VLATest: Testing and Evaluating Vision-Language-Action Models for Robotic Manipulation

ZHIJIE WANG, University of Alberta, Canada
ZHEHUA ZHOU, University of Alberta, Canada
JIAYANG SONG, University of Alberta, Canada
YUHENG HUANG, The University of Tokyo, Japan
ZHAN SHU, University of Alberta, Canada
LEI MA, The University of Tokyo, Japan and University of Alberta, Canada

The rapid advancement of generative AI and multi-modal foundation models has shown significant potential in advancing robotic manipulation. Vision-language-action (VLA) models, in particular, have emerged as a promising approach for visuomotor control by leveraging large-scale vision-language data and robot demonstrations. However, current VLA models are typically evaluated using a limited set of hand-crafted scenes, leaving their general performance and robustness in diverse scenarios largely unexplored. To address this gap, we present VLATest, a fuzzing framework designed to generate robotic manipulation scenes for testing VLA models. Based on VLATest, we conducted an empirical study to assess the performance of seven representative VLA models. Our study results revealed that current VLA models lack the robustness necessary for practical deployment. Additionally, we investigated the impact of various factors, including the number of confounding objects, lighting conditions, camera poses, unseen objects, and task instruction mutations, on the VLA model's performance. Our findings highlight the limitations of existing VLA models, emphasizing the need for further research to develop reliable and trustworthy VLA applications.

CCS Concepts: • **Software and its engineering → Software defect analysis**; • **General and reference → Empirical studies**; • **Computer systems organization → Robotics**.

Additional Key Words and Phrases: Vision-Language-Action Models, Robotic Manipulation, Robustness, Empirical Study

ACM Reference Format:
Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. 2025. VLATest: Testing and Evaluating Vision-Language-Action Models for Robotic Manipulation. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE073 (July 2025), 24 pages. https://doi.org/10.1145/3729343

1 Introduction

Robotic manipulation is widely regarded as one of the most important areas within cyber-physical systems (CPS). Over the past few decades, robotic manipulation and its applications have been implemented across various domains, such as industrial automation [20, 25], healthcare [33, 41], and logistics and warehousing [15, 64]. With the rapid advancement of AI techniques, researchers and practitioners have been exploring the integration of AI for planning [27, 62] and control [69, 77]

Authors' Contact Information: Zhijie Wang, University of Alberta, Edmonton, Canada, zhijie.wang@ualberta.ca; Zhehua Zhou, University of Alberta, Edmonton, Canada, zhehua1@ualberta.ca; Jiayang Song, University of Alberta, Edmonton, Canada, jiayan13@ualberta.ca; Yuheng Huang, The University of Tokyo, Tokyo, Japan, yuhenghuang42@g.ecc.u-tokyo.ac.jp; Zhan Shu, University of Alberta, Edmonton, Canada, zshu1@ualberta.ca; Lei Ma, The University of Tokyo, Tokyo, Japan and University of Alberta, Edmonton, Canada, ma.lei@acm.org.

ACM 2994-970X/2025/7-ARTFSE073
https://doi.org/10.1145/3729343

in robotic manipulation. For instance, recent studies have employed deep reinforcement learning for robotics control to enhance the adaptability of the system against unseen scenarios [30, 58].

Meanwhile, the emergence of foundation models, such as large language models (LLMs) and vision-language models (VLMs), has introduced new opportunities for advancing AI-enabled robotic manipulation [16, 66]. In particular, LLMs and VLMs have demonstrated promising capabilities to participate in the loop of robotics system development and operation to enhance corresponding reasoning, perception, and task-planning abilities [18, 36, 48]. In contrast to the general purpose-oriented LLMs and VLMs, vision-language-action (VLA) models were developed exclusively to generate robot actions for manipulation based on visual observations from cameras and task instructions provided by users' natural language input. Techniques such as reinforcement learning from human feedback (RLHF) [4] further enhance these models' ability to interpret human intent from natural language commands. As a result, it is now possible to let robotics perform manipulation tasks simply by prompting the VLA models. Additionally, the large-scale pre-training nature of such foundation models makes it possible to adapt to a diverse range of downstream manipulation tasks without task- and environment-specific niche design. A recent study demonstrates that the state-of-the-art (SOTA) VLA model, RT-2 [8], can be extended to complex tasks such as symbol understanding and human recognition that were unseen during training [8].

However, the data-driven nature of VLA models makes them difficult to interpret, which hinders the safety and applicability of VLA models to more robotics applications in practice. To enhance the reliability and trustworthiness of VLA models for robotic manipulation, the development of quality assurance techniques such as testing, debugging, and repairing has become an urgent need in both industrial and academic communities. In particular, comprehensive and adequate testing is essential for assessing a VLA model's capabilities and limitations. Unfortunately, current VLA models are typically evaluated using only a limited set of hand-crafted scenes, where the comprehensiveness and effectiveness of such scenes are often inadequate, which subsequently leaves the general performance and behavior characteristics of VLA models largely unexplored. Moreover, unlike conventional LLMs, which only process natural language text inputs, VLA models are empowered by the multimodality perception ability; therefore, when deploying VLA models in real-world scenarios, visual factors such as confounding objects, lighting conditions, and camera angles may significantly impact their performance. Despite this, there is currently a lack of deep understanding regarding the robustness of VLA models against such environmental factors.

To address this gap, we conducted a large-scale empirical study to evaluate the overall performance and robustness of popular VLA models. We first propose VLATest, one of the earliest testing frameworks specifically designed for evaluating VLA models in robotic manipulation. We introduce a set of ten testing operators and design a generation-based fuzzing framework that automatically produces testing scenes to assess the performance and robustness of VLA models in specific robotic manipulation tasks. VLATest is implemented within the Maniskill2 simulation environment [26]. By leveraging VLATest, we aim to efficiently identify potential bugs in using VLA models for robotic manipulation, enabling comprehensive evaluation of VLA models under varying conditions. We designed a set of research questions to examine key factors in reliable robotic manipulation with VLA: *basic performance*, *task complexity*, *perception robustness*, *OOD (out-of-distribution) robustness*, and language model robustness. Specifically, we investigated the following six research questions:

RQ1 How do VLA models perform in popular robotic manipulation tasks?
RQ2 How does the number of confounding objects affect a VLA model's performance?
RQ3 Does the change in lighting conditions affect a VLA model's performance?
RQ4 Does the change of camera pose affect a VLA model's performance?

RQ5 How robust do VLA models perform against unseen objects?

RQ6 How robust do VLA models perform against task instruction mutations?

To investigate these research questions, we generated 18,604 testing scenes across four different robotic manipulation tasks. We selected *seven* popular publicly available VLA models: RT-1-1k, RT-58k, RT-400k [7], RT-1-X [59], Octo-small, Octo-base [75], and OpenVLA-7b [40]. The experiments included a total of 78,604 rounds of simulation execution, which took more than 580 GPU hours.

Our analysis shows that current VLA models exhibit **subpar performance** across the four robotic manipulation tasks studied. As the number of confounding objects increases, it becomes more challenging for these models to accurately locate and manipulate the correct objects. Additionally, we observed that the subject VLA models lack robustness when subjected to changes in lighting conditions and camera angles, performing much worse compared to the default settings. Meanwhile, we found that VLA models with large-scale pre-training generally show better robustness against such changes. Finally, we found that these VLA models struggle significantly with unseen objects, exhibiting a significant drop in performance compared to manipulating seen objects. Current VLA models also often fell short in understanding the same task instruction paraphrased in different ways. Our findings provide practical guidelines for developers working with VLA models on specific robotic manipulation tasks and underscore the need for more advanced VLA models, as well as novel quality assurance techniques to enhance the robustness and reliability of the VLA models.

In summary, this paper makes the following contributions:

- **An empirical study.** We conducted a large-scale empirical study to evaluate the performance and robustness of seven popular VLA models across four robotic manipulation tasks under various conditions, including variations in the number of confounding objects, lighting conditions, camera poses, and the manipulation of both seen and unseen objects.
- **A testing framework.** To conduct our empirical study, we designed and implemented a generation-based fuzzing framework, VLATest, to test VLA models by incorporating various operators in robotic manipulation tasks.
- **Implications.** We discussed the challenges and limitations of current VLA models, along with the implications and future opportunities for enhancing their robustness and reliability.
- **Artifacts.** Our artifacts, including the replication packages and the generated testing scenes, are available on a GitHub repository: https://github.com/ma-labo/VLATest.

2 Background: Vision-Language-Action Models

Vision-Language-Action (VLA) models [7, 40, 59, 75] are a type of deep neural network that takes natural language input from the user as task instructions and visual input from a camera as observations. The output of a VLA model is a set of actions to achieve the designated manipulation according to the task instructions, such as moving a robotic arm's joints and opening the gripper.

2.1 VLA Model for Robotic Manipulation

Fig. 1 illustrates the common architecture of a VLA model. Given a natural language input T and an image input I_1 at the first timestamp, the VLA model's natural language tokenizer and visual encoder first project T and I_1 into sets of tokens, $\mathbf{T} = \{t_1, \ldots, t_m\}$ and $\mathbf{I}_1 = \{i_{11}, \ldots, i_{1n}\}$, respectively. These tokens are then concatenated and fed into a transformer model to predict action token(s) $\mathbf{A}_1$. An action head layer then de-tokenizes $\mathbf{A}_1$ into robot action values $[\Delta_{x1}, \Delta_{\theta 1}, \Delta_{grip1}]$, where $\Delta_{x1} \in \mathbb{R}^3$, $\Delta_{\theta 1} \in \mathbb{R}^3$, and $\Delta_{grip1} \in \mathbb{R}^1$ denote the translation (x, y, z movement), rotation, and gripper actions of the end-effector that should be performed after the observation, respectively.

After the robot performs these actions, the action tokens $\mathbf{A}_1$ and a new set of image tokens $\mathbf{I}_2 = \{i_{21}, \ldots, i_{2n}\}$, based on the visual observation at the next timestamp, are concatenated with $\mathbf{T}$

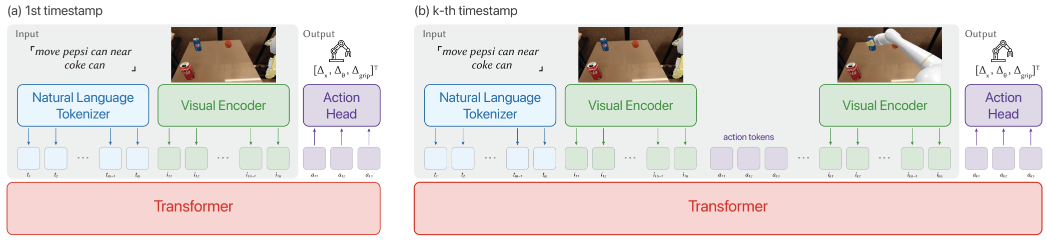

(a) 1st timestamp
Input
move pepsi can near coke can
Output
$[\Delta_x, \Delta_\theta, \Delta_{grip}]^T$
Natural Language Tokenizer
Visual Encoder
Action Head
t_1 t_2 t_{m-1} t_m
i_{11} i_{12} i_{1n-1} i_{1n}
a_{11} a_{12} a_{13}
Transformer
(b) k-th timestamp
Input
move pepsi can near coke can
Output
$[\Delta_x, \Delta_\theta, \Delta_{grip}]^T$
Natural Language Tokenizer
Visual Encoder
action tokens
Visual Encoder
Action Head
t_1 t_2 t_{m-1} t_m
i_{11} i_{12} i_{1n-1} i_{1n}
a_{11} a_{12} a_{13}
i_{k1} i_{k2} i_{kn-1} i_{kn}
a_{k1} a_{k2} a_{k3}
Transformer

Fig. 1. **Architecture and workflow of a VLA model.** (a) Generating robot actions at the first timestamp. (b) Generating robot actions at the *k*-th timestamp.

and $\mathbf{I}_1$. A new input sequence $\{\mathbf{T}, \mathbf{I}_1, \mathbf{A}_1, \mathbf{I}_2\}$ is then fed into the transformer to predict new action tokens $\mathbf{A}_2$. The VLA model continues this process until the task is completed or the pre-defined maximum number of steps is exceeded.

2.2 Training and Evaluation of the VLA Model

Training. There are typically two approaches to training a VLA model: (1) training from scratch and (2) fine-tuning a general-purpose VLM. *Training from scratch* involves building a VLA model directly on robot demonstration data. This approach is commonly applied for models with relatively small-scale architectures and limited computational resources, such as RT-1 [7] and Octo [75], which typically have fewer than 100 million parameters. By contrast, fine-tuning from an existing VLM leverages the flexibility of re-using a larger model (more than 1 billion parameters), such as Llava [40, 51], which has been pre-trained on massive amounts of image and text data from diverse domains. The large-scale pre-training may enhance the generalizability of the fine-tuned VLA models, particularly when manipulating unseen objects and tasks.

Evaluation. A VLA model is usually evaluated by measuring its performance on specific skills, such as picking up an object in a scene. To evaluate a particular skill, developers must first create an adequate testing scene, which involves configuring target objects, confounding objects, and environmental factors such as lighting conditions. Additionally, a text prompt should be meticulously crafted before performing the manipulation task. To assess the model's performance, developers usually design a set of evaluation metrics. For instance, when performing the task "picking up an object", the metrics include: (1) whether the robot grasps the correct object, (2) whether the object is successfully lifted, and (3) whether the robot can sustain lifting the object for a short period. The execution of the VLA model and the robot can occur in either a simulated or real-world environment. In real-world scenarios, manual labeling is required to compute these metrics, whereas in simulated environments, the evaluation process can be automated.

3 VLATEST

In this section, we first introduce the operators included in the testing scene for a robotic manipulation task. Next, we describe the algorithm used to generate a testing scene in VLATEST.

3.1 Operators

We consider **four** categories of testing operators in VLATEST as shown in Fig. 2, resulting in a total of ten testing operators. We choose these operators because they are the most essential ones in the configuration of a robotic manipulation task [55, 65]. Thus, VLATEST can be easily adapted for a wide range of tasks. Additionally, by only changing these operators, our generated scenes will not change the intrinsic nature of a manipulation task. This ensures that all test cases will not require the VLA models to perform actions that were never performed during training.

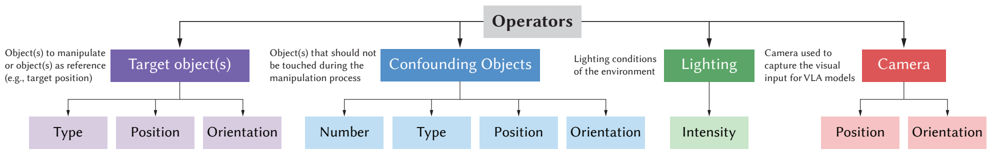

Operators
Object(s) to manipulate or object(s) as reference (e.g., target position)
Target object(s)
Object(s) that should not be touched during the manipulation process
Confounding Objects
Lighting conditions of the environment
Lighting
Camera used to capture the visual input for VLA models
Camera
Type
Position
Orientation
Number
Type
Position
Orientation
Intensity
Position
Orientation

Fig. 2. Operators used for testing VLA models in VLATest.

- ***Target object(s).*** A target object refers to an object that is required to be manipulated in a task (e.g., an object to pick up) or an object that serves as a reference in a task (e.g., an object to be placed on). For each target object, we consider three different operators: (1) type of object (e.g., *Pepsi can*), (2) position, and (3) orientation.
- ***Confounding objects.*** Different from *target object(s)*, confounding objects are typically not part of the task, and the robot is expected to avoid contacting these objects during manipulation. In addition to the three operators for *target object(s)*, we also consider the number of confounding objects as a distinct operator in the confounding objects category.
- ***Lighting.*** The lighting condition affects the rendering of images captured by the camera, thereby influencing the visual input of the VLA models. We consider lighting intensity as one operator.
- ***Camera.*** The camera's pose can also affect the visual input of the VLA models. We consider its position and orientation as two separate operators in VLATest.

3.2 Testing Scene Generation

We summarize the algorithm for generating a testing scene in Algorithm 1. Intuitively, there are four distinct steps: (1) sampling semantically valid target object(s), (2) sampling confounding objects (optional), (3) mutating lighting conditions (optional), and (4) mutating camera poses (optional).

Sampling semantically valid target object(s). For a robotic manipulation task, we first randomly sample target object(s) without replacement from the object database O (Lines 5-9). We also check if the selected objects are *semantically valid* according to the task requirements (Line 4). For instance, in a task requiring object A to be placed over object B (Task 3), it is possible to put an apple on a plate, but it is impractical to put an apple on a ball, since placing an apple on a ball stably could be very challenging. Similarly, for Task 4, it is impossible to put an apple inside a Coke can. We consider scenes that can not be successfully manipulated by the robots to be *semantically invalid*. We manually created a list of invalid objects for tasks 3 and 4 to check if a generated test scene is semantically valid. After selecting the target object(s), we proceed to sample their positions and orientations (Line 7). In our empirical study, we used a random sampler for pose selection. If there are multiple target objects, we ensure they remain far enough apart ($> safe\_dist$) to avoid generating invalid testing scenes. For instance, in the previous example, if object A is already placed on object B during the sampling of their positions, the testing scene becomes invalid.

Sampling confounding objects. If the number of confounding objects $N_{confound}$ is greater than 0 (Line 11), we further randomly sample confounding objects from O without replacement (Lines 12-16). Note that we consider similar objects with different semantics to be different. For instance, a Coke can is considered different from a Pepsi can. Therefore, having both of them as confounding objects in one scene is possible. For each confounding object, we follow the same procedure as used for sampling the target objects to assign their position and orientation (Line 14). In our empirical study, we utilized a random sampler to generate positions and orientations.

Mutating lighting conditions and camera poses. For lighting conditions, we generate a factor α to mutate the lighting intensity value (Lines 18-20). For the camera, we adjust its position by a distance d and rotate it by an angle θ (Lines 21-23). Note that both d and θ are kept small to ensure

Algorithm 1: Testing scene generation in VLATest.

```
Input: an object database $O$, the robotic manipulation task $T$, the number of target objects $N_{target}$, the number of confounding objects $N_{confound}$, the lighting mutation option $lighting\_flag$, and the camera mutation option $camera\_flag$, the minimum distance between objects $safe\_dist$
Output: a testing scene
1  $O \leftarrow$ random_shuffle($O$);
2  $O_{target}, O_{confound} \leftarrow \emptyset, \emptyset$;
3  $intensity, cam_{pos}, cam_{ori} \leftarrow$ default values;
4  while not semantic_valid($O_{target}, T$) do
5      for $i$ in $1, \ldots, N_{target}$ do
6          $obj \leftarrow O$.pop();
7          $pos, ori \leftarrow$ pose_sampler($O_{target}, safe\_dist$);
8          $O_{target} \leftarrow O_{target} \bigcup (obj, pos, ori)$;
9      end
10 end
11 if $N_{confound} > 0$ then
12     for $i$ in $1, \ldots, N_{confound}$ do
13         $obj \leftarrow O$.pop();
14         $pos, ori \leftarrow$ pose_sampler($O_{target} \bigcup O_{confound}$, $safe\_dist$);
15         $O_{confound} \leftarrow O_{target} \bigcup (obj, pos, ori)$;
16     end
17 end
18 if $lighting\_flag$ then
19     $intensity \leftarrow$ mutate_lighting($intensity$);
20 end
21 if $camera\_flag$ then
22     $cam_{pos}, cam_{ori} \leftarrow$ mutate_camera($cam_{pos}$, $cam_{ori}$);
23 end
24 return $\{O_{target}, O_{confound}, intensity, (cam_{pos}, cam_{ori})\}$;
```

that the entire scene remains within the camera's view. In our empirical study, we used a random generator to generate α, d, and θ within specified ranges.

After generating each scene, we compared its configuration with existing ones to prevent duplication. Throughout our experiments, no identical test scenes were generated.

4 Empirical Study

Based on VLATest, we conducted an empirical study to investigate the performance of SOTA VLA models. In this section, we first list our research questions, followed by the empirical setup, which includes subject VLA models, subject robotic manipulation tasks, prompt template, and implementation details.

4.1 Research Questions

Our empirical study investigates the following research questions to examine the key factors in reliable robotic manipulation: *basic performance* (RQ1), *task complexity* (RQ2), *perception robustness* (RQ3 & RQ 4), *OOD robustness* (RQ5), and language model robustness (RQ6).

- ***RQ1. How do VLA models perform in popular robotic manipulation tasks?***
  This research question seeks to evaluate the performance of SOTA VLA models in various robotic manipulation tasks. While previous studies [7, 75] typically rely on a few hand-crafted test cases, we use VLATest to generate a large number of test cases. This approach enables a more comprehensive assessment of SOTA VLA models' performance, providing insights into current challenges and opportunities.
- ***RQ2. How does the number of confounding objects affect a VLA model's performance?***
  Intuitively, the presence of more confounding objects (i.e., objects unrelated to the assigned manipulation task) increases the complexity of a robotic manipulation task for VLA models. However, it remains unclear whether there is an upper limit to the task complexity that a VLA model can handle effectively. To address this, we conduct controlled experiments to help practitioners understand how VLA models perform as the number of confounding objects varies.
- ***RQ3. Does the change in lighting conditions affect a VLA model's performance?***
  When deploying robotics and VLA models in real-world environments, external conditions can vary significantly. Ideally, a VLA model should be robust against different lighting conditions, such as varying illumination intensities. This research aims to offer practitioners practical guidelines on the robustness of VLA models under diverse lighting setups.
- ***RQ4. Does the change of camera pose affect a VLA model's performance?***
  Since VLA models are pre-trained on large-scale vision datasets, they are expected to demonstrate

Table 1. Overview of subject VLA models.

VLA Models	**RT-1**			**RT-1-X**	**Octo-small**	**Octo-base**	**OpenVLA-7b**
	1k	58k	400k				
Model Size	35M			35M	27M	93M	7.6B
Release Date	Dec. 2022			Oct. 2023	Dec. 2023	Dec. 2023	Jun. 2024
Report Performance	—	—	92%	73%	30%	71%	71%
(Dataset)	RT-1 Paper 6 Skills				UR5		WidowX

robustness when input images are captured from various angles. This research question seeks to determine whether and to what extent current VLA models maintain robustness when operating with different camera poses.

- ***RQ5. How robust do VLA models perform against unseen objects?***
  Objects that were unseen in the robotic demonstration data may present when deploying robotics and VLA models in practical scenarios. It is unclear whether the large-scale pre-training of VLA models can make it generalizable to these unseen objects. Also, there is currently limited understanding of the performance gap between seen-object and unseen-object robotic manipulation tasks. To address this, we investigate this research question by leveraging an external object database to assess the limitations and challenges.
- ***RQ6. How robust do VLA models perform against task instruction mutations?***
  The task instruction plays a vital role in the robotic manipulation with VLA models. Task instructions for a VLA model can be paraphrased using different words and sentence structures. Ideally, a VLA model is expected to perform robustly against different mutations of task instructions that carry the same meanings. In this research question, we seek to understand whether VLA models could demonstrate such *language model robustness*.

4.2 Subject VLA Models

To investigate our research questions, we studied four different series of open-source VLA models: RT-1 [7], RT-1-X [59], Octo [75], and OpenVLA [40]. For RT-1 [7], we studied three publicly available variants: *RT-1-1k*, *RT-1-58k*, and *RT-1-400k* [1]. We also studied two variants for Octo [75]: *Octo-small* and *Octo-base*. Note that we were unable to study the SOTA VLA model, RT-2 [8], as it has not yet been made publicly available. Thus, we included a total of seven subject VLA models in our empirical study. Table 1 summarizes the characteristics of these models, and we provide more details for each model below.

- **RT-1** [7] model was released by Google Research. The RT-1 model includes a FiLM EfficientNet-B3 [61, 74] pre-trained on the ImageNet dataset [63] for tokenizing visual inputs and language instructions before connecting to a transformer to generate robotic manipulation actions. The RT-1 model was trained on an unpublished set of 130k robot demonstrations collected by Google.
- **RT-1-X** [59] model was released by Google DeepMind. RT-1-X shares the same model architecture as RT-1. However, RT-1-X was trained on the open-source dataset, Open X-Embodiment [59], which includes 160k robot demonstrations collected from 22 different robots.
- **Octo** [75] model, released by UC Berkeley, includes a ViT model [17] as its backbone transformer. It introduces the "readout" token, allowing developers to flexibly add new observation inputs or action output heads to the model during downstream fine-tuning. Octo was trained on a subset of the Open X-Embodiment dataset, which includes about 65k robot demonstrations. Octo-small (27M parameters) and Octo-base (93M parameters) are two variants of Octo, utilizing ViT-S and ViT-B as their backbone transformers, respectively.
- **OpenVLA-7b** [40] is the most recent VLA model released by Stanford University. OpenVLA-7b includes a 600M-parameter visual encoder consisting of pre-trained SigLIP [92] and DinoV2 [57]

[1]The number denotes the training steps taken for training an RT-1 model.

models and a 7B-parameter Llama 2 [76] as the language model backbone. OpenVLA-7b was then fine-tuned on the Open X-Embodiment dataset.

4.3 Robotic Manipulation Tasks

We included four different robotic manipulation tasks in our study. Now, we briefly introduce each of them. We also refer to the supplementary materials for the video demo of each task.

- ***Task 1: Pick up an object.*** This task requires a VLA model to identify the target object and generate the appropriate control signals to grasp and lift it. To succeed, the robot must grasp the correct object and lift it at least 0.02 meters for five consecutive frames.
- ***Task 2: Move object A to object B.*** This task requires a VLA model to first identify the source object (A) and then output the corresponding control signals to move it near the target object (B). To succeed, the robot must move the correct object to within 0.05 meters of the target object.
- ***Task 3: Put object A on top of object B.*** Different from Task 2, this task requires the VLA model to output control signals that could stack object A on top of object B. To succeed, object A should be placed stably on top of object B.
- ***Task 4: Put object A into object B.*** Different from Task 3, this task requires the VLA model to generate control signals that place object A inside object B (e.g., into a kitchen sink or a basket). To succeed, object A must be completely inside object B.

For each task, VLATest automatically verifies whether a VLA model has successfully performed it by checking if all corresponding specifications (e.g., the lifted object's height) are met.

4.4 Prompt Templates

For each task in RQ1 ~ RQ5, we follow the previous work [7, 40, 44, 59] to use the standard prompt template for each task: (1) `pick up [object name]`, (2) `move [object name] near [object name]`, (3) `put [object name] on [object name]`, and (4) `put [object name] into [object name]`. In RQ6, we use the prompts mutated from these standard templates.

4.5 Implementation Details

We conducted all experiments on a server with an AMD 5955WX CPU and two NVIDIA RTX A6000 GPUs. The operating system is 64-bit Ubuntu 20.04 LTS with Python 3.10 and CUDA 12.2. We implemented VLATest with Maniskill2 simulation environments [26, 44]. We used two object databases: (1) the default object database in Maniskill2 ($N = 18$) in RQ1 ~ RQ4 and RQ6, and (2) YCB object database ($N = 56$) in RQ5. The average execution time of one manipulation task was around 19.8 seconds. Our empirical study took about 586 GPU hours in total.

5 Results

5.1 RQ1: How Do VLA Models Perform in Popular Robotic Manipulation Tasks?

To investigate this research question, we leveraged VLATest to generate 1,000 scenes for each of the four robotic manipulation tasks by randomly selecting target object(s) and sampling 0 to 3 confounding objects. We also randomly assigned poses to these objects. To avoid collision overlaps, we maintained a minimum distance of 0.15 meters between objects during the assignment. The default lighting setups and camera poses were used for this RQ.

Table 2 presents the performance of seven VLA models on these tasks and scenes. Overall, we find that current VLA models do not perform well in the four studied robotic manipulation tasks. The average success rates across the seven VLA models for the four tasks are 12.4%, 6.0%, 1.2%, and 0.5%, respectively. In Task 1, the best-performing VLA model, RT-1-400k, succeeded in only 34.4% of the test scenes. We also observed significant performance drops in Task 2, Task 3, and

Table 2. (**RQ1**) Performance of seven subject VLA models on different manipulation tasks. The top-1, top-2, and top-3 success rates for each step and task are highlighted, respectively.

VLA Models	Task 1: Pick Up			Task 2: Move Near			Task 3: Put On			Task 4: Put In		
	Grasp	Lift	Success	Grasp	Move	Success	Grasp	Move	Success	Grasp	Move	Success
RT-1-1k	10.1%	1.2%	0.7%	9.6%	3.2%	1.4%	0.4%	0.0%	0.0%	1.1%	0.0%	0.0%
RT-1-58k	44.5%	32.6%	28.2%	38.6%	25.2%	10.9%	0.3%	0.0%	0.0%	1.3%	0.0%	0.0%
RT-1-400k	48.4%	41.0%	34.4%	38.8%	23.7%	9.4%	10.4%	1.3%	0.5%	8.9%	0.1%	0.1%
RT-1-X	34.0%	25.8%	19.5%	25.4%	15.2%	5.8%	17.2%	4.4%	2.3%	17.8%	0.7%	0.4%
Octo-small	9.0%	2.0%	0.8%	14.8%	4.1%	1.5%	27.5%	4.6%	2.2%	34.6%	1.1%	1.1%
Octo-base	2.3%	0.4%	0.0%	4.9%	1.5%	0.6%	19.1%	2.6%	1.2%	32.5%	1.1%	1.1%
OpenVLA-7b	15.1%	7.1%	5.9%	43.0%	23.3%	12.7%	36.8%	5.4%	2.1%	19.9%	1.1%	1.1%
Avg.	23.3%	15.7%	12.8%	25.0%	13.7%	6.0%	16.0%	2.6%	1.2%	16.6%	0.6%	0.5%

Task 4 compared to Task 1. The best-performing VLA model in Task 2, OpenVLA-7b, completed only 12.7% of the test scenes. In Task 3 and Task 4, the best-performing VLA models (RT-1-X and Octo-small) achieved success rates of just 2.2% and 2.1%, respectively. One possible reason is that these three tasks are more complex than Task 1, as they require multiple steps of reasoning (i.e., identifying the source and target objects and their positions before generating the corresponding control signals). Among the seven VLA models studied, we did not find any model that performed significantly better than the others across different tasks.

Finding 1

All VLA models exhibit ***subpar performance*** *in the four studied robotic manipulation tasks, particularly in those that require identifying multiple target objects (i.e., Task 2, Task 3, and Task 4). These results suggest that the development of VLA models is still in its early stages, as they are far from being ready for deployment in real-world scenarios.*

To understand the rationale behind the subpar performance of these VLA models, we broke down each testing scene according to the different steps required to complete a task successfully. Specifically, we measured the success rates of each step in a testing scene for the four tasks, as presented in Table 2. For instance, Task 1 requires the robot to (1) grasp the target object (the column *Grasp* in Table 2), (2) lift the object significantly (the column *Lift* in Table 2), and (3) continue lifting for five consecutive frames (the column *Success* in Table 2). In all four tasks, we found that the success rates dropped significantly between steps. For example, in Task 1, the average success rate of correctly grasping the target object is 23.3%. The success rates for lifting and maintaining the lift then drop to 15.7% and 12.4%, respectively. These results indicate that the current VLA models struggle to interpret natural language instructions that require the robot to perform multiple sequential actions.

In Task 2, Task 3, and Task 4, VLA models successfully picked up source objects in 16% to 25% of scenes. However, they then failed to identify the target objects, with only 0.6% to 13.7% of objects being correctly moved and 0.5% to 6.0% of objects being correctly placed. To improve, one could consider the idea of *chain-of-thought* prompting in LLMs. Specifically, the complex task instructions could be decoupled into individual action steps, allowing the VLA model to be prompted step-by-step.

Finding 2

Current VLA models cannot successfully execute tasks that require multiple steps of action, high-lighting the urgent need to improve their capabilities in accurately interpreting task instructions.

Testing Sufficiency. To evaluate the quality of the test scenes generated by VLATest, we further calculated test coverage. However, to the best of our knowledge, no existing coverage metrics are applicable to either VLA models or robotic manipulation. Metrics such as neuron coverage could be

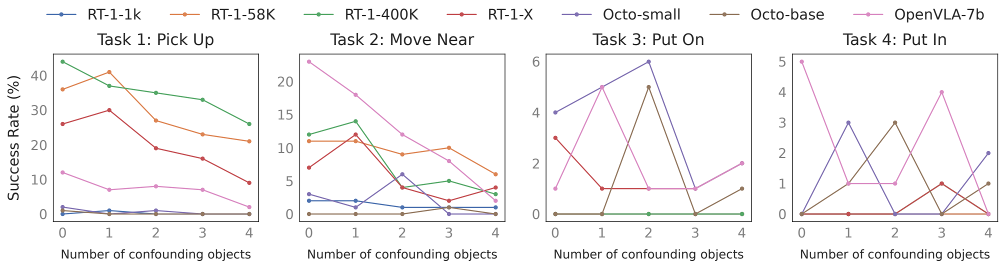


Fig. 3. (**RQ2**) VLA performance vs. the number of confounding objects.

computationally intensive to calculate, as VLA models are typically trained on very large datasets with millions or even billions of parameters.

Table 3. Trajectory coverage results with different number of test cases (*n*).

n	Task 1	Task 2	Task 3	Task 4
10	0.09	0.09	0.10	0.10
100	0.64	0.66	0.63	0.62
1000	1.00	1.00	1.00	1.00

We followed previous work in autonomous driving to implement trajectory coverage [35]. We believe this is a pragmatic choice for enabling test coverage measurements. We acknowledge its limitations in Sec. 8. Specifically, we calculated how many novel target object positions were covered *w.r.t.* the size of the manipulation platform (i.e., the desk). As shown in Table 3, we found that increasing the number of generated test cases *n* from 10 to 1000 allowed VLATest to achieve 100% coverage across all four tasks, indicating that our generated test scenes are sufficient.

5.2 RQ2: How Does the Number of Confounding Objects Affect a VLA Model's Performance?

To investigate RQ2, we used VLATest to generate 100 scenes for each task with a fixed number of n confounding objects, where n was set to 0, 1, 2, 3, and 4, resulting in a total of 500 scenes per task. We used the default lighting conditions and camera poses when investigating this RQ.

The results are depicted in Fig. 3. In the first two tasks (i.e., Task 1 and Task 2), we observed that the VLA model's success rate decreased as the number of confounding objects increased. The average success rates across different VLA models and scenes dropped from 17.3% to 8.3% and from 8.3% to 1.1% when increasing the number of confounding objects from 0 to 4 in Task 1 and Task 2, respectively. However, in Task 3 and Task 4, we did not find a similar pattern. This may be because all VLA models performed poorly even without any confounding objects ($n = 0$). The average success rates across different VLA models and scenes were 1.2% and 0.7% when $n = 0$ and 1.1% and 0.4% when $n = 4$ for Task 3 and Task 4, respectively.

Finding 3

The number of confounding objects affects the VLA model's performance, indicating that VLA models become unreliable in more complex environments. When there were four confounding objects presented in the scene, the VLA models only passed 8.2%, 2.3%, 1.0%, and 0.4% of the test scenes in Task 1, Task 2, Task 3, and Task 4, respectively.

Similar to RQ1, we also analyzed each testing scene and its success at each individual step. We counted the number of testing scenes that succeeded at each step and presented the results in Fig. 4. We found that the major challenge faced by the current VLA models lies in their inability to identify the correct object to manipulate.

In all four tasks, we found that the success rates for grasping the target object decreased as the number of confounding objects increased. These results indicate that when there are multiple confounding objects, the VLA models struggle to locate the correct object to manipulate. Specifically, without any confounding objects, the VLA models successfully located the object to manipulate in

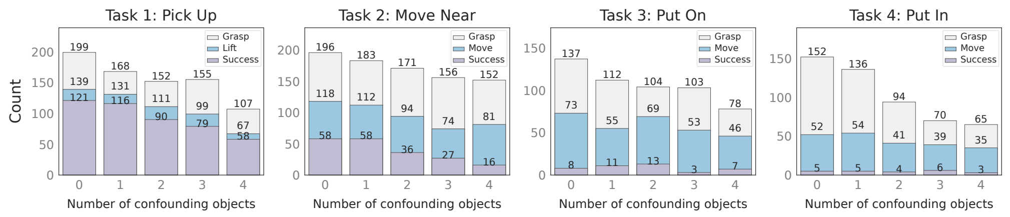


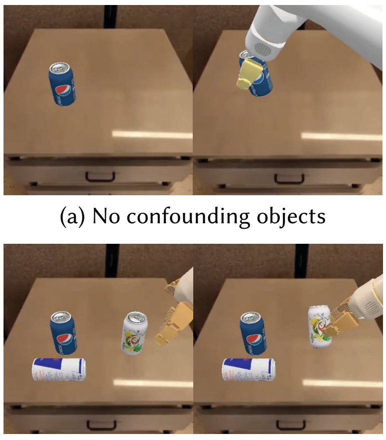
Fig. 4. (**RQ2**) The number of successful scenes at different steps vs. the number of confounding objects.

199, 196, 137, and 152 out of 700 cases for Task 1, Task 2, Task 3, and Task 4, respectively. However, with four confounding objects, the VLA models successfully located the correct object in only 107, 152, 78, and 65 out of 700 cases for the four tasks, respectively. The average success rates of grasping across VLA models dropped to 53.7%, 77.6%, 47.4%, and 42.8% for four tasks, respectively.

Finding 4

With multiple confounding objects in the scene, the VLA models' ability to accurately locate the correct object to manipulate can be significantly affected.

Fig. 5 shows two different testing scenes of Task 1. In both scenes, the VLA model, RT-1-X, was asked to pick up the *Pepsi can*. When there was no confounding object (Fig. 5a), RT-1-X successfully grasped and lifted the Pepsi can. However, when there were two confounding objects (i.e., *7 Up can* and *Red Bull can*), RT-1-X failed to locate the Pepsi can and eventually picked up the 7 Up Can.



(a) No confounding objects

(b) Two confounding objects

Fig. 5. RT-1-X perform Task 1 with different number of confounding objects.

The Impact of Confounding Objects' Similarity. In addition to the number of confounding objects, another factor that may impact the performance of VLA models is the similarity between the confounding object(s) and the target object(s). We manually examined our object database and identified two groups of similar objects: (1) pop cans of different brands and (2) cubes of different colors. Then, we categorized our experiment results from RQ2 into two groups: (1) cases where the target object(s) and confounding object(s) were similar and (2) cases where they were dissimilar. We compared the success rates of these two groups under different n across the four tasks. Specifically, we conducted Mann-Whitney U tests, which showed no significant performance difference between the two groups, with p-values of 0.443, 0.614, 0.657, and 0.443 for the four tasks, respectively. Effect sizes are 0.291, 0.234, 0.271, and 0.257. These results indicate that the similarity between the target objects and the confounding object(s) has little impact on the VLA model's performance.

5.3 RQ3: Does the Change in Lighting Conditions Affect a VLA Model's Performance?

To answer this research question, we first collected all the successfully executed scenes for each VLA model in RQ1, resulting in a total of 1,434 passed test cases. We then re-executed these test cases three times after randomly increasing or decreasing the lighting intensities ($N = 4,302$). Specifically, we randomly sampled a factor $\alpha \in (1, 20]$ for increasing or $\alpha \in [1/20, 1)$ for decreasing the lighting intensity. This factor was then multiplied by the default lighting intensity used in RQ1. We manually checked the cases when α was set to 20 and 1/20 and confirmed that the images captured under such lighting conditions were still recognizable by humans.

Table 4. (**RQ3**) Performance of subject VLA models under default (**Def.**) and mutated (**Mut.**) lighting conditions. Each cell represents the number of successfully passed test scenes by different VLA models in different tasks. The top-1, top-2 and top-3 robust VLA models are highlighted, respectively.

VLA Models	**Task 1 Pick Up**		**Task 2 Move Near**		**Task 3 Put On**		**Task 4 Put In**		**Overall**		
	Def.	Mut.†	Def.	Mut.†	Def.	Mut.†	Def.	Mut.†	Def.	Mut.†	Mut./Def. (%)
RT-1-1k	7	0.7	14	14.0	0	—	0	—	21	14.7	70.0%
RT-1-58k	282	142.0	109	109.0	0	—	0	—	491	251.0	51.1%
RT-1-400k	344	163.0	94	92.0	5	5.0	1	1.0	444	261.0	58.8%
RT-1-X	195	77.3	58	57.0	22	20.7	4	3.0	279	158.0	56.6%
Octo-small	8	0.0	15	2.3	21	6.0	10	1.0	54	9.3	17.2%
Octo-base	0	—	6	1.7	11	3.0	11	0.7	28	5.4	19.3%
OpenVLA-7b	59	21.0	127	127.0	20	20.0	11	11.0	217	169.0	77.9%
Tot.	895	404.0	423	403.0	79	54.7	37	16.7	1434	878.4	61.3%

† Averaged results over three mutations.

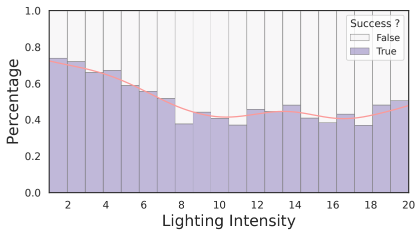
We present the experiment results in Table 4. Overall, we found that randomly perturbing the lighting conditions significantly affected the performance of the VLA models. Out of 1,434 passed test cases with default lighting conditions, only 878.4 test cases (average results over three mutations for each test case) could still be successfully executed with perturbed lighting conditions. The perturbed lighting conditions had a significant impact on Task 1, Task 3, and Task 4, where only about half of the test cases could still be passed after the perturbation. In Task 2, however, five out of seven VLA models exhibited robust performance despite changes in lighting conditions.

Among the seven VLA models, we found that OpenVLA-7b was the most robust, with 77.9% of the test cases still being passed under mutated lighting conditions. A plausible explanation is that OpenVLA-7b leverages two vision models (i.e., SigLIP [92] and DinoV2 [57]) pre-trained on a diverse set of images and fine-tuned on the LLaVA 1.5 data mixture (1 million images and texts). As a result, OpenVLA-7b can effectively interpret images captured under varying lighting conditions. In contrast, Octo-small and Octo-base were trained solely on images captured from robot demonstrations, making them less robust when faced with lighting conditions different from the default ones. Only 17.2% and 19.3% of the test cases could still be successfully executed after changing the lighting conditions, respectively.

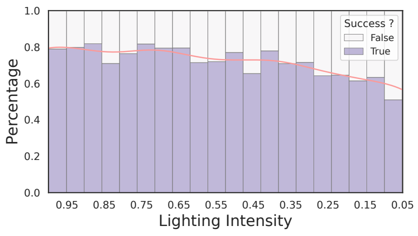
Finding 5

Overall, the VLA models are not sufficiently robust against lighting changes. Only 61.3% of the test cases could still be successfully executed under mutated lighting conditions. Among the seven VLA models, OpenVLA-7b demonstrated the highest robustness to lighting changes.

We further investigated the extent to which the VLA models' performance could be affected by increasing or decreasing lighting intensities. Fig. 6a and Fig. 6b present the results.



(a) Increasing the lighting intensity



(b) Decreasing the lighting intensity

Fig. 6. VLA performance vs. different lighting intensities.

Increasing the lighting intensity ($\alpha > 1$). Even when the lighting intensity is increased by less than 2.5 times (i.e., $\alpha < 2.5$) of the default value, the success rate of the VLA models immediately drops to around 0.7. These results indicate that even a small increase in lighting intensity can

Table 5. (**RQ4**) Performance of subject VLA models with default (**Def.**) and mutated (**Mut.**) camera poses. Each cell represents the number of successfully passed test scenes by different VLA models in different tasks. The top-1, top-2 and top-3 robust VLA models are highlighted, respectively.

VLA Models	**Task 1 Pick Up**		**Task 2 Move Near**		**Task 3 Put On**		**Task 4 Put In**		**Overall**		
	Def.	Mut.†	Def.	Mut.†	Def.	Mut.†	Def.	Mut.†	Def.	Mut.†	Mut./Def. (%)
RT-1-1k	7	0.7	14	4.3	0	—	0	—	21	5.0	23.8%
RT-1-58k	282	121.3	109	27.0	0	—	0	—	491	39.3	30.2%
RT-1-400k	344	180.3	94	21.3	5	0.0	1	0.7	444	202.3	45.6%
RT-1-X	195	43.0	58	13.3	22	0.7	4	0.0	279	57.0	20.4%
Octo-small	8	0.3	15	3.3	21	1.0	10	0.3	54	4.9	9.1%
Octo-base	0	—	6	0.7	11	1.7	11	0.3	28	2.7	9.6%
OpenVLA-7b	59	20.7	127	44	20	3.0	11	0.3	217	68.0	31.3%
Tot.	895	366.3	423	113.9	79	6.4	37	1.6	1434	488.2	34.0%

† Averaged results over three mutations.

significantly degrade the performance of the VLA models. As the factor α increases, the success rate of the VLA models continues to decrease. In extreme cases (e.g., $\alpha > 7.5$), the VLA models succeed in less than half of the scenes.

Finding 6

The performance of the VLA models degrades with increasing lighting intensity. When the lighting intensity exceeds 8 times the default value, the VLA models succeed in only about 40% of the testing scenes that could be passed under default lighting conditions.

Decreasing the lighting intensity ($\alpha < 1$). Similar to increasing the lighting intensity, the VLA models' performance is also affected by even a small perturbation (e.g., $\alpha > 0.9$) when decreasing the intensity. The VLA models' performance degrades with decreasing α. However, compared to increasing the lighting intensity, decreasing it has less significant effects on the models' performance. When $\alpha < 0.2$, the VLA models can still pass around 60% of the testing scenes.

Finding 7

The effect of decreasing the lighting intensity is less significant than increasing it on the VLA models' performance. Even when the lighting intensity is reduced to 0.15 of its default value, the VLA models can still pass 60% of the test cases.

5.4 RQ4: Does the Change of Camera Pose Affect a VLA Model's Performance?

Similar to RQ3, we re-executed the 1,434 passed test cases collected from RQ1 three times after randomly moving and rotating the camera from its default position ($N = 4,302$). Specifically, we rotated the camera around each axis by an angle randomly sampled between -5° and 5°. Additionally, we randomly moved the camera away from its center by a distance randomly sampled between 0 and 5 cm. We manually checked the corner cases where the camera was rotated by 5° and moved by 5 cm, and confirmed that the images captured with these camera angles still included the entire scene and objects. We report our experiment results in Table 5

Overall, we found that the VLA model's performance was greatly affected by the mutated camera poses. With mutated camera poses, the subject VLA models only passed 34.0% of the test cases that could be passed with default camera poses. These results indicate that the current VLA models are very sensitive to the camera's extrinsic calibration results. When deploying VLA models, to achieve compatible performance, the developers need to carefully set up the camera poses, as in the robot demonstration data used for training. However, these may limit the generalizability of the VLA models. Future work may consider data augmentation to improve the VLA model's robustness under different camera settings to enhance generalizability.

Table 6. (**RQ5**) Performance of subject VLA models with seen/unseen objects. Each cell in the column *Seen* and *Unseen* represents the success rate of different VLA models in different tasks. Darker the color, the larger the negative differences between the performance on seen and unseen objects.

VLA Models	Task 1: Pick Up			Task 2: Move Near			Task 3: Put On			Task 4: Put In		
	Seen	Unseen	Diff.†	Seen	Unseen	Diff.†	Seen	Unseen	Diff.†	Seen	Unseen	Diff.†
RT-1-1k	0.7%	1.4%	+100.0%	1.4%	0.9%	-35.7%	0.0%	0.0%	—	0.0%	0.0%	—
RT-1-58k	28.2%	5.0%	-82.3%	10.9%	3.9%	-64.2%	0.0%	0.0%	—	0.0%	0.0%	—
RT-1-400k	34.4%	10.3%	-70.1%	9.4%	2.9%	-69.2%	0.5%	0.4%	-20.0%	0.1%	0.1%	—
RT-1-X	19.5%	3.0%	-84.6%	5.8%	1.3%	-77.6%	2.3%	0.6%	-73.9%	0.4%	0.7%	+75.0%
Octo-small	0.8%	0.1%	-87.5%	1.5%	1.5%	—	2.2%	0.9%	-59.1%	1.1%	0.8%	-27.3%
Octo-base	0.0%	0.0%	—	0.6%	0.4%	-33.3%	1.2%	0.2%	-83.3%	1.1%	0.5%	-54.5%
OpenVLA-7b	5.9%	3.5%	-40.7%	12.7%	2.8%	-78.0%	2.1%	0.9%	-57.1%	1.1%	1.0%	-0.9%
Avg.	12.8%	3.3%	-74.2%	6.0%	2.0%	-66.7%	1.2%	0.4%	-66.7%	0.5%	0.4%	-20.0%

† Diff. = (Unseen - Seen) / Seen

Finding 8

Current VLA models are not robust against mutated camera poses, resulting in degraded performance when the visual input is captured from an angle that varies from the default one. When rotating the camera for a maximum angle of 5° and moving it for a maximum distance of 5cm, the performance dropped to 34.0% of the performance on average with default camera poses.

Among seven subject VLA models, the most robust one, RT-1-400k, succeeded in 45.6% of the test cases with the mutated camera poses that were passed with the default camera poses. OpenVLA-7b also passed 31.3% of the test cases. We noticed a significant performance gap between the two Octo models and other models, where both Octo models only passed less than 10% of the test cases, while the other five models all passed more than 20%. These results may have been largely attributed to the fact that Octo models used the smaller training dataset (~ 65K robot demonstrations) compared with the other ones (130K ~ 160K robot demonstrations). As a result, the Octo model's generalizability to visual inputs captured from different camera angles is significantly affected.

Finding 9

Among the seven subject VLA models, RT-1-400k was the most robust one against the perturbation of camera poses. Octo-series models performed significantly less robustly than the other models. This may be largely attributed to the fact that Octo models were trained with only half of the robot demonstration data compared with the other subject models.

5.5 RQ5: How Robust Do VLA Models Perform Against Unseen Objects?

To investigate this research question, we used an external object dataset, YCB [9]. YCB contains 56 objects that are not included in the Open-Embodiment-X dataset (the training/fine-tuning dataset of our seven subject VLA models). Similar to RQ1, we leveraged VLATest to generate 1,000 testing scenes for each of the four subject robotic manipulation tasks while sampling the target object and the confounding objects from the YCB dataset. We compared the performance of these subject VLA models with objects from the YCB dataset to the results in RQ1.

Table 6 shows the performance of seven subject VLA models when manipulating seen and unseen objects. Overall, when manipulating unseen objects, the performance of the subject VLA models dropped significantly compared with manipulating seen objects. The average performance across different VLA models dropped by 74.2%, 66.7%, 66.7%, and 20.0% in Task 1, Task 2, Task 3, and Task 4, respectively. Notably, we also found that none of these VLA models completely failed when manipulating unseen objects (except those that also failed with seen objects). These results suggest that, while all VLA models exhibit certain possibilities of generalizing themselves to unseen objects, the performance is by far unreliable.

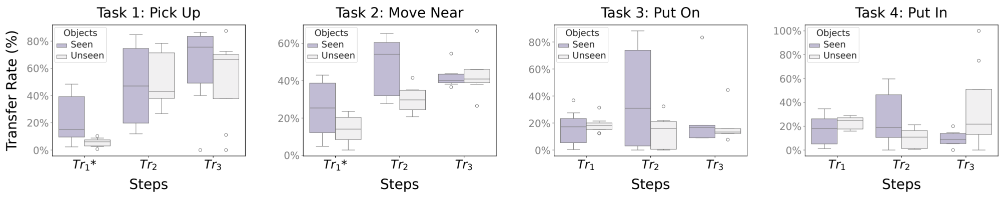


Fig. 7. (**RQ5**) Transfer rate (Tr) of each task when manipulating with seen/unseen objects. Entries with * mean that their mean differences are statistically significant ($p < 0.05$).

Finding 10

Current VLA models, though exhibit potential for generalization, are still unreliable for handling unseen objects. Our subject VLA models saw a performance drop ranging from 20.0% to 74.2% across four tasks when manipulating unseen objects compared with the seen objects.

We also examined each testing scene and the success of the VLA model at every individual step, similar to the investigation of RQ1 and RQ2, to understand how unseen objects might have impacted its performance. Specifically, since the model must succeed at each step sequentially to complete the testing scene, we define a metric called the **transfer rate** (Tr), which calculates the model's ability to transfer success from one step to the next, as follows:

$$Tr_n = \frac{Success\_rate_n}{Success\_rate_{n-1}}, \tag{1}$$

where $Success\_rate_n$ denotes the success rate of a task at step n. We define $Success\_rate_0 = 100.0\%$.

We present the results in Fig. 7. We also performed a paired t-test to examine the statistical differences between Tr of seen and unseen objects. Overall, though Tr_1, Tr_2, and Tr_3 all decreased when manipulating with unseen objects instead of the seen ones, we only saw significant statistical differences on Tr_1 in Task 1 and Task 2 ($p = 0.011$ and $p = 0.007$, Cohen's d effect sizes are 1.34 and 0.891, respectively). These results reveal that the biggest challenge for current VLA models when dealing with unseen objects remains to be accurately locating the correct object to manipulate, suggesting that the VLA models may not be able to recognize unseen objects in many cases. We did not observe significant statistical differences on Tr_1 in Task 3 and Task 4. A plausible explanation is due to the small number of passed test cases (~ 20 out of 1000) in both tasks.

Finding 11

Current VLA models struggled to recognize unseen objects, which was the primary cause of failure in most test scenes.

5.6 RQ6: How Robust Do VLA Models Perform Against Task Instruction Mutations?

To study this research question, for each task, we prompted GPT-4o to generate 10 mutations of the standard task instruction (Sec. 4.4) that conveyed the same meanings. We only generated a limited number of mutations so that we could manually verify each mutation. Two of the authors manually validated and confirmed that all generated mutations were semantically equivalent to the corresponding standard task instruction. We then re-executed all test scenes generated in RQ1 (Sec. 5.1) for each of the four tasks. For each test scene, we randomly applied one of the mutated task instructions. These mutated instructions can be found in our Git repository.

Table 7 presents the performance of seven subject VLA models when instructed with both the standard and mutated task instructions. When using the mutated task instructions, the average performance across the seven VLA models dropped by 32.8%, 1.7%, and 8.3% on Task 1, Task 2, and Task 3, respectively, while the performance difference on Task 4 was negligible.

Table 7. (**RQ6**) Performance of subject VLA models with standard (**Def.**) and mutated (**Mut.**) task instructions. Each cell represents the success rate of different VLA models in different tasks. Darker the color, the larger the negative differences between the performance on standard and mutated task instructions.

VLA Models	**Task 1: Pick Up**			**Task 2: Move Near**			**Task 3: Put On**			**Task 4: Put In**		
	Def.	Mut.	Diff.†	Def.	Mut.	Diff.†	Def.	Mut.	Diff.†	Def.	Mut.	Diff.†
RT-1-1k	0.7%	0.3%	-57.1%	1.4%	0.9%	-35.7%	0.0%	0.0%	—	0.0%	0.0%	—
RT-1-58k	28.2%	18.8%	-33.3%	10.9%	6.4%	-41.3%	0.0%	0.0%	—	0.0%	0.1%	—
RT-1-400k	34.4%	24.5%	-28.7%	9.4%	8.8%	-6.4%	0.5%	0.4%	-20.0%	0.1%	0.1%	—
RT-1-X	19.5%	10.1%	-48.2%	5.8%	5.6%	-3.4%	2.3%	1.5%	-34.8%	0.4%	0.5%	+25.0%
Octo-small	0.8%	0.1%	-87.5%	1.5%	1.5%	—	2.2%	2.1%	-4.5%	1.1%	0.4%	-63.6%
Octo-base	0.0%	0.3%	—	0.6%	0.4%	-33.3%	1.2%	1.5%	+25.0%	1.1%	0.5%	-54.5%
OpenVLA-7b	5.9%	6.4%	+8.5%	12.7%	12.2%	-0.4%	2.1%	2.0%	-4.8%	1.1%	2.0%	+81.8%
Avg.	12.8%	8.6%	-32.8%	6.0%	5.9%	-1.7%	1.2%	1.1%	-8.3%	0.5%	0.5%	—

† Diff. = (Mut. - Def.) / Def.

We also found that larger models, such as OpenVLA-7b, were more robust against task instruction mutations. Specifically, OpenVLA-7b performed even better with the mutated task instructions on Task 1 and Task 4. For Task 2 and Task 3, the performance was only marginally affected. This robustness may be attributed to OpenVLA-7b's use of Llama 2 as its language model, which potentially enhanced its language comprehension capabilities.

For instance, a mutated task instruction for Task 2 could be: `place [OBJECT A] near [OBJECT B]`. When prompted with this mutated instruction, OpenVLA-7b successfully manipulated 17.6% of the test scenes. In contrast, none of the other VLA models succeeded in more than 5% of the test scenes under the same instruction.

Finding 12

Current VLA models showed limited robustness against task instruction mutations. However, VLA models incorporating a large language model (e.g., OpenVLA-7b uses Llama 2-7b as the language model) are much more robust.

6 Discussion

Our study reveals several key implications that can lead to the development of better and more reliable VLA models for robotic manipulation. In this section, we discuss these implications and explore future research opportunities.

VLA models for robotic manipulation—too good to be true for now. While VLA models have the potential to revolutionize robotic manipulation, our experiment results in RQ1 and RQ2 reveal that the current models are still **unreliable** for common robotic manipulation tasks at the time of this study. Deploying VLA models in high-stakes and safety-critical applications remains impractical.

Our detailed analysis suggests that the deficiency mainly comes from the lack of capabilities in precisely interpreting complex task requirements and accurately localizing the correct target objects to manipulate. We believe there is a huge space for improvement in addressing these issues. One potential solution is to scale up the model size. Currently, even the largest model in our study, OpenVLA-7b, has only 7 billion parameters, which is significantly smaller than state-of-the-art (SOTA) models in other domains. For instance, one of the largest open-source LLMs, Llama 3.1, has 405 billion parameters. The closed-source LLMs such as GPT-4 and Claude-3.5 typically have even higher numbers of model parameters. By scaling up the model, we may also observe *emergent capabilities* for VLA models [8].

In addition to scaling up the size of VLA models, exploring more effective prompting strategies could be another promising direction. Previous studies have demonstrated that improved prompting can significantly enhance the performance of LLMs across various tasks, such as solving mathematical problems [83, 88] and code generation [11, 93]. Our Finding 2 suggests that current

VLA models struggle to decompose complex task instructions into multiple steps of action. To fix this, one may consider prompting the VLA model step by step. This also aligns with one of the popular prompting strategies in the field of LLMs – *chain-of-thought* prompting. While our study has only explored the most basic prompts, future research should explore more advanced prompting techniques to evaluate their impact on VLA model performance.

To demonstrate the potential of improving VLA models' performance through prompting, we crafted a new prompt for Task 4 by separating the task into two steps: `pick up [object name] and then put [object name] into [object name]`. We re-ran the 1000 test scenes generated in RQ1 (Sec. 5.1) for Task 4 using this new prompt and compared the performance with the original prompt. Our results in Table 8 show that with this new prompt, the success rate of grasping the correct object increased in three out of seven VLA models, while the success rate of moving the correct object improved in five out of seven models. The average success rate of moving the correct object increased from 0.6% to 6.2%. Although the overall task success rate did not increase, these results suggest that exploring prompting strategies for VLA models could be a promising research direction.

Table 8. Different prompts for Task 4.

Model	Original Prompt			New Prompt		
	Grasp	Move	Success	Grasp	Move	Success
RT-1-1k	1.1%	0.0%	0.0%	1.4%	0.0%	0.0%
RT-1-58k	1.3%	0.0%	0.0%	4.2%	0.6%	0.0%
RT-1-400k	8.9%	0.1%	0.1%	6.0%	0.1%	0.0%
RT-1-X	17.8%	0.7%	0.4%	28.0%	7.8%	0.4%
Octo-small	34.6%	1.1%	1.1%	30.0%	17.4%	0.0%
Octo-base	32.5%	1.1%	1.1%	22.1%	12.9%	0.2%
OpenVLA	19.9%	1.1%	1.1%	14.1%	4.4%	0.1%

Furthermore, one can also introduce multi-agent systems to split the robotic manipulation tasks among multiple VLA agents, which is a strategy widely used in other domains [84, 87, 94].

Addressing the robustness challenges. Our study results from RQ3 and RQ4 reveal that current VLA models lack robustness against several external factors, such as lighting conditions and camera poses. We also found that models with large-scale pre-training or those trained with larger datasets of robot demonstration data exhibit greater robustness compared to others. This highlights a promising research direction: enriching the robot demonstration data. Since manually collecting real-world robot demonstration data requires significant labeling efforts, researchers may consider leveraging data augmentation techniques [2, 5] or employing *sim2real* translation [32, 96] to scale up training data by utilizing simulation environments. For instance, in the future, one could leverage a well-designed traditional controller to guide the robot in solving a test scene. The robot's demonstration could then serve as data for re-training or fine-tuning.

Moreover, our Finding 6, Finding 7, and Finding 8 show that when external factors like lighting and camera angles deviate significantly from default settings, VLA model performance drops accordingly. This suggests that existing robot demonstration datasets may lack diversity, especially regarding variations in external conditions. In future work, when collecting datasets for training/fine-tuning factors, practitioners should take these environmental factors into account, which may potentially lead to the training of more robust VLA models.

Assessing the capabilities of VLA models. Our Finding 10 and Finding 11 (RQ5) suggest that current VLA models struggled to perform tasks with unseen objects. In practice, it is unrealistic to expect VLA models to successfully perform manipulation for every possible task scenario. Thus, in parallel to the development of more powerful and robust VLA models, it is also important to design novel techniques to comprehensively assess the capabilities of VLA models and derive proper guidelines about the use of VLA models. To address these, potential solutions include offline benchmarking and online risk assessment.

In terms of offline benchmarking, our 18,604 generated test scenes across four tasks can serve as one of the early benchmarks for VLA models. In future work, practitioners may focus on expanding this benchmark to include a broader range of robotic manipulation tasks and diverse robot settings, covering various object types, environmental factors, and task complexities. In terms of online risk assessment, one may consider adopting SOTA techniques for risk assessment in other domains, such as uncertainty estimation [38, 54] and safety monitoring [86]. While general techniques may be directly used in the context of robotic manipulation with VLA models, it is unclear to what extent they can help with assessing the quality and reliability of decisions made by VLA models.

Towards efficient testing for VLA models. Although our proposed VLATest successfully identified numerous failed test scenes across our subject VLA models, it also incurred significant time overheads. Specifically, we relied on a *random sampler* for sampling target objects and confounding objects in the testing scenes, which may not have been the most efficient approach. Future research could focus on optimizing the $pose_sampler$ in Algorithm 1 to strategically assign critical positions and orientations for objects and confounding objects. For instance, one can consider metamorphic-based methods [24, 81, 90] or search-based methods [13, 97] to efficiently generate test scenes for VLA models. Meanwhile, researchers may also work on test prioritization [14, 22, 47] or test selection [1, 23, 34, 37] towards efficient testing for VLA models.

Nevertheless, as VLA models continue to evolve, we argue that testing strategies must also evolve in parallel. The complexity and capabilities of these models are likely to increase, requiring more sophisticated and adaptive testing frameworks that can keep pace with advancements.

7 Related Work

Foundation Models for Robotics. A large body of research has been done on the use of foundation models for robotics. Based on the data modality, recent work can be roughly categorized into two groups: (1) those that use LLMs for robotics and (2) those that use multi-modality foundation models (e.g., VLMs) for robotics. One of the early attempts in using LLMs for robotics is to leverage LLMs as a reward designer for Deep Reinforcement Learning-based robotic manipulation [68]. This work proposes a self-refined framework to let the LLM automatically generate and refine reward functions used to train policies for robotic control. Alternatively, Zhou et al. [98] adopts LLMs to convert natural language input into a Planning Domain Definition Language formulation before outputting action plan sequences.

Different from the use of LLMs, multi-modality foundation models such as VLMs enable the possibility of advancing AI-enabled robotics in several novel tasks, e.g., for robotic manipulation [7, 8, 75], visual question answering with robots [19], and visual state representations [56]. Our work is most closely related to those using vision-language-action models, a specialized category of VLMs, for robotic manipulation [7, 8, 40, 45, 46, 71, 75]. One of the pioneer works in VLA models for robotic manipulation is RT-1 [7], which uses a combination of a FiLM EfficientNet and a transformer to learn control policies from 130k real-world robot demonstrations. Since the release of the Open X-Embodiment dataset [59], a series of VLA models have been proposed by either training or fine-tuning on Open X-Embodiment [8, 40, 46, 75]. Our work is parallel to these works. We propose a general testing framework to test VLA models and present an empirical study to comprehensively assess the performance of VLA models.

Testing Cyber-Physical Systems. Quality assurance of traditional CPS and AI-driven CPS is a vital topic in software engineering, as it ensures the safety and trustworthiness of such complex systems in safety-critical domains. Substantial efforts have been made by researchers and industrial practitioners to safeguard the quality of CPS from various perspectives, such as testing [3, 12, 43, 53, 95], analysis [6, 70, 89] and repairing [28, 67, 80]. Considering robotic manipulation as a widely deployed representative CPS, our work is most related to the testing of CPS.

In particular, Lee et al. [43] propose MOTIF, a gray-box fuzzing-based approach to generate test data for software deployed in CPS. This method monitors the coverage achieved by the vanilla and the mutated functions, thereby exploring the behavior space of the software under test. Differing from the existing testing solutions, MOTIF specifically targets the testing challenges of C and C++ languages that are widely used in CPS contexts. In terms of the black-box approach, Menghi et al. [53] introduce ARIsTEO, a search-based testing method with a loop of approximation and refinement to identify requirement violations of CPS models, and Chen et al. [12] propose an active fuzzing approach to find test suites for packet-level CPS network attacks. Moving forward with CPS with an AI module embedded, Zhang et al. [95] design a series of time-aware coverage criteria for DNN controllers in CPS and develop a falsification framework, FalsifAI, to find system defects by leveraging the coverage information from the proposed criteria. It is worth noting that the aforementioned works are not well applicable to VLA models, which is attributable to the multimodality characteristic, the autoregressive generation mechanism, and the large-scale model size. In contrast, our study initiates a testing framework tailored for VLA models in the context of different robotic manipulation tasks. VLATest encompasses four multimodality-aware testing operators to comprehensively assess the VLA models from different perspectives.

Benchmarking and Testing Foundation Models. After the remarkable success of foundation models, an important research direction is to explore and understand the boundaries of their capabilities, which can provide a basis for further safeguarding and enhancement. Based on this motivation, numerous studies have focused on evaluating large language models in the text domain across various properties, such as correctness [49], factuality [21, 50], robustness [79, 99], fairness [72], and privacy [29]. These studies offer a wealth of resources for the general performance assessment of LLMs. On the other hand, researchers have also made significant efforts to evaluate specific capabilities of LLMs, such as code-related abilities. These efforts include validating the correctness of generated code [10, 52, 91], examining whether LLMs can address real-world GitHub issues [39], and analyzing error patterns of LLM-generated code [60, 82].

While the aforementioned studies provide valuable insights, most of them focus on performing analysis on static benchmarks. Orthogonal to these related works, there are also several attempts at dynamically generating test cases to evaluate one or more key properties of models. For example, existing studies use rule-based methods to generate test cases that measure bias [78] or linguistic capabilities [42], employ metamorphic testing for evaluating language translation [31, 73], and utilize mutation-based frameworks to assess robustness [85]. Our work differs from these previous approaches in two main ways: (1) our framework centers on VLA, a novel architecture that integrates multi-modal inputs and generates complex control commands, distinguishing it from language-focused foundation models; (2) our framework generates test cases that account for complex interactions in robotic manipulation within a 3D environment, presenting a new challenge compared to prior text-only testing frameworks.

8 Threats to Validity

We discuss the threats to the validity of our empirical study results, as well as the mitigating factors.

Internal Validity. One potential threat comes from the randomness of our experiments. To mitigate this, we generated 18,604 testing scenes in our empirical study. Overall, our empirical study took over 580 GPU hours to finish. While we only executed each testing scene once per VLA model, we believe this has limited impact on our findings as our goal is to evaluate the VLA model's overall performance on each task instead of the success of one specific testing scene.

Another potential threat to internal validity is the choice of prompt templates. To mitigate this, we followed the previous works [7, 8, 40, 75] to use the standard prompt template when evaluating VLA models for robotic manipulation in RQ1 ~ RQ5. In RQ6, we evaluated the VLA models' performance with the mutated prompt templates. However, the prompt templates mutated by GPT-4o may not carry the same meanings as the original ones. To mitigate the threat, two of the authors manually validated that the mutated prompts were semantically equivalent to the corresponding standard prompts. In addition, we discuss the impact of prompts and the potential of improving VLA models with better prompting in Sec. 6.

External Validity. Threats to external validity include whether our study results can generalize to different experimental settings, such as different robotic manipulation tasks and different VLA models. To mitigate these threats, we select the four most popular robotic manipulation tasks according to the Open X-Embodiment dataset [59]. In terms of the VLA models, we included the SOTA publicly available VLA models by the time we conducted this study. Given that the VLA model for robotic manipulation is a fast-developing research area, we also plan to extend our evaluation to other SOTA VLA models, e.g., RT-2 [8], once they are made available.

Finally, we have only implemented VLATest and evaluated the VLA models within one simulation environment, Maniskill2 [26]. To reduce the impact of distribution shifts between simulation and real robot execution, we chose an improved version of Maniskill2 [44], which provides *visual matching* to render realistic images as the visual input for VLA models.

Construct Validity. When designing our testing framework, VLATest, we have only considered a limited number of testing operators. To combat the threat, we included testing operators covering different aspects, e.g., task difficulties (i.e., number of confounding objects) and environmental factors (i.e., the lighting intensity and camera poses). In future work, one may continue to improve VLATest by adding more testing operators, such as the number of lighting sources, the intrinsic parameters of cameras, and the resolution of cameras, for a more comprehensive evaluation of VLA models. Another threat to construct validity is the coverage metric we used in RQ1. Although our trajectory coverage metric may not be the best metric for measuring test sufficiency, we believe it is a pragmatic choice since there is no well-established coverage metric for either VLA models or robotic manipulation. In future work, one may consider designing a new coverage metric given the special characteristics of robotic manipulation with VLA models.

9 Conclusion

In this paper, we propose VLATest, one of the early testing frameworks for testing VLA models for robotic manipulation. VLATest is a generation-based fuzzing framework based on ten operators covering different perspectives in the test scene of robotic manipulation. Upon implementing VLATest with Maniskill2 simulation environments, we further conducted a large-scale empirical study to assess the performance and robustness of seven popular VLA models across four robotic manipulation tasks. We generated 18,604 testing scenes, conducted more than 580 GPU hours of simulation, and performed a detailed analysis of the challenges and limitations faced by the current VLA models. At the end of the paper, we discuss the implications of our study, shedding light on several future research directions to improve the quality and reliability of VLA models.

10 Data Availability

Our artifacts, including the replication packages and the generated testing scenes, are available on a Git repository: https://github.com/ma-labo/VLATest.

Acknowledgments

This work was supported in part by the Canada CIFAR AI Chairs Program, Natural Sciences and Engineering Research Council of Canada, JST CRONOS Grant (No. JPMJCS24K8), JSPS KAKENHI Grant (No. JP21H04877, No. JP23H03372, and No. JP24K02920), and the Autoware Foundation.

References

[1] Zohreh Aghababaeyan, Manel Abdellatif, Mahboubeh Dadkhah, and Lionel Briand. 2024. DeepGD: A Multi-Objective Black-Box Test Selection Approach for Deep Neural Networks. *ACM Transactions on Software Engineering and Methodology* 33, 6 (2024), 1–29.

[2] Khaled Alomar, Halil Ibrahim Aysel, and Xiaohao Cai. 2023. Data Augmentation in Classification and Segmentation: A Survey and New Strategies. *Journal of Imaging* 9, 2 (2023), 46.

[3] Jon Ayerdi, Valerio Terragni, et al. 2021. Generating metamorphic relations for cyber-physical systems with genetic programming: an industrial case study. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1264–1274.

[4] Yuntao Bai, Andy Jones, Kamal Ndousse, et al. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862* (2022).

[5] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. 2024. RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4788–4795.

[6] Matteo Biagiola and Paolo Tonella. 2024. Testing of Deep Reinforcement Learning Agents with Surrogate Models. *ACM Transactions on Software Engineering and Methodology* 33, 3 (2024), 1–33.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, et al. 2022. RT-1: Robotics Transformer for Real-World Control at Scale. *arXiv preprint arXiv:2212.06817* (2022).

[8] Anthony Brohan, Noah Brown, Justice Carbajal, et al. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv preprint arXiv:2307.15818* (2023).

[9] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. 2015. The YCB object and Model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*. IEEE, 510–517.

[10] Federico Cassano, John Gouwar, Daniel Nguyen, et al. 2023. MultiPL-E: A Scalable and Extensible Approach to Benchmarking Neural Code Generation. *IEEE Transactions on Software Engineering* 49, 7 (2023), 3675–3691.

[11] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching Large Language Models to Self-Debug. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=KuPixIqPiq

[12] Yuqi Chen, Bohan Xuan, Christopher M Poskitt, Jun Sun, and Fan Zhang. 2020. Active Fuzzing for Testing and Securing Cyber-Physical Systems. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 14–26.

[13] Mingfei Cheng, Yuan Zhou, and Xiaofei Xie. 2023. BehAVExplor: Behavior Diversity Guided Testing for Autonomous Driving Systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 488–500.

[14] Xueqi Dang, Yinghua Li, Mike Papadakis, Jacques Klein, Tegawendé F Bissyandé, and Yves Le Traon. 2024. Test Input Prioritization for Machine Learning Classifiers. *IEEE Transactions on Software Engineering* (2024).

[15] Amandeep Dhaliwal. 2020. The Rise of Automation and Robotics in Warehouse Management. In *Transforming Management Using Artificial Intelligence Techniques*. CRC Press, 63–72.

[16] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. 2023. Task and Motion Planning with Large Language Models for Object Rearrangement. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2086–2092.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[18] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, et al. 2023. PaLM-E: An Embodied Multimodal Language Model. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 8469–8488.

[19] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. 2023. Vision-Language Models as Success Detectors. In *Proceedings of The 2nd Conference on Lifelong Learning Agents (Proceedings of Machine Learning Research, Vol. 232)*. PMLR, 120–136.

[20] Andrius Dzedzickis, Jurga Subačiūtė-Žemaitienė, Ernestas Šutinys, Urtė Samukaitė-Bubnienė, and Vytautas Bučinskas. 2021. Advanced applications of industrial robotics: New trends and possibilities. *Applied Sciences* 12, 1 (2021), 135.

[21] Nouha Dziri, Ehsan Kamalloo, Sivan Milton, Osmar Zaiane, Mo Yu, Edoardo M Ponti, and Siva Reddy. 2022. FaithDial: A Faithful Benchmark for Information-Seeking Dialogue. *Transactions of the Association for Computational Linguistics* 10 (2022), 1473–1490.

[22] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. DeepGini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 177–188.

[23] Xinyu Gao, Yang Feng, Yining Yin, Zixi Liu, Zhenyu Chen, and Baowen Xu. 2022. Adaptive test selection for deep neural networks. In *Proceedings of the 44th International Conference on Software Engineering*. 73–85.

[24] Xinyu Gao, Zhijie Wang, Yang Feng, Lei Ma, Zhenyu Chen, and Baowen Xu. 2024. MultiTest: Physical-Aware Object Insertion for Testing Multi-sensor Fusion Perception Systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[25] Ruchi Goel and Pooja Gupta. 2020. *Robotics and Industry 4.0.* Springer International Publishing, Cham, 157–169.

[26] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, et al. 2023. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. *arXiv preprint arXiv:2302.04659* (2023).

[27] Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. 2023. Recent Trends in Task and Motion Planning for Robotics: A Survey. *Comput. Surveys* 55, 13s (2023), 1–36.

[28] Zhengang Guo, Yingfeng Zhang, Xibin Zhao, and Xiaoyu Song. 2020. CPS-Based Self-Adaptive Collaborative Control for Smart Production-Logistics Systems. *IEEE transactions on cybernetics* 51, 1 (2020), 188–198.

[29] Aamir Hamid, Hemanth Reddy Samidi, Tim Finin, Primal Pappachan, and Roberto Yus. 2023. GenAIPABench: A Benchmark for Generative AI-based Privacy Assistants. *arXiv preprint arXiv:2309.05138* (2023).

[30] Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. 2023. A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation. *Sensors* 23, 7 (2023), 3762.

[31] Pinjia He, Clara Meister, and Zhendong Su. 2020. Structure-Invariant Testing for Machine Translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 961–973.

[32] Sebastian Höfer, Kostas Bekris, Ankur Handa, et al. 2021. Sim2Real in Robotics and Automation: Applications and Challenges. *IEEE Transactions on Automation Science and Engineering* 18, 2 (2021), 398–400.

[33] Jane Holland, Liz Kingston, Conor McCarthy, Eddie Armstrong, Peter O'Dwyer, Fionn Merz, and Mark McConnell. 2021. Service Robots in the Healthcare Sector. *Robotics* 10, 1 (2021), 47.

[34] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Mike Papadakis, Lei Ma, and Yves Le Traon. 2023. Aries: Efficient Testing of Deep Neural Networks via Labeling-Free Accuracy Estimation. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1776–1787.

[35] Zhisheng Hu, Shengjian Guo, Zhenyu Zhong, and Kang Li. 2021. Coverage-based Scene Fuzzing for Virtual Autonomous Driving Testing. *arXiv preprint arXiv:2106.00873* (2021).

[36] Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. 2023. Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model. *arXiv preprint arXiv:2305.11176* (2023).

[37] Yuheng Huang, Jiayang Song, Qiang Hu, Felix Juefei-Xu, and Lei Ma. 2024. Active Testing of Large Language Model via Multi-Stage Sampling. *arXiv preprint arXiv:2408.03573* (2024).

[38] Yuheng Huang, Jiayang Song, Zhijie Wang, et al. 2025. Look Before You Leap: An Exploratory Study of Uncertainty Analysis for Large Language Models. *IEEE Transactions on Software Engineering* 51, 2 (2025), 413–429.

[39] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *The Twelfth International Conference on Learning Representations*.

[40] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246* (2024).

[41] Maria Kyrarini, Fotios Lygerakis, Akilesh Rajavenkatanarayanan, Christos Sevastopoulos, Harish Ram Nambiappan, Kodur Krishna Chaitanya, Ashwin Ramesh Babu, Joanne Mathew, and Fillia Makedon. 2021. A Survey of Robots in Healthcare. *Technologies* 9, 1 (2021), 8.

[42] Jaeseong Lee, Simin Chen, Austin Mordahl, Cong Liu, Wei Yang, and Shiyi Wei. 2024. Automated Testing Linguistic Capabilities of NLP Models. *ACM Transactions on Software Engineering and Methodology* (2024).

[43] Jaekwon Lee, Enrico Viganò, Oscar Cornejo, Fabrizio Pastore, and Lionel Briand. 2023. Fuzzing for CPS Mutation Testing. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1377–1389.

[44] Xuanlin Li, Kyle Hsu, Jiayuan Gu, et al. 2024. Evaluating Real-World Robot Manipulation Policies in Simulation. In *8th Annual Conference on Robot Learning*.

[45] Xinghang Li, Minghuan Liu, Hanbo Zhang, et al. 2024. Vision-Language Foundation Models as Effective Robot Imitators. In *The Twelfth International Conference on Learning Representations*.

[46] Xiang Li, Cristina Mata, Jongwoo Park, et al. 2025. LLaRA: Supercharging Robot Learning Data for Vision-Language Policy. In *The Thirteenth International Conference on Learning Representations*.

[47] Yinghua Li, Xueqi Dang, Lei Ma, Jacques Klein, and Tegawendé F Bissyandé. 2024. Prioritizing test cases for deep learning-based video classifiers. *Empirical Software Engineering* 29, 5 (2024), 111.

[48] Jacky Liang, Wenlong Huang, et al. 2023. Code as Policies: Language Model Programs for Embodied Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9493–9500.

[49] Percy Liang, Rishi Bommasani, Tony Lee, et al. 2023. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research* (2023).

[50] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 3214–3252.

[51] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual Instruction Tuning. *Advances in neural information processing systems* 36 (2024).

[52] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2024. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. *Advances in Neural Information Processing Systems* 36 (2024).

[53] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. 2020. Approximation-Refinement Testing of Compute-Intensive Cyber-Physical Models: An Approach Based on System Identification. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 372–384.

[54] Subhabrata Mukherjee and Ahmed Awadallah. 2020. Uncertainty-aware Self-training for Few-shot Text Classification. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 21199–21212.

[55] Richard M Murray, Zexiang Li, and S Shankar Sastry. 2017. *A Mathematical Introduction to Robotic Manipulation*. CRC press.

[56] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. 2023. R3M: A Universal Visual Representation for Robot Manipulation. In *Proceedings of The 6th Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 205)*. PMLR, 892–909.

[57] Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. 2023. DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research* (2023).

[58] James Orr and Ayan Dutta. 2023. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey. *Sensors* 23, 7 (2023), 3625.

[59] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, et al. 2023. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. *arXiv preprint arXiv:2310.08864* (2023).

[60] Rangeet Pan, Ali Reza Ibrahimzada, et al. 2024. Lost in Translation: A Study of Bugs Introduced by Large Language Models while Translating Code. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*.

[61] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. FiLM: Visual Reasoning with a General Conditioning Layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[62] Abu Rayhan. 2023. Artificial intelligence in robotics: From automation to autonomous systems.

[63] Olga Russakovsky, Jia Deng, Hao Su, et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International journal of computer vision* 115 (2015), 211–252.

[64] Mohamed Shamout, Rabeb Ben-Abdallah, et al. 2022. A Conceptual Model for the Adoption of Autonomous Robots in the Supply Chain and Logistics Industry. *Uncertain Supply Chain Management* 10, 2 (2022), 577–592.

[65] B Siciliano. 2008. *Springer Handbook of Robotics*. Springer.

[66] Ishika Singh, Valts Blukis, Arsalan Mousavian, et al. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11523–11530.

[67] Jiayang Song, Xuan Xie, and Lei Ma. 2023. SIEGE: A Semantics-Guided Safety Enhancement Framework for AI-Enabled Cyber-Physical Systems. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4058–4080.

[68] Jiayang Song, Zhehua Zhou, Jiawei Liu, Chunrong Fang, Zhan Shu, and Lei Ma. 2023. Self-Refined Large Language Model as Automated Reward Function Designer for Deep Reinforcement Learning in Robotics. *arXiv preprint arXiv:2309.06687* (2023).

[69] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. 2023. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cognitive Robotics* 3 (2023), 54–70.

[70] Andrea Stocco and Paolo Tonella. 2022. Confidence-driven weighted retraining for predicting safety-critical failures in autonomous driving systems. *Journal of Software: Evolution and Process* 34, 10 (2022), e2386.

[71] Austin Stone et al. 2023. Open-World Object Manipulation using Pre-trained Vision-Language Models. In *Proceedings of The 7th Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 229)*. PMLR, 3397–3417.

[72] Lichao Sun, Yue Huang, et al. 2024. TrustLLM: Trustworthiness in Large Language Models. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*. PMLR, 20166–20270.

[73] Zeyu Sun, Jie M Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic Testing and Improvement of Machine Translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 974–985.

[74] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114.

[75] Octo Model Team, Dibya Ghosh, Homer Walke, et al. 2024. Octo: An Open-Source Generalist Robot Policy. *arXiv preprint arXiv:2405.12213* (2024).

[76] Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288* (2023).

[77] Manas Wakchaure, BK Patle, and AK Mahindrakar. 2023. Application of AI techniques and robotics in agriculture: A review. *Artificial Intelligence in the Life Sciences* 3 (2023), 100057.

[78] Yuxuan Wan, Wenxuan Wang, Pinjia He, Jiazhen Gu, Haonan Bai, and Michael R Lyu. 2023. BiasAsker: Measuring the Bias in Conversational AI System. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 515–527.

[79] Boxin Wang, Weixin Chen, Hengzhi Pei, et al. 2023. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[80] Renzhi Wang, Zhehua Zhou, Jiayang Song, Xuan Xie, Xiaofei Xie, and Lei Ma. 2024. MORTAR: A Model-based Runtime Action Repair Framework for AI-enabled Cyber-Physical Systems. *arXiv preprint arXiv:2408.03892* (2024).

[81] Shuai Wang and Zhendong Su. 2020. Metamorphic Object Insertion for Testing Object Detection Systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1053–1065.

[82] Zhijie Wang, Zijie Zhou, Da Song, Yuheng Huang, Shengmai Chen, Lei Ma, and Tianyi Zhang. 2025. Towards Understanding the Characteristics of Code Generation Errors Made by Large Language Models. In *Proceedings of the IEEE/ACM 47th International Conference on software Engineering (ICSE '25)*.

[83] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 24824–24837.

[84] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying LLM-based Software Engineering Agents. *arXiv preprint arXiv:2407.01489* (2024).

[85] Mingxuan Xiao, Yan Xiao, et al. 2023. LEAP: Efficient and Automated Test Method for NLP Software. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*. 1136–1148.

[86] Xuan Xie, Jiayang Song, Zhehua Zhou, Yuheng Huang, Da Song, and Lei Ma. 2024. Online Safety Analysis for LLMs: a Benchmark, an Assessment, and a Path Forward. *arXiv preprint arXiv:2404.08517* (2024).

[87] John Yang, Carlos E. Jimenez, et al. 2024. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In *Advances in Neural Information Processing Systems*, Vol. 37. Curran Associates, Inc., 50528–50652.

[88] Shunyu Yao, Dian Yu, Jeffrey Zhao, et al. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 11809–11822.

[89] Rajaa Vikhram Yohanandhan, Rajvikram Madurai Elavarasan, et al. 2020. Cyber-Physical Power System (CPPS): A Review on Modeling, Simulation, and Analysis With Cyber Security Applications. *IEEE Access* 8 (2020), 151019–151064.

[90] Boxi Yu, Zhiqing Zhong, Xinran Qin, Jiayi Yao, Yuancheng Wang, and Pinjia He. 2022. Automated testing of image captioning systems. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 467–479.

[91] Hao Yu, Bo Shen, Dezhi Ran, Jiaxin Zhang, Qi Zhang, Yuchi Ma, Guangtai Liang, Ying Li, Qianxiang Wang, and Tao Xie. 2024. CoderEval: A Benchmark of Pragmatic Code Generation with Generative Pre-trained Models. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–12.

[92] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid Loss for Language Image Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11975–11986.

[93] Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. 2023. Self-Edit: Fault-Aware Code Editor for Code Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers*. Association for Computational Linguistics, 769–787.

[94] Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. 2024. AutoCodeRover: Autonomous Program Improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 1592–1604.

[95] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. 2022. FalsifAI: Falsification of AI-Enabled Hybrid Control Systems Guided by Time-Aware Coverage Criteria. *IEEE Transactions on Software Engineering* 49, 4 (2022), 1842–1859.

[96] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. 2020. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 737–744.

[97] Yuan Zhou, Yang Sun, Yun Tang, et al. 2023. Specification-Based Autonomous Driving System Testing. *IEEE Transactions on Software Engineering* 49, 6 (2023), 3391–3410.

[98] Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024. ISR-LLM: Iterative Self-Refined Large Language Model for Long-Horizon Sequential Task Planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2081–2088.

[99] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, et al. 2024. PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis* (Salt Lake City, UT, USA) *(LAMPS '24)*. 57–68.

Received 2024-09-11; accepted 2025-04-01