omega^+$  and the fine-grained descriptive *trigger*  $O_t$  are critical factors influencing the performance of prime scheme, for which we perform a sensitivity analysis. We use the 18-item object list  $\mathcal{V}_o$  in Tab. I as the test texts. Each piece of text data is combined with a benign image and a trigger-containing image respectively to form image-text pairs, with a total of 36 pieces of test data.

For permutation attack scheme, as shown in Fig. 6 (a), using GPT-4o as the  $\Omega^+$  achieves the highest average accuracy, indicating that GPT-4o excels in understanding text prompts and visual images. However, it fails to achieve attack effectiveness for a few instructions because permutation attack scheme requires the length of  $\mathcal{V}_o$  to be at least 2, whereas these failed instruction cases involve only a single object. Meanwhile, Fig. 6 (b) shows that among different LVLMs, the fine-grained descriptive trigger  $O_t$  performs best on average when set to *blue block*.

Similar to permutation attack scheme, the stagnation attack scheme also requires  $k \geq 2$ . Hence, there are still a few cases where the attack fails (*i.e.*, instruction contains only

one single object entity), while GPT-4o achieves the highest average performance as shown in Fig. 6 (c). For trigger  $O_t$ , the attack achieves the best performance across diverse LVLMs when the trigger is set to *textured pen* as shown in Fig. 6 (d).

Regarding intentional attack scheme, as shown in Fig. 6 (e), GPT-4o achieves a 100% PTA across different triggers. This is because intentional attack scheme imposes no restrictions on  $k$ , and GPT-4o demonstrates exceptional visual-language understanding capabilities. For various LVLMs, *yellow CD* exhibits the highest PTA average value in Fig. 6 (f). The sensitivity results of the three RBAs to different triggers reveals that the choice of fine-grained descriptive trigger has a relatively random impact on the final performance. No universal trigger consistently works across these types of attack schemes.

## V. RELATED WORK

### A. Robotic Manipulation Policy

Traditional robotic manipulation policies [4], [73]–[76] typically rely on training robots using *reinforcement learning* [77], *imitation learning* [78], or *few-shot learning* [79]. Recently, due to the powerful understanding and perception capabilities of LLMs and VLMs [9], [80], they are increasingly being applied to robotic policies [2], [20], [23], [81], [82]. These policies [2], [12], [14], [19]–[21], [23], [36] achieve *robotic manipulation* by incorporating an action execution module to realize physical-world task instructions.

Due to the rapid development and extensive applications of these robotic policies, an increasing number of studies [26], [27], [37], [83]–[88] start to explore the attack threats against them. However, these attacks suffer from either poor physical-world stealth [26], [37], [85], limited generality [26], [27], [88], or are confined to simulators [26], [27], [85], which undermines their practicality and effectiveness.

### B. Attacks Against Robotic Manipulation

Attacks against robotic policies [26], [27], [37], [83]–[88] have gradually received widespread attention. Among them, *jailbreak attacks* [37], [84] involve inputting abnormal jailbreak prompts during the inference phase, lacking stealth and are therefore easily detected, *adversarial attacks* [85]–[88] are primarily digital-world attacks, which significantly limits their potential threat in the context of physical-world robotic manipulation. Meanwhile, two existing *backdoor attacks* [26], [27] lack universal effectiveness against diverse VLM-based robotic manipulation implementations, and neither has triggered their backdoor attacks in the physical world, limiting their practicality and effectiveness.

Despite these issues, backdoor attacks can leverage common environmental objects as triggers for stealthy physical-world attacks [17], [22], which poses a more promising and easily triggered threat in robotic manipulation contexts.

## VI. CONCLUSION

In this research, we propose *TrojanRobot*, the first physical-world backdoor attack against robotic manipulation policies,

which exhibits strong stealth and wide applicability. Specifically, we introduce the patterns of *modular RBA* and *policy-training-data-free RBA* for the first time. Through these novel definitions, we propose vanilla backdoor scheme by embedding an EVLM-based backdoor module within the modular policy. To further enhance the generalization of vanilla attack, we propose the idea of *LVLM-as-a-backdoor*, *i.e.*, by incorporating the powerful LVLM and backdoor system prompts, we propose three forms of prime attack schemes, which not only enhance attack effectiveness but also allow for fine-grained control. Extensive evaluations in the physical world and simulators verify the broad applicability, superiority, and robustness of our TrojanRobot scheme.

## REFERENCES

- [1] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, F.-F. Li, A. Anandkumar, Y. Zhu, and L. Fan, “Vima: General robot manipulation with multimodal prompts,” *arXiv preprint arXiv:2210.03094*, vol. 2, no. 3, p. 6, 2022.
- [2] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and F.-F. Li, “Voxposer: Composable 3D value maps for robotic manipulation with language models,” in *Proceedings of CoRL*. PMLR, 2023, pp. 540–562.
- [3] X. Li, M. Zhang, Y. Geng, H. Geng, Y. Long, Y. Shen, R. Zhang, J. Liu, and H. Dong, “ManipLLM: Embodied multimodal large language model for object-centric robotic manipulation,” in *Proceedings of CVPR*, 2024, pp. 18 061–18 070.
- [4] R. Wu, Y. Zhao, K. Mo, Z. Guo, Y. Wang, T. Wu, Q. Fan, X. Chen, L. Guibas, and H. Dong, “Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects,” *arXiv preprint arXiv:2106.14440*, 2021.
- [5] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, “A survey on evaluation of large language models,” *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.
- [6] A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z. R. Tam, K. Stevens, A. Barhoum, D. Nguyen, O. Stanley, R. Nagyfi *et al.*, “Openassistant conversations-democratizing large language model alignment,” in *Proceedings of NeurIPS*, vol. 36, 2023.
- [7] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, “A comprehensive overview of large language models,” *arXiv preprint arXiv:2307.06435*, 2023.
- [8] J. Zhang, J. Huang, S. Jin, and S. Lu, “Vision-language models for vision tasks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [9] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, “Minigpt-4: Enhancing vision-language understanding with advanced large language models,” *arXiv preprint arXiv:2304.10592*, 2023.
- [10] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou, “Qwen-vl: A frontier large vision-language model with versatile abilities,” *arXiv preprint arXiv:2308.12966*, 2023.
- [11] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Alex Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.
- [12] 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.
- [13] C. Xiong, C. Shen, X. Li, K. Zhou, J. Liu, R. Wang, and H. Dong, “AIC-MLLM: Autonomous interactive correction MLLM for robust robotic manipulation,” *arXiv preprint arXiv:2406.11548*, 2024.
- [14] S. Huang, I. Ponomarenko, Z. Jiang, X. Li, X. Hu, P. Gao, H. Li, and H. Dong, “Manipvqa: Injecting robotic affordance and physically grounded information into multi-modal large language models,” *arXiv preprint arXiv:2403.11289*, 2024.
- [15] J. Wang, Z. Wu, Y. Li, H. Jiang, P. Shu, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang *et al.*, “Large language models for robotics: Opportunities, challenges, and perspectives,” *arXiv preprint arXiv:2401.04334*, 2024.
- [16] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “Badnets: Evaluating backdooring attacks on deep neural networks,” *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.
- [17] H. Zhang, S. Hu, Y. Wang, L. Y. Zhang, Z. Zhou, X. Wang, Y. Zhang, and C. Chen, “Detector collapse: Backdooring object detection to catastrophic overload or blindness in the physical world,” in *Proceedings of IJCAI*, 2024.
- [18] Y. Liu, X. Ma, J. Bailey, and F. Lu, “Reflection backdoor: A natural backdoor attack on deep neural networks,” in *Proceedings of ECCV*. Springer, 2020, pp. 182–199.
- [19] J. Varley, S. Singh, D. Jain, K. Choromanski, A. Zeng, S. B. R. Chowdhury, A. Dubey, and V. Sindhwani, “Embodied AI with two arms: Zero-shot learning, safety and modularity,” *arXiv preprint arXiv:2404.03570*, 2024.
- [20] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *Proceedings of ICRA*. IEEE, 2023, pp. 9493–9500.
- [21] L. Chen, Y. Lei, S. Jin, Y. Zhang, and L. Zhang, “Rlingua: Improving reinforcement learning sample efficiency in robotic manipulations with large language models,” *IEEE Robotics and Automation Letters*, 2024.
- [22] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, “Backdoor attacks against deep learning systems in the physical world,” in *Proceedings of CVPR*, 2021, pp. 6206–6215.
- [23] Y. Jin, D. Li, A. Yong, J. Shi, P. Hao, F. Sun, J. Zhang, and B. Fang, “RobotGPT: Robot manipulation learning from ChatGPT,” *IEEE Robotics and Automation Letters*, 2024.
- [24] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, “Physically grounded vision-language models for robotic manipulation,” in *Proceedings of ICRA*. IEEE, 2024, pp. 12 462–12 469.
- [25] M. Zhu, Y. Zhu, J. Li, J. Wen, Z. Xu, Z. Che, C. Shen, Y. Peng, D. Liu, F. Feng *et al.*, “Language-conditioned robotic manipulation with fast and slow thinking,” *arXiv preprint arXiv:2401.04181*, 2024.
- [26] R. Jiao, S. Xie, J. Yue, T. SATO, L. Wang, Y. Wang, Q. A. Chen, and Q. Zhu, “Can we trust embodied agents? exploring backdoor attacks against embodied LLM-based decision-making systems,” in *Proceedings of ICLR*, 2025.
- [27] A. Liu, Y. Zhou, X. Liu, T. Zhang, S. Liang, J. Wang, Y. Pu, T. Li, J. Zhang, and W. Zhou, “Compromising LLM driven embodied agents with contextual backdoor attacks,” *IEEE Transactions on Information Forensics and Security*, 2025.
- [28] R. Philipp, A. Mladenow, C. Strauss, and A. Völz, “Machine learning as a service: Challenges in research and applications,” in *Proceedings of iiWAS*, 2020, pp. 396–406.
- [29] Z. Huang, Y. Mao, and S. Zhong, “{UBA-Inf}: Unlearning activated backdoor attack with {Influence-Driven} camouflage,” in *Proceedings of USENIX Security*, 2024, pp. 4211–4228.
- [30] X. Gong, Y. Chen, W. Yang, H. Huang, and Q. Wang, “B3: Backdoor attacks against black-box machine learning models,” *ACM Transactions on Privacy and Security*, vol. 26, no. 4, pp. 1–24, 2023.
- [31] J. Wei, J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou *et al.*, “Larger language models do in-context learning differently,” *arXiv preprint arXiv:2303.03846*, 2023.
- [32] M. Minderer, A. Gritsenko, and N. Houlsby, “Scaling open-vocabulary object detection,” in *Proceedings of NeurIPS*, vol. 36, 2023.
- [33] J. Chen, D. Zhu, X. Shen, X. Li, Z. Liu, P. Zhang, R. Krishnamoorthi, V. Chandra, Y. Xiong, and M. Elhoseiny, “MiniGPT-v2: Large language model as a unified interface for vision-language multi-task learning,” *arXiv preprint arXiv:2310.09478*, 2023.
- [34] F. Zhu, Z. Liu, X. Y. Ng, H. Wu, W. Wang, F. Feng, C. Wang, H. Luan, and T. S. Chua, “Mmdocbench: Benchmarking large vision-language models for fine-grained visual document understanding,” *arXiv preprint arXiv:2410.21311*, 2024.
- [35] Universal Robots, “Universal robots - collaborative robotic arm solutions,” <https://www.universal-robots.com/>, accessed: 2024-11-04.
- [36] S. Liu, J. Zhang, R. X. Gao, X. V. Wang, and L. Wang, “Vision-language model-driven scene understanding and robotic object manipulation,” in *Proceedings of CASE*. IEEE, 2024, pp. 21–26.
- [37] H. Zhang, C. Zhu, X. Wang, Z. Zhou, C. Yin, M. Li, L. Xue, Y. Wang, S. Hu, A. Liu *et al.*, “BadRobot: Manipulating embodied LLMs in the physical world,” in *Proceedings of ICLR*, 2025.
- [38] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” in *Proceedings of NeurIPS*, 2022, pp. 24 824–24 837.

- [39] N. B. Gurur, H. Ren, J. Deng, and X. Xie, "Task-oriented robotic manipulation with vision language models," *arXiv preprint arXiv:2410.15863*, 2024.
- [40] W. Zhao, J. Chen, Z. Meng, D. Mao, R. Song, and W. Zhang, "Vlmpc: Vision-language model predictive control for robotic manipulation," *arXiv preprint arXiv:2407.09829*, 2024.
- [41] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion, "Mdet-modulated detection for end-to-end multi-modal understanding," in *Proceedings of ICCV*, 2021, pp. 1780–1790.
- [42] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen *et al.*, "Simple open-vocabulary object detection with vision transformers," *arXiv preprint arXiv:2205.06230*, vol. 2, 2022.
- [43] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progpprompt: Generating situated robot task plans using large language models," in *Proceedings of ICRA*. IEEE, 2023, pp. 11523–11530.
- [44] S. Hu, W. Liu, M. Li, Y. Zhang, X. Liu, X. Wang, L. Y. Zhang, and J. Hou, "PointCRT: Detecting backdoor in 3D point cloud via corruption robustness," in *Proceedings of ACM MM*, 2023, pp. 666–675.
- [45] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 5–22, 2022.
- [46] OpenAI, "Gpt-4 and gpt-4 turbo," <https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo>, 2024.
- [47] S. Liang, J. Liang, T. Pang, C. Du, A. Liu, E.-C. Chang, and X. Cao, "Revisiting backdoor attacks against large vision-language models," *arXiv preprint arXiv:2406.18844*, 2024.
- [48] Z. Ni, R. Ye, Y. Wei, Z. Xiang, Y. Wang, and S. Chen, "Physical backdoor attack can jeopardize driving with vision-large-language models," *arXiv preprint arXiv:2404.12916*, 2024.
- [49] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang, "Gpt-ner: Named entity recognition via large language models," *arXiv preprint arXiv:2304.10428*, 2023.
- [50] F. Qi, Y. Chen, M. Li, Y. Yao, Z. Liu, and M. Sun, "ONION: A simple and effective defense against textual backdoor attacks," in *Proceedings of EMNLP*, 2021, pp. 9558–9566.
- [51] Z. Wang, Z. Wang, M. Jin, M. Du, J. Zhai, and S. Ma, "Data-centric NLP backdoor defense from the lens of memorization," *arXiv preprint arXiv:2409.14200*, 2024.
- [52] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun, "Rethinking stealthiness of backdoor attack against nlp models," in *Proceedings of ACL*, 2021, pp. 5543–5557.
- [53] M. Xue, C. He, Y. Wu, S. Sun, Y. Zhang, J. Wang, and W. Liu, "Ptb: Robust physical backdoor attacks against deep neural networks in real world," *Computers & Security*, vol. 118, p. 102726, 2022.
- [54] E. Wenger, R. Bhattacharjee, A. N. Bhagoji, J. Passananti, E. Andere, H. Zheng, and B. Zhao, "Finding naturally occurring physical backdoors in image datasets," in *Proceedings of NeurIPS*, vol. 35, 2022, pp. 22103–22116.
- [55] O. Avrahami, D. Lischinski, and O. Fried, "Blended diffusion for text-driven editing of natural images," in *Proceedings of CVPR*, 2022, pp. 18208–18218.
- [56] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Proceedings of NeurIPS*, vol. 36, 2024.
- [57] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," in *Proceedings of CVPR*, 2024, pp. 26296–26306.
- [58] Y. Zhang, R. Zhang, J. Gu, Y. Zhou, N. Lipka, D. Yang, and T. Sun, "Llavar: Enhanced visual instruction tuning for text-rich image understanding," *arXiv preprint arXiv:2306.17107*, 2023.
- [59] Orbbec, "Gemini 3351 stereo vision camera," 2025, accessed: 2025-01-18. [Online]. Available: <https://www.orbbec.com/products/stereo-vision-camera/gemini-3351/>
- [60] OpenAI, "Chatgpt," <https://chatgpt.com>, 2025, accessed: 2025-01-03.
- [61] J. Jiang, X. Luo, Q. Luo, L. Qiao, and M. Li, "An overview of hand-eye calibration," *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 1, pp. 77–97, 2022.
- [62] T. Gupta and A. Kembhavi, "Visual programming: Compositional visual reasoning without training," in *Proceedings of CVPR*, 2023, pp. 14953–14962.
- [63] V. Korrapati, "Moondream2: A small vision language model," <https://huggingface.co/vikhyat/moondream2>, 2024, accessed: 2024-11-04.
- [64] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *Proceedings of ICCV*, 2021, pp. 16463–16472.
- [65] R. Zhang, H. Li, R. Wen, W. Jiang, Y. Zhang, M. Backes, Y. Shen, and Y. Zhang, "Instruction backdoor attacks against customized LLMs," in *Proceedings of USENIX Security*, 2024, pp. 1849–1866.
- [66] E. Robotics, "Elephant robotics official website," 2024, accessed: 2024-11-18. [Online]. Available: <https://www.elephantrobotics.com/en/>
- [67] Elephant Robotics, "Camera flange 2.0," 2025, accessed: 2025-01-19. [Online]. Available: <https://shop.elephantrobotics.com/en-sg/products/camera-flange-2-0>
- [68] Z. Sha, X. He, P. Berrang, M. Humbert, and Y. Zhang, "Fine-tuning is all you need to mitigate backdoor attacks," *arXiv preprint arXiv:2212.09067*, 2022.
- [69] K. Chen, X. Lou, G. Xu, J. Li, and T. Zhang, "Clean-image backdoor: Attacking multi-label models with poisoned labels only," in *Proceedings of ICLR*, 2022.
- [70] Z. Liu, Z. Zhao, and M. Larson, "Image shortcut squeezing: Countering perturbative availability poisons with compression," in *Proceedings of ICML*, 2023.
- [71] T. Y. Liu, Y. Yang, and B. Mirzasoleiman, "Friendly noise against adversarial noise: a powerful defense against data poisoning attack," in *Proceedings of NeurIPS (NeurIPS'22)*, vol. 35, 2022, pp. 11947–11959.
- [72] X. Liu, M. Li, H. Wang, S. Hu, D. Ye, H. Jin, L. Wu, and C. Xiao, "Detecting backdoors during the inference stage based on corruption robustness consistency," in *Proceedings of CVPR*, 2023, pp. 16363–16372.
- [73] Y. Wang, R. Wu, K. Mo, J. Ke, Q. Fan, L. J. Guibas, and H. Dong, "AdaAfford: Learning to adapt manipulation affordance for 3D articulated objects via few-shot interactions," in *Proceedings of ECCV*. Springer, 2022, pp. 90–107.
- [74] C. Ning, R. Wu, H. Lu, K. Mo, and H. Dong, "Where2explore: Few-shot affordance learning for unseen novel categories of articulated objects," in *Proceedings of NeurIPS*, vol. 36, 2023.
- [75] A. Xie, L. Lee, T. Xiao, and C. Finn, "Decomposing the generalization gap in imitation learning for visual robotic manipulation," in *Proceedings of ICRA*. IEEE, 2024, pp. 3153–3160.
- [76] T. Dai, J. Wong, Y. Jiang, C. Wang, C. Gokmen, R. Zhang, J. Wu, and L. Fei-Fei, "Acdc: Automated creation of digital cousins for robust policy learning," *arXiv preprint arXiv:2410.07408*, 2024.
- [77] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.
- [78] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, 2024.
- [79] Y. Song, T. Wang, P. Cai, S. K. Mondal, and J. P. Sahoo, "A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–40, 2023.
- [80] H. Huang, O. Zheng, D. Wang, J. Yin, Z. Wang, S. Ding, H. Yin, C. Xu, R. Yang, and Q. Zheng, "ChatGPT for shaping the future of dentistry: the potential of multi-modal large language model," *International Journal of Oral Science*, vol. 15, no. 1, p. 29, 2023.
- [81] 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 *Proceedings of CoRL*, 2023, pp. 2165–2183.
- [82] G. Cheng, C. Zhang, W. Cai, L. Zhao, C. Sun, and J. Bian, "Empowering large language models on robotic manipulation with affordance prompting," *arXiv preprint arXiv:2404.11027*, 2024.
- [83] W. Zhang, X. Kong, T. Braunl, and J. B. Hong, "Safeembodai: A safety framework for mobile robots in embodied AI systems," *arXiv preprint arXiv:2409.01630*, 2024.
- [84] A. Robey, Z. Ravichandran, V. Kumar, H. Hassani, and G. J. Pappas, "Jailbreaking LLM-controlled robots," *arXiv preprint arXiv:2410.13691*, 2024.
- [85] S. Liu, J. Chen, S. Ruan, H. Su, and Z. Yin, "Exploring the robustness of decision-level through adversarial attacks on LLM-based embodied models," in *Proceedings of ACM MM*, 2024, p. 8120–8128.
- [86] X. Wu, R. Xian, T. Guan, J. Liang, S. Chakraborty, F. Liu, B. Sadler, D. Manocha, and A. S. Bedi, "On the safety concerns of deploying LLMs/VLMs in robotics: Highlighting the risks and vulnerabilities," *arXiv preprint arXiv:2402.10340*, 2024.
- [87] C. M. Islam, S. Salman, M. Shams, X. Liu, and P. Kumar, "Malicious path manipulations via exploitation of representation vulnerabilities of vision-language navigation systems," in *Proceedings of IROS*, 2024.
- [88] T. Wang, D. Liu, J. C. Liang, W. Yang, Q. Wang, C. Han, J. Luo, and R. Tang, "Exploring the adversarial vulnerabilities of vision-language-action models in robotics," *arXiv preprint arXiv:2411.13587*, 2024.