From Mystery to Mastery: Failure Diagnosis for Improving Manipulation Policies

Anonymous authors

***Abstract*—Robot manipulation policies often fail for unknown reasons, posing significant hurdles for safe real-world deployment. Existing diagnostic approaches rely on hand-crafted rules and exhaustive scenario enumeration, which are labor-intensive and prone to missing critical failure modes (FM). Reinforcement learning (RL) offers a systematic way to explore high-dimensional environmental variations by optimizing policies toward failure-inducing conditions. We present Robot Manipulation Diagnosis (RoboMD), a policy-agnostic framework that leverages deep RL over a learned vision–language embedding to efficiently search the failure space. Rather than manually specifying each scenario, RoboMD produces a probabilistic failure-likelihood map, highlighting high-risk conditions without exhaustive enumeration. Evaluations on diverse simulation benchmarks and a robot arm show that RoboMD discovers up to 23% more FM than state-of-the-art vision–language models and uncovers subtle vulnerabilities missed by heuristic methods. By mapping failures in embedding space, RoboMD provides actionable insights to guide targeted policy improvements and enhance robustness in unseen environments.**

I. Introduction

To deploy a robot in the real-world, it must be robust enough to operate under diverse and often unpredictable variations of its intended environment. For instance, a robot picking up a cup should adapt to variations in cup shape, size, and material; operate seamlessly in rooms with differing layouts and surfaces; and remain consistent under varying lighting conditions [1, 2]. However, no matter how well we train a robot model, it will fail under some operational conditions in the physical world [3]. Addressing these unknown failures is not merely a matter of bookkeeping specific errors, but it also requires a fundamental shift toward diagnosing failures before deployment and leveraging these insights for more efficient policy improvement. Without such systematic frameworks, robotic systems are unlikely to achieve the reliability needed for seamless real-world deployment.

High-dimensional manipulation tasks are especially prone to policy failures arising from unanticipated environmental variations, hereafter referred to as *unseen* environments. The intricate interactions between policies and their environments produce a vast range of potential failure modes (FMs) [1, 2]. Manually hypothesizing which environmental perturbations may cause failures often overlooks subtle but critical edge cases, and exhaustively evaluating every potential variation is computationally infeasible. Consequently, we require scalable methods that can explore this high-dimensional space and systematically uncover failure modes. To this end, instead of relying on less-scalable brute-force search for failure diagnosis [1], we propose learning a deep RL policy to efficiently diagnose failures in manipulation tasks across diverse environment variations.

While our deep RL-based method efficiently captures many failures, its discrete action space, representing a predefined set of potential FMs, cannot account for failures that emerge under *unseen* environmental conditions. For instance, as illustrated in Fig. 1, if we set the action space of RL to orange fanta bottle, green sprite bottle, and white milk carton, it will quantify FM probabilities of each, but it is not able to predict anything beyond these three conditions—for instance, if picking up a red cuboid will fail or not. To generalize failure diagnosis for such unseen environment conditions, we design and train a vision-language model (VLM) embedding that represents failures and successes in manipulation tasks. This embedding provides an abstract yet semantically meaningful representation of manipulation policy performance. Our failure diagnostic deep RL agent, RoboMD, explores failure modes with a *continuous action space on this embedding*. Taking an action on this embedding is akin to exploring an arbitrary environment condition, allowing RoboMD to explore and learn the failure probabilities without requiring expensive full-environment rollouts. Unlike prior methods that query standard VLMs to determine task failure [4], our framework learns failures on a VLM embeddings. For instance, RoboMD identified 23.3% more failures in behavioral cloning policies for the cube picking up task compared to the best performing VLM, Gemini 1.5 Pro. This gain stems from the RoboMD's ability to actively navigate the failure space. Furthermore, RoboMD quantifies and ranks failure likelihoods, providing actionable insights to guide policy fine-tuning efforts. The main contributions of the paper are:

1) A deep RL-based framework for diagnosing failures in pre-trained manipulation policies.
2) Continuous exploration via a vision–language embedding to uncover and quantify failures in unseen environments.
3) Systematic policy improvement driven by the diagnosed failures.

II. Related work

Failure detection in large models can be characterized by querying vision-language foundation models [5, 4, 6, 7, 8] or searching for failures [9]. As we further verify in experiments, the former does not show strong performance in deciphering failures as they do not iteratively interact with the policy.

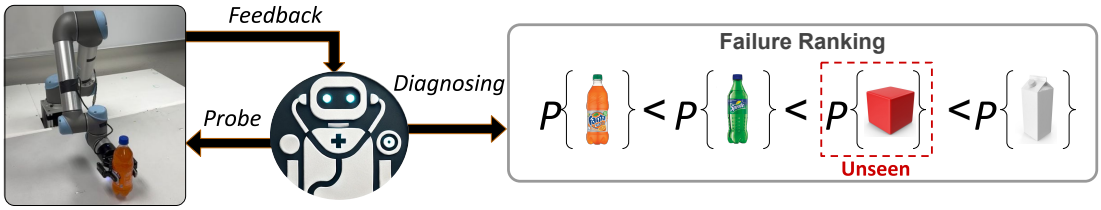

Fig. 1: Overview of the RoboMD failure diagnosis pipeline. Given a pre-trained manipulation policy (left), RoboMD systematically perturbs environment conditions and collects binary success/failure feedback. From this feedback it computes failure-mode probabilities and produces a ranked ordering (top right), highlighting both seen and unseen modes (red dashed box). Bottom panels show example perturbations in simulation (blue) and their validation in the real world (green).

Further, VLM models are not yet capable of making highly accurate quantitative predictions such as probabilities. In the latter approach, outside of robotics, deep RL has recently been employed in machine learning to identify errors in image classification and generation [9]. [10, 11] utilized RL to explore challenging rainy conditions. [12] highlights the role of sequential decision making models in ensuring the safety of black-box systems. Not only these methods are not considered in realistic physical systems such as manipulation but also they are not able to generalize beyond a fixed set of known failures.

Out-of-distribution (OOD) detection methods can also be used to identify unseen inputs, for instance, in automotive perception [13], runtime policy monitoring [5], and regression [14]. However, OOD is not the same as failure, as not all OOD samples result in failure, and failures can occur in-distribution. We aim to characterize failures both within and beyond the training distribution, not merely flag OOD instances. A related area of research is uncertainty quantification, which underpins many OOD detection methods. While many attempts have been made to characterize the epistemic uncertainty [15], the unknown unknowns, in robot perception systems [16, 17], only a few attempts have been made to address this challenge in deep RL [18] and imitation learning [19, 20, 21]. As robot policy models grow increasingly complex, formally characterizing epistemic uncertainty becomes extremely challenging. Even if we can, such techniques do not inform engineers where the models fail, making it harder to further improve the policies.

Generalized robots are less prone to failures. Toward achieving this goal, generalization in robotics has been extensively studied to enable robots to adapt to diverse and unforeseen scenarios. Large-scale simulation frameworks have been developed to evaluate the robustness of robotic policies across varied tasks and environmental conditions [1, 22]. Vision-language-action models trained on multimodal datasets have demonstrated significant advancements in improving adaptability to real-world scenarios [23, 24]. Additionally, approaches such as curriculum learning and domain randomization have proven effective in enhancing generalization by exposing models to progressively complex or randomized environments [25]. These methodologies collectively address the challenges of policy robustness. Others have tackled safety from control-theoretic [26, 27], human-factors [28], statistical-certification [29, 30, 31, 32], and formal-methods [33] perspectives. While these enhance robustness, our framework diagnoses failures pre-deployment to help guide policy improvement.

III. Methodology

In this section, we introduce Robot Manipulation Diagnosis (RoboMD), a failure diagnosis methodology designed to be agnostic to the underlying training method of the manipulation policy. Whether the policy is trained via behavioral cloning, reinforcement learning, diffusion processes, foundation models, or any future methods, RoboMD operates solely based on policy rollouts, making it adaptable to a wide range of robot manipulation policies. An overview is depicted in Fig. 2.

A. Failure Diagnosis on Candidate Environment Variations

We now lay the foundation to our methodology by searching failures over a set of potential failures, which we generalize in the next sections. In practice, candidates of failure sets, $\mathcal{C}$, can be a combination of historical failures in robot manipulation as well as engineers' know-how and apprehensions. For instance, we know manipulation policies are generally sensitive to lighting conditions, background table colors, etc. However, since we do not know how this large set of potential failures exactly affect the pre-trained manipulation policy, we search this discrete space using deep RL. RL offers a systematic approach to exploring the action space by optimizing the selection of actions based on their potential to induce failures. See Appendix II for a detailed rationale for choosing RL.

The failure-diagnosis process is modeled as an Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$, where $\mathcal{S}$ consists of visual inputs of the robotic environment; $\mathcal{A} = \mathcal{C}$ is the set of discrete changes to the environment. For instance, changing a red table to blue is an action. (See Sec. III-B for continuous action representations); $\mathcal{P}(s'|s,a)$ is given by the physics engine/real robot (with stochastic noise). The reward $R(s,a) = C_{\text{failure}}$ if failure, otherwise $R(s,a) = -C_{\text{success}}\, t$, where C_{success}, C_{failure} are constants and t is the horizon of the manipulation policy. We set the discount factor to $\gamma = 0.99$

We expect RoboMD to gradually modify the environment by applying a finite number of predefined actions $(a_1, a_2, \ldots, a_n)$ to induces failures. For example, the sequence of actions could be: *change table color to black* $\rightarrow$ *adjust light level to 50%* $\rightarrow$ *set table size to X*, resulting in an environment with a black table of size X under 50% lighting conditions.

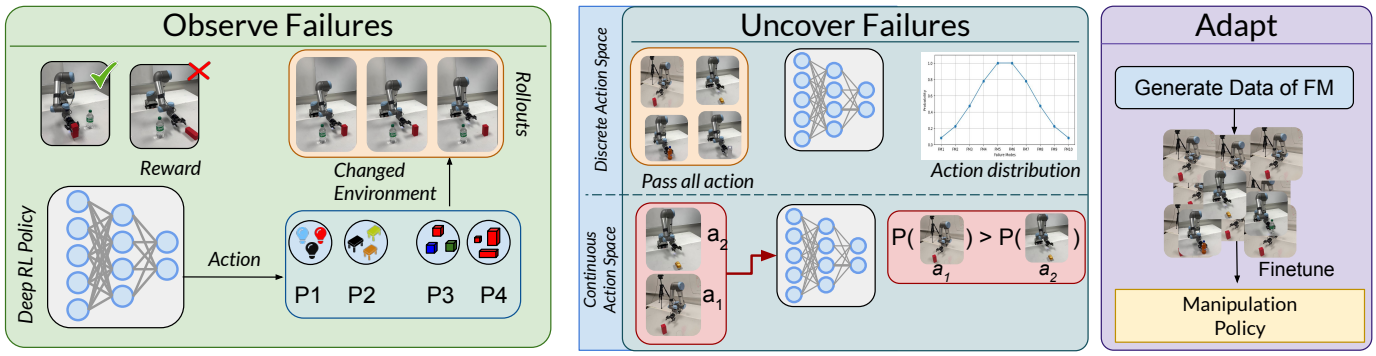

Fig. 2: RoboMD operates in three stages: (1) a PPO-based agent perturbs the environment to reveal configurations that cause failures under the pre-trained policy; (2) its learned action distribution—conditioned on the input observation—is converted into failure-mode probabilities, either directly over discrete candidates or via a continuous embedding for novel changes; and (3) those probabilities are used to fine-tune the manipulation policy.

Given our need for generalization to continuous action spaces in the forthcoming Section III-B, and the necessity of exploring diverse scenarios, which will be validated through experiments, we employ Proximal Policy Optimization (PPO) [34] as the learning algorithm. Additionally, the RoboMD policy π^{MD} is trained by interacting with the environment via $a_t \sim \pi^{\text{MD}}(a_t \mid s_t)$, receiving rewards for reaching failures. PPO refines π^{MD} toward failure-inducing variations. After training, RoboMD estimates the failure probability for each candidate FM in $\mathcal{C}$ (see Section III-C).

B. Generalizing Failure Diagnosis for Unseen Environments

In Section III-A, we apply discrete actions in $\mathcal{A} = \mathcal{C}$ to perturb the environment. but this limits failure discovery to known candidates. We argue that in order to predict failures of unseen environments, we need at least two pieces of information: 1) some belief of where failures might occur and 2) semantic similarity between an unseen environment and our belief on failures. For the former, we can utilize the candidate failure set $\mathcal{C}$, and for the latter we train a vision-language embedding that can be used to interpret failures. With these two, we train π^{MD} policy on the *continuous action space of the trained embedding*. Even if $\mathcal{C}$ does not fully cover the entire space of failures, π^{MD} is now capable of searching over $\mathcal{C}'$ because it operates on a projected continuous space, as opposed to the explicit discrete action space in Section III-A. In other words, this can be thought of as projecting a few expensive rollout samples to an embedding, and performing many cheap RoboMD RL evaluations on this projected space. In the following sections, we discuss how to train the vision-language embedding in a way that makes it an effective space for discovering failures, and how to train the π^{MD} on this continuous space to predict failures beyond the observed environment.

Training the Vision-Language Embedding. The objective of training this embedding is to provide a continuous space that encodes some information about success-failures for the RL agent to start with. We collect a small number of rollouts, M, of a given manipulation policy for a given task with $\mathcal{D} = \{(x_i^{\text{vision}}, x_i^{\text{lang}}), y_i\}_{i=1}^{M}$, where x^{vision} is the raw image input that we typically provide to manipulation policies, x^{lang} is a short textual description of the task, and $y \in \{\text{failure}, \text{success}\}$. Since we know the action (environment variation) we apply, the textual description can be automatically constructed (Refer Appendix III). With this data, as shown in Fig. 3, we train a new dual backbone architecture that consists of:

1) A Vision Transformer (ViT) backbone [35] to convert x_i^{vision} to visual features.
2) A CLIP encoder [36] to process semantic information from x_i^{lang}.

The dual architecture combines complementary strengths: ViT captures detailed spatial information, while CLIP aligns visual data with semantic meaning, resulting in a robust multimodal embedding that enables better generalization across diverse scenarios and environments for failures.

To train this vision-language embedding, a contrastive learning objective is employed, where the model learns to group embeddings of semantically similar actions (e.g., actions that change table colors) closer together while pushing apart embeddings of semantically dissimilar actions (e.g., actions that change lighting and actions that change table size). We train the embedding by minimizing the contrastive loss,

Algorithm 1 Failure Diagnosis

Init: steps N, embeddings $\mathcal{E}$ (for unseen), $a_{\text{old}} \leftarrow \emptyset$, manipulation policy π^{R}
for $i = 1$ to N **do**
  Sample $a \sim \pi^{\text{MD}}(a \mid s)$
  $a_{\text{new}} \leftarrow \begin{cases} a_{\text{old}} + a, & \text{discrete} \\ \arg\min_{e \in \mathcal{E}} \|a - e\|, & \text{continuous} \end{cases}$
  Execute π^{R} with a_{new}
  if failure detected **then**
    Reset; $a_{\text{old}} \leftarrow \emptyset$
  else
    $a_{\text{old}} \leftarrow a_{\text{new}}$
  end if
end for

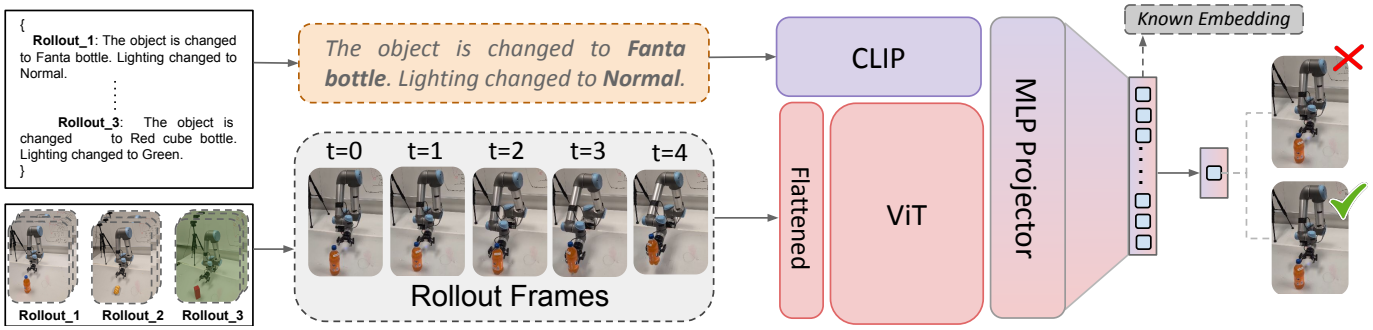

Fig. 3: The pipeline shows how rollouts with disruptions (e.g., object or lighting changes) are processed to learn meaningful embeddings. Text and visual data from the rollouts are embedded using CLIP and ViT, then projected to an MLP to generate text, image to failure aligned representations.

$\sum_{i,j\in\mathcal{D}}\left[\mathbb{1}_{y_i=y_j}\cdot d_{ij}+\mathbb{1}_{y_i\neq y_j}\cdot\max(0,\mathrm{m}-d_{ij})\right]$ where the indicator function $\mathbb{1}_{y_i=y_j}$ measures if the two labels are from the same class, $d_{ij}=\|\mathbf{e}_i-\mathbf{e}_j\|_2$ represents the Euclidean distance between embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$, and m is a hyperparameter that defines the minimum separation between e.

Training RoboMD deep RL policy with continuous actions. In continuous action spaces, the agent navigates the embedding space guided by *known embeddings*, $\mathcal{E}=\{\mathbf{e}_i; i\in\mathcal{D}\}$, the small set of pre-computed embeddings derived from $\mathcal{D}$. These embeddings serve as reference points in the action space, representing well-understood regions where failure/success is already observed. As shown in Algorithm 1, the RoboMD RL policy samples an action a' from the embeddings space and finds the closest embedding in $\mathcal{E}$ to obtain its corresponding a. Note that this action is now an embedding. Thus, performing an action implicitly applies a variation to the environment, although we are not explicitly changing the environment. Therefore, these actions are extremely cheap compared to explicitly changing the environment and running rollouts as in the discrete case discussed in Section III-A. We define the reward function to encourage discovering failure regions while discouraging large deviations from $\mathcal{E}$. This is because significant deviations from $\mathcal{E}$ lead to uncertain or poorly understood regions. This rewards mechanism can be captured

$$R(s,a)=\begin{cases}\frac{C_{\text{failure}}}{\text{penalty}+1}-k\cdot\mathcal{N}(a), & \text{if failure,}\\ -\frac{C_{\text{success}}}{\text{horizon}\times(\text{penalty}+1)}, & \text{if success.}\end{cases}\tag{1}$$

Here, T is the episode length (number of timesteps per rollout). The distance penalty scales with $\|a-\mathcal{E}\|$. The frequency penalty $\mathcal{N}(a)$ counts consecutive repeats of a. In our experiments, we set the penalty coefficient k=5. This formulation encourages adaptive exploration while leveraging prior knowledge of success and failure-prone regions. Figure 4 illustrates how a few success/failure examples are embedded into a semantic space and then used to train a generalized failure detector, π^{MD} . During training, RL samples this continuous embedding rewarded for steering toward failure-prone regions without requiring full policy roll-outs.

C. Uncovering Failures

Because RoboMD's policy concentrates probability on failure-inducing actions, we can read the likely failure modes directly from its distribution.

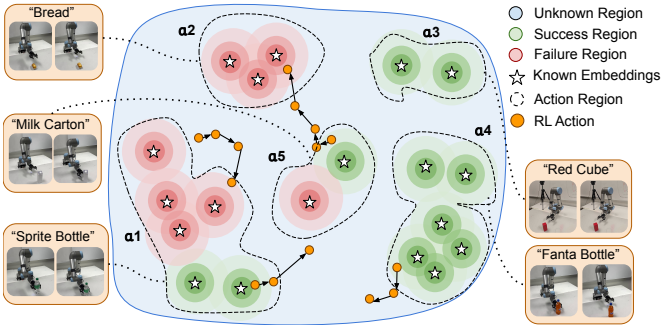

Fig. 4: Continuous action space exploration: regions of Unknown (blue), Success (green), and Failure (red). Stars denote known embeddings guiding exploration; orange circles and arrows show the agent's transition sequence. Dashed boundaries group similar outcomes, and by biasing transitions toward failures, the policy π^{MD} encodes the failure distribution.

a) Discrete actions (Sec. III-A): The policy is given by $\pi^{\text{MD}}(a\mid s)=\frac{\exp(f_a(s))}{\sum_{a'}\exp(f_{a'}(s))}$. assigns a PMF over the discrete action set $\mathcal{A}$, where $f_a(s)$ is the logit for action a. As training proceeds, π^{MD} allocates higher mass to failure-causing actions, enabling systematic exploration of discrete FM's.

b) Continuous actions (Sec. III-B): Here $\pi^{\text{MD}}(a\mid s)$ is a Gaussian PDF $p(a\mid s)$ on $\mathbb{R}^d$. Although $p(a_0)=0$ for any exact a_0, likelihood ratios remain well-defined. To compare two perturbations a_1,a_2, we compute $\frac{p^{\text{MD}}(a_1|s)}{p^{\text{MD}}(a_2|s)}$, analogous to PPO's probability-ratio objective.

D. Using Failures for Policy Improvement

For discrete actions (seen environment variations or FMs), we obtain the probability of each FM, $\{(a_1,p_1),(a_2,p_2),\ldots,(a_n,p_n)\}$. For continuous actions (seen or unseen environment variations), we obtain the order of FMs based on their likelihood (e.g., $a_1>a_3>a_4>a_2$). Rather than collecting rollouts on the entire FM candidate set, $\mathcal{C}$, and other random environment variations (Fig 16), these probabilities or likelihoods allow the user to identify the top FMs and generate targeted rollouts on them to fine-tune the manipulation policy.

IV. Experimental Results

In this section, we investigate the following questions:

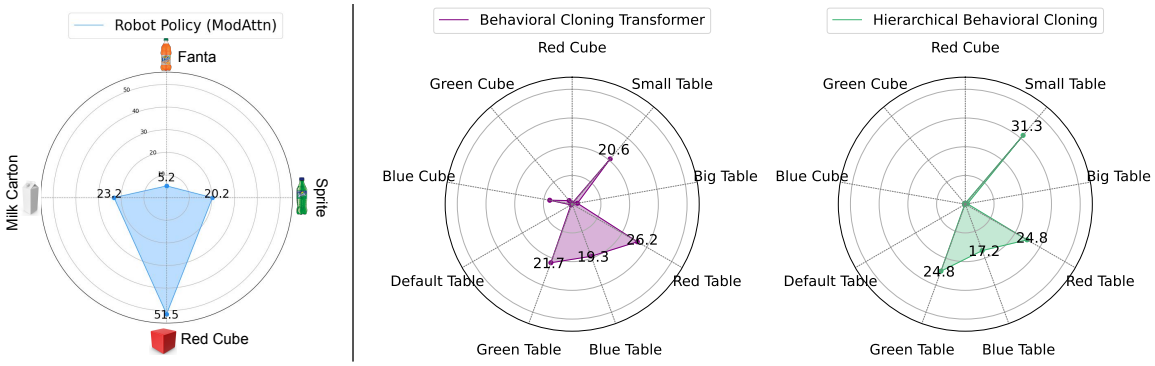

Fig. 5: Individual FM analysis of multiple models. Each radar plot shows failure likelihood for different actions across environments (real in left, simulation middle/right). Numbers along each axis indicate probability of failure.

Reinforcement Learning Models				
Model	Lift	Square	Pick Place	Avg. Score
A2C	74.2%	79.0%	72.0%	75.0
PPO	**82.3%**	**84.0%**	**76.0%**	**80.7**
SAC	51.2%	54.6%	50.8%	52.2
Vision–Language Models				
Qwen2-VL	32.0%	24.6%	**57.4%**	38.0
Gemini 1.5 Pro	**59.0%**	36.4%	37.4%	44.3
GPT-4o	57.0%	44.0%	32.0%	33.3
GPT-4o-ICL (5 Shot)	57.4%	**48.6%**	57.0%	**54.3**

TABLE I: Benchmark results comparing RL controllers (PPO, A2C, SAC), vision–language models, and lightweight CNN/ResNet failure detectors each paired with a BC-MLP low-level policy. RoboMD (PPO Continuous) consistently outperforms other baselines.

1) How does RoboMD's failure-mode detection in seen scenarios compare to other RL methods and VLMs?
2) How well does RoboMD generalize to both seen and unseen environment variations?
3) How do the FMs diagnosed by RoboMD help improve pre-trained BC models?

A. Benchmark Comparisons

Experimental setup: We evaluate RoboMD using models trained in RoboSuite [37] using datasets from RoboMimic [38] and MimicGen [39]. Benchmarking is performed across Lift, Stack, Threading, and Can tasks, which represent common set of manipulation challenges with varying levels of difficulty.

To evaluate the accuracy of RoboMD, we construct a dataset of success-failure pairs, where each pair consists of a randomly selected success case and a failure case. Since a successful action will rank higher than a failure, this provides ground truth to evaluate RoboMD's ranking consistency. The results, presented in Table I, highlights that RoboMD consistently outperforms all baselines in accuracy across tasks. As VLMs are the most popular way to characterize failures [5, 4, 6, 7, 8], we conducted evaluations with state-of-the-art proprietary models (GPT-4o and Gemini 1.5 Pro) and an open-source model (Qwen2-VL). Additionally, we extended the evaluation of GPT-4o by employing in-context learning (ICL) with 5-shot demonstrations to gauge its adaptability, ICL improves the performance of GPT-4o, particularly in the *Square* task. However, overall VLM performance remains below 60%, indicating they struggle with reliably predicting configurations.

TABLE II: Failure characteristics for discrete action spaces.

Model	**Entropy** (↓)	**NFM** (↓)	**FSI** (↓)
IQL [42]	2.49	4	0.72
BCQ [43]	2.79	6	1.15
BC Transformer	2.47	5	0.98
HBC [44]	2.11	4	0.68
BC (Two-Image Input)	2.14	3	0.63
BC (Proprioceptive + Image)	2.58	3	0.75

TABLE III: RoboMD failure detection accuracy across tasks.

Algorithm	**Lift**	**Pick Place**	**Threading**	**Stack**
BC	82.5%	76.0%	68.0%	88.0%
BCQ	61.5%	72.5%	62.0%	72.0%
HBC	83.5%	79.0%	73.0%	81.0%
BC Transformer	74.5%	72.0%	82.0%	70.5%
Diffusion	85.0%	71.0%	71.0%	62.0%

To compare exploration across RL algorithms, we analyze the action distributions of A2C [40], SAC [41] and PPO over 21 environment variations. As shown in Table II, PPO achieves the highest entropy (2.8839), indicating broader exploration for failure discovery.

Having established the superior performance of PPO, we further use RoboMD to evaluate the performance of a variety of policies learned using different training methods. The results in Table III demonstrate that RoboMD generalizes well across different tasks and policy architectures, effectively detecting failures in diverse learning frameworks. Overall, these findings verifies that RoboMD can be applied across different manipulation tasks.

B. Generalization to Seen and Unseen Environments

To assess the generalization capabilities of RoboMD, we evaluate its performance in both seen and unseen environment variations. See Appendix I for experimental setup.

Seen Environments: Figure 5 visualizes the failure distributions across different actions, which provides a detailed failure mode (FM) analysis of different models, demonstrating that lower entropy corresponds to more structured yet concentrated failures, making them easier to identify and address.

TABLE IV: In real (a_r: {*Bread* [unseen], *Red Cube*, *Milk Carton*, *Sprite*}) vs. simulated (a_s: {*Red Table*, *Black Table* [unseen], *Green Lighting*}) environments, rank consistency measures ordering and accuracy is computed over 21 unseen variations.

Task ID	Algorithm	Continuous Rank	Ground Truth Rank	Consistency	Accuracy
Real Robot (UR5e)	ModAttn [45]	$a_r1 > a_r2 > a_r3 > a_r4$	$a_r1 > a_r2 > a_r3 > a_r4$	✓	-
Sim. Can	HBC [44]	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	61%
Sim. Square	Diffusion [46]	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	68%
Sim. Stack	BCQ [43]	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	80%
Sim. Threading	BC Transformer	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	74%

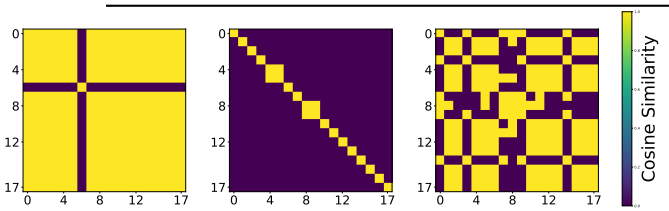

Fig. 6: Confusion matrices of embeddings trained using a) Binary Cross-Entropy (BCE) loss, b) BCE and Contrastive Loss, and c) both losses but no text encoder. Diagonal is better.

TABLE V: Deviation scores measuring embedding quality across different loss functions. Lower MSE and Frobenius-norm distances (to the identity) indicates better separation

Eval Loss	Image BCE	Image BCE+Contr.	Image+Text BCE	Image+Text BCE+Contr.
MSE (↓)	0.6495	0.6179	0.8426	0.1801
Fro dist. (↓)	14.5060	14.1497	16.5227	7.6387

To quantify the failure characteristics of each model, Table II summarizes the entropy values and the number of FM identified for each model. The Failure Severity Index (FSI) quantifies the weighted impact of failures defined by $\sum_{i=1}^{N} P_{\text{failure}}(a_i) \cdot W_i$ where P_{failure}) represents the probability of failure for action a_i, and W_i is the normalized weight such that the FM with the highest probability is assigned a weight of 1, while others are scaled proportionally. Models such as *HBC* demonstrate lower entropy and fewer FM, highlighting their robustness under discrete action perturbations.

Unseen Environments: To assess generalization to unseen environment variation (i.e., actions), We first construct a dataset of 100 unseen success-failure pairs, similar to Sec IV-A, to evaluate generalization. We further test RoboMD on an unseen action not used during RL training to check if failure rankings remain consistent. Table IV shows that most actions maintain their rankings, verifying the reliability of failure identification.

C. Ablation: Quality of Vision-Language Embeddings

To evaluate the quality of our trained vision-language embeddings, we compute cosine similarity-based confusion matrices for different action embeddings. An ideal embedding should produce a confusion matrix with a strong diagonal structure, where each embedding is highly similar to itself while being distinct from others. We compare three configurations as shown in Fig. 6. As shown in Table V, the Image + Text backbone trained with BCE and Contrastive loss achieves the lowest MSE (0.1801) and Frobenius norm distance (7.6387). This good embedding quality leads to better action separability compared to other configurations.

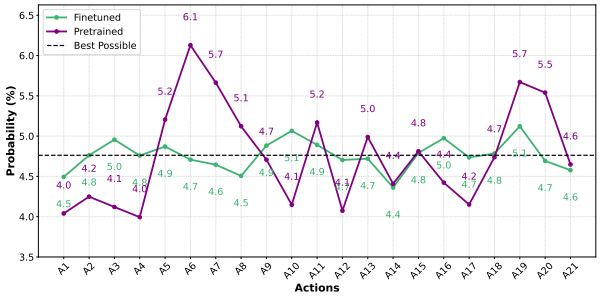

Fig. 7: Failure distribution before and after fine-tuning "Lift" BC policy on FMs chosen by RoboMD. The ideal distribution (dashed black) represents zero failure across all actions.

D. Failure-Guided Fine-Tuning

We analyze how incorporating failure examples during training and fine-tuning enhances the model's ability to generalize across different conditions. We compare behavior cloning policies on the Lift task before and after fine-tuning using failure-inducing samples. Fine-tuning is conducted under multiple conditions which is shown in AppendixIV.

Fig. 7 compares failure distributions of pretrained vs. fine-tuned manipulation policy with failure-inducing samples. The pretrained policy exhibits higher failure probabilities across multiple actions, deviating significantly from the ideal distribution. Fine-tuning reduces failures (details in Appendix IV). This improvement is quantitatively supported by the Wasserstein distance: the fine-tuned policy is closer to the ideal distribution (0.0014) compared to the pretrained policy (0.0051). This also shows that targeted fine-tuning on diverse failure cases enhances policy robustness by reducing failure probabilities across a broader range of environment conditions.

V. Conclusion

We introduced *RoboMD*, a framework for diagnosing failure modes in robot manipulation policies by leveraging deep RL in both discrete and continuous action spaces. Our experiments show that *RoboMD*, especially when built on PPO in continuous spaces, outperforms baseline RL models and VLMs, captures subtle failure scenarios, ranks failure-inducing actions, and generalizes to unseen environment variations. Moreover, *RoboMD* can be integrated into continuous testing pipelines to automatically flag regressions as policies evolve, enabling rapid feedback loops during development. These capabilities pave the way for deploying more reliable manipulation systems in real-world settings. Future work will aim to train a generalist PPO model to handle combined tasks and broader environment distributions, further strengthening policy robustness in unstructured settings.

References

[1] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024. URL https://arxiv.org/pdf/2402.08191.

[2] Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3153–3160. IEEE, 2024. URL https://arxiv.org/abs/2307.03659.

[3] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024. URL https://arxiv.org/abs/2410.18647.

[4] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation. *arXiv preprint arXiv:2410.00371*, 2024. URL https://arxiv.org/pdf/2410.00371.

[5] Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *arXiv preprint arXiv:2410.04640*, 2024. URL https://arxiv.org/pdf/2410.04640.

[6] Lukas Klein, Kenza Amara, Carsten T Lüth, Hendrik Strobelt, Mennatallah El-Assady, and Paul F Jaeger. Interactive Semantic Interventions for VLMs: A Human-in-the-Loop Investigation of VLM Failure. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL https://openreview.net/pdf?id=3kMucCYhYN.

[7] Rakshith Subramanyam, Kowshik Thopalli, Vivek Narayanaswamy, and Jayaraman J Thiagarajan. Decider: Leveraging foundation model priors for improved model failure detection and explanation. In *European Conference on Computer Vision*, pages 465–482. Springer, 2025. URL https://arxiv.org/pdf/2408.00331.

[8] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023. URL https://arxiv.org/abs/2306.15724.

[9] Som Sagar, Aditya Taparia, and Ransalu Senanayake. Failures are fated, but can be faded: Characterizing and mitigating unwanted behaviors in large-scale vision and language models. *arXiv preprint arXiv:2406.07145*, 2024. URL https://arxiv.org/pdf/2406.07145.

[10] Harrison Delecki, Masha Itkina, Bernard Lange, Ransalu Senanayake, and Mykel J Kochenderfer. How do we fail? stress testing perception in autonomous vehicles. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5139–5146. IEEE, 2022. URL https://arxiv.org/pdf/2203.14155.

[11] Zhang-Wei Hong, Idan Shenfeld, Tsun-Hsuan Wang, Yung-Sung Chuang, Aldo Pareja, James Glass, Akash Srivastava, and Pulkit Agrawal. Curiosity-driven red-teaming for large language models. *arXiv preprint arXiv:2402.19464*, 2024. URL https://arxiv.org/pdf/2402.19464.

[12] Anthony Corso, Robert Moss, Mark Koren, Ritchie Lee, and Mykel Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research*, 72:377–428, 2021. URL https://arxiv.org/pdf/2005.02979.

[13] Julia Nitsch, Masha Itkina, Ransalu Senanayake, Juan Nieto, Max Schmidt, Roland Siegwart, Mykel J Kochenderfer, and Cesar Cadena. Out-of-distribution detection for automotive perception. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2938–2943. IEEE, 2021. URL https://arxiv.org/pdf/2011.01413.

[14] Jayaraman J Thiagarajan, Vivek Narayanaswamy, Puja Trivedi, and Rushil Anirudh. PAGER: A Framework for Failure Analysis of Deep Regression Models. *arXiv preprint arXiv:2309.10977*, 2023. URL https://arxiv.org/pdf/2309.10977.

[15] Ransalu Senanayake. The role of predictive uncertainty and diversity in embodied ai and robot learning. *arXiv preprint arXiv:2405.03164*, 2024. URL https://arxiv.org/pdf/2405.03164.

[16] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012. URL https://arxiv.org/pdf/1811.10156.

[17] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017. URL https://arxiv.org/pdf/1703.04977.

[18] Yiding Jiang, J Zico Kolter, and Roberta Raileanu. On the importance of exploration for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://arxiv.org/pdf/2306.05483.

[19] Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. *Advances in neural information processing systems*, 31, 2018. URL https://papers.nips.cc/paper_files/paper/2018/file/943aa0fcda4ee2901a7de9321663b114-Paper.pdf.

[20] Daniel Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*, pages 1165–1177. PMLR, 2020. URL https://papers.nips.cc/paper_files/paper/2018/file/943aa0fcda4ee2901a7de9321663b114-Paper.pdf.

[21] Deepak Ramachandran and Eyal Amir. Bayesian Inverse Reinforcement Learning. In *IJCAI*, volume 7,

pages 2586–2591, 2007. URL https://www.ijcai.org/Proceedings/07/Papers/416.pdf.

[22] Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

[23] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/pdf/2212.06817.

[24] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/pdf/2307.15818.

[25] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39 (1):3–20, 2020. URL https://arxiv.org/pdf/1808.00177.

[26] Andrea Bajcsy and Jaime F Fisac. Human-AI Safety: A Descendant of Generative AI and Control Systems Safety. *arXiv preprint arXiv:2405.09794*, 2024. URL https://arxiv.org/pdf/2405.09794.

[27] Philipp Grimmeisen, Friedrich Sautter, and Andrey Morozov. Concept: Dynamic Risk Assessment for AI-Controlled Robotic Systems. *arXiv preprint arXiv:2401.14147*, 2024. URL https://arxiv.org/pdf/2401.14147.

[28] Lindsay Sanneman and Julie A Shah. The situation awareness framework for explainable AI (SAFE-AI) and human factors considerations for XAI systems. *International Journal of Human–Computer Interaction*, 38 (18-20):1772–1788, 2022. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC7338174/.

[29] Alec Farid, David Snyder, Allen Z Ren, and Anirudha Majumdar. Failure prediction with statistical guarantees for vision-based robot control. *arXiv preprint arXiv:2202.05894*, 2022. URL https://arxiv.org/pdf/2202.05894.

[30] Allen Z Ren and Anirudha Majumdar. Distributionally robust policy learning via adversarial environment generation. *IEEE Robotics and Automation Letters*, 7(2): 1379–1386, 2022. URL https://arxiv.org/pdf/2107.06353.

[31] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020. URL https://arxiv.org/abs/2001.07715.

[32] Joseph A Vincent, Haruki Nishimura, Masha Itkina, and Mac Schwager. Full-Distribution Generalization Bounds for Imitation Learning Policies. In *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023. URL https://openreview.net/pdf?id=JZkwYiyy9I.

[33] Jana Tmov, Luis I Reyes Castro, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Minimum-violation LTL planning with conflicting specifications. In *2013 American Control Conference*, pages 200–205. IEEE, 2013. URL https://arxiv.org/pdf/1303.3679.

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/pdf/1707.06347.

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/pdf/2010.11929.

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. URL https://arxiv.org/pdf/2103.00020.

[37] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. URL https://arxiv.org/abs/2009.12293.

[38] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021. URL https://arxiv.org/abs/2108.03298.

[39] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2310.17596.

[40] Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016. URL https://arxiv.org/abs/1602.01783.

[41] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. URL https://arxiv.org/abs/1801.01290.

[42] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv*

preprint arXiv:2110.06169, 2021. URL https://arxiv.org/abs/2110.06169.

[43] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019. URL https://arxiv.org/abs/1812.02900.

[44] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020. URL https://arxiv.org/abs/2003.06085.

[45] Yifan Zhou, Shubham Sonawani, Mariano Phielipp, Simon Stepputtis, and Heni Ben Amor. Modularity through attention: Efficient training and transfer of language-conditioned policies for robot manipulation. *arXiv preprint arXiv:2212.04573*, 2022. URL https://arxiv.org/abs/2212.04573.

[46] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023. URL https://arxiv.org/abs/2303.04137.

APPENDIX

I. EXPERIMENTAL SETUP

A. Real-World Experiment Setup

Real-world experiments were conducted using a UR5e robotic arm equipped with high-resolution cameras and a standardized workspace. The setup is shown below in Fig 8.

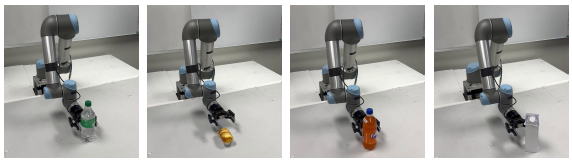

Fig. 8: Scenes from experiments on real world robot

B. Simulation Experiment Setup

Simulation experiments were performed using the MuJoCo physics engine integrated with Robosuite. The simulated environments included variations in object positions, shapes, and textures. The simulation allowed extensive testing across diverse scenarios. Below we show a few samples in Fig 8.

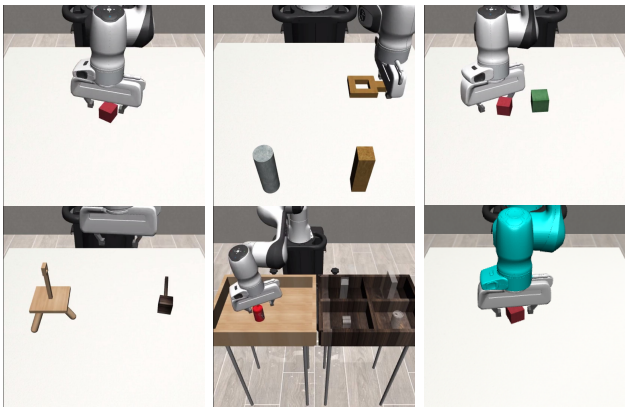

Fig. 9: Scenes from experiments on Robosuite

C. Baselines

To validate the effectiveness of our method, we compared it against two categories of baselines: Reinforcement Learning (RL) baselines and Vision-Language Model (VLM) baselines. Below, we detail their implementation, hyperparameters, and specific configurations.

1) Reinforcement Learning (RL) Baselines: The RL baselines were implemented using well-established algorithms, each optimized for the task to ensure a fair comparison. The following RL methods were included:

- **Proximal Policy Optimization (PPO):** A policy-gradient method known for its stability and efficiency. Key hyperparameters included:
  - Learning rate: 3×10^{-4}
  - Discount factor (γ): 0.99
  - Clipping parameter (ϵ): 0.2
  - Number of epochs: 10
  - Batch size: 64
  - Actor-Critic network layers: [128, 256, 128]
- **Soft Actor-Critic (SAC):** A model-free off-policy algorithm optimized for continuous action spaces. The key hyperparameters were:
  - Learning rate: 1×10^{-3}
  - Discount factor (γ): 0.99
  - Replay buffer size: 1×10^{6}
  - Target entropy: $-\dim(\text{action space})$
  - Batch size: 128
- **Advantage Actor Critic (A2C):**
  - Learning rate: 2.5×10^{-4}
  - Discount factor (γ): 0.99
  - Exploration strategy: Epsilon-greedy (ϵ decayed from 1.0 to 0.1 over 500,000 steps)
  - Replay buffer size: 1×10^{6}
  - Batch size: 64
  - Neural network layers: [128, 256, 128]

Each RL baseline was evaluated using the same metrics, ensuring consistency across comparisons.

2) Vision-Language Model (VLM) Baselines: The VLM baselines take advantage of the interplay between visual and textual modalities for task representation. We evaluated 3 state-of-the-art VLMs adapted to our task:

1) GPT-4o
2) Gemini 1.5 Pro
3) Qwen2-VL

Additionally, we leverage GPT-4o with in-context learning, using five demonstrations. First, we process the output trajectories into videos and compute the appropriate frame rate to generate video sequences equivalent to 15 frames per trajectory pair. These sequences, representing perturbation scenarios, are provided to the VLMs along with a system prompt that includes a detailed policy description, training configuration, and a natural language task description. For evaluation, we structure the testing dataset using a pairwise comparison framework, where each model is prompted to assess two input video sequences and rank which is more likely to result in task success. The results are recorded in a CSV file, and we compute comparison scores by analyzing model rankings against ground-truth rollouts in the simulated perturbation.

II. RATIONALE FOR USING REINFORCEMENT LEARNING

RL is employed in the RoboMD framework due to its ability to explore high-dimensional, complex action spaces and optimize sequential decision-making under uncertainty. This section outlines the key motivations for choosing RL as the core methodology:

Exploration of High-Risk Scenarios: Traditional approaches to analyzing robot policy failures often rely on deterministic sampling or exhaustive evaluation, which become infeasible in large, dynamic environments. RL allows targeted exploration by learning an agent that actively seeks out environmental configurations likely to induce policy failures. This

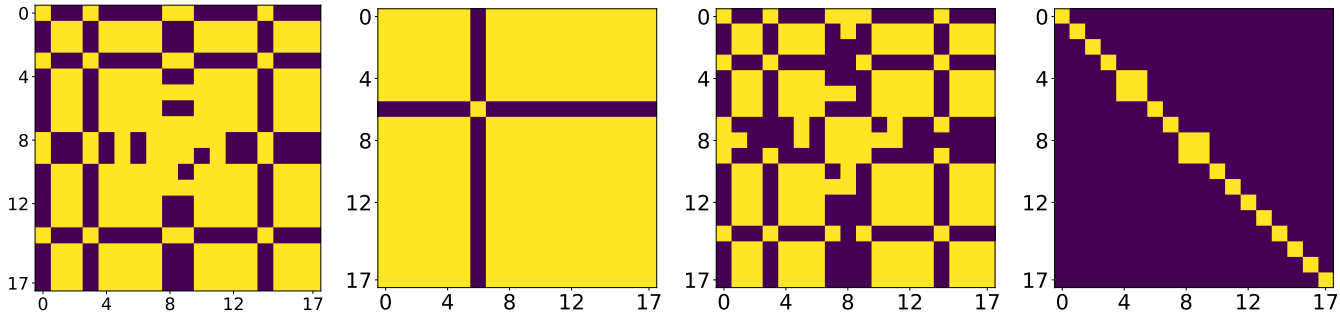

Fig. 10: The order in which the confusion matrix is a) Image Ecoder + BCE b) Image + Text Encoder + BCE loss c) Image Encoder + BCE + Contrastive loss d) Image + Text Encoder + BCE + Contrastive loss

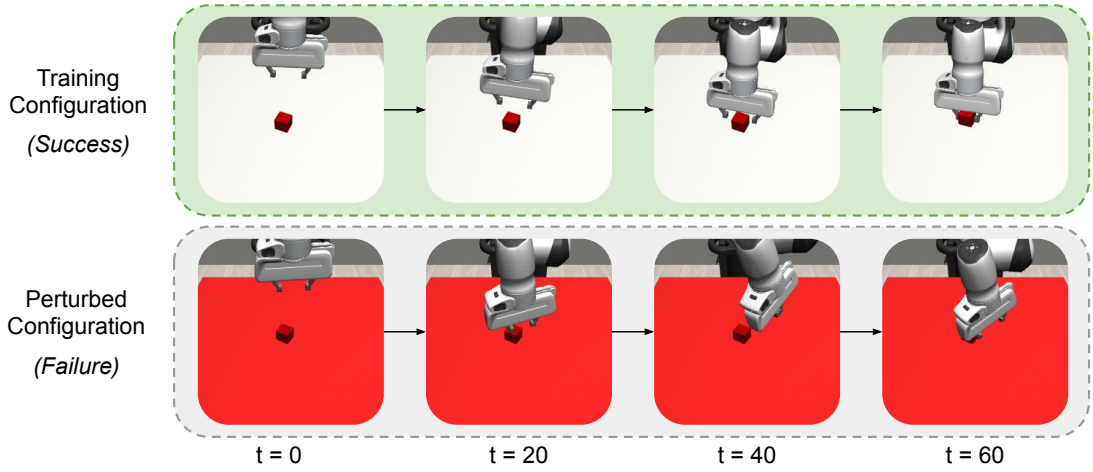

Fig. 11: Testing Robustness Under Visual Perturbations: Successful Rollout in Training vs. Failure Induced by Red Table Distraction

capability is particularly useful for systematically uncovering vulnerabilities in high-dimensional environments.

Optimization of Failure Discovery: The objective of RoboMD is to maximize the occurrence of failures in pre-trained policies. RL frameworks, such as PPO, are well-suited for this task as they iteratively refine policies to achieve specific goals, such as identifying high-risk states. The reward function incentivizes the agent to find configurations where the manipulation policy fails by going through multiple actions to induce failures. Fig 11 shows several steps of the manipuation policy rollout.

Comparison with Alternative Methods: While other methods, such as supervised learning or heuristic-based exploration, can provide valuable insights into specific failure cases, they are limited in their scope and adaptability. Supervised learning approaches rely heavily on labeled data, which is challenging to obtain for failure analysis, particularly for rare or unseen failure modes. These methods also lack the ability to adapt dynamically to changes in the environment, reducing their effectiveness in exploring novel or complex failure scenarios. Similarly, heuristic-based exploration methods, such as grid search or predefined sampling strategies, can identify failure cases under controlled conditions but struggle to generalize in high-dimensional environments where the space of possible failure configurations is vast. These methods are also constrained by their reliance on static, predefined rules, which often fail to capture the intricate interactions between environmental factors and failure likelihoods. In contrast, reinforcement learning excels in scenarios where exploration and generalization are critical. Through reward-driven learning, RL agents actively seek configurations that maximize the probability of failure, uncovering patterns and interactions that static methods are likely to miss. Moreover, RL does not require a fully labeled dataset; it iteratively refines its policy through interaction with the environment, making it highly adaptive and scalable. By focusing on cumulative rewards, RL is uniquely positioned to generalize across a wide range of failure-inducing conditions, including edge cases and scenarios resulting from complex factor interactions. This adaptability and exploratory capability make RL an ideal framework for large-scale failure analysis in dynamic and uncertain environments, surpassing the limitations of traditional supervised learning or heuristic-based approaches.

III. Continuous Action Space Embedding

Embedding actions in a continuous space is crucial for efficiently capturing the underlying structure of decision-

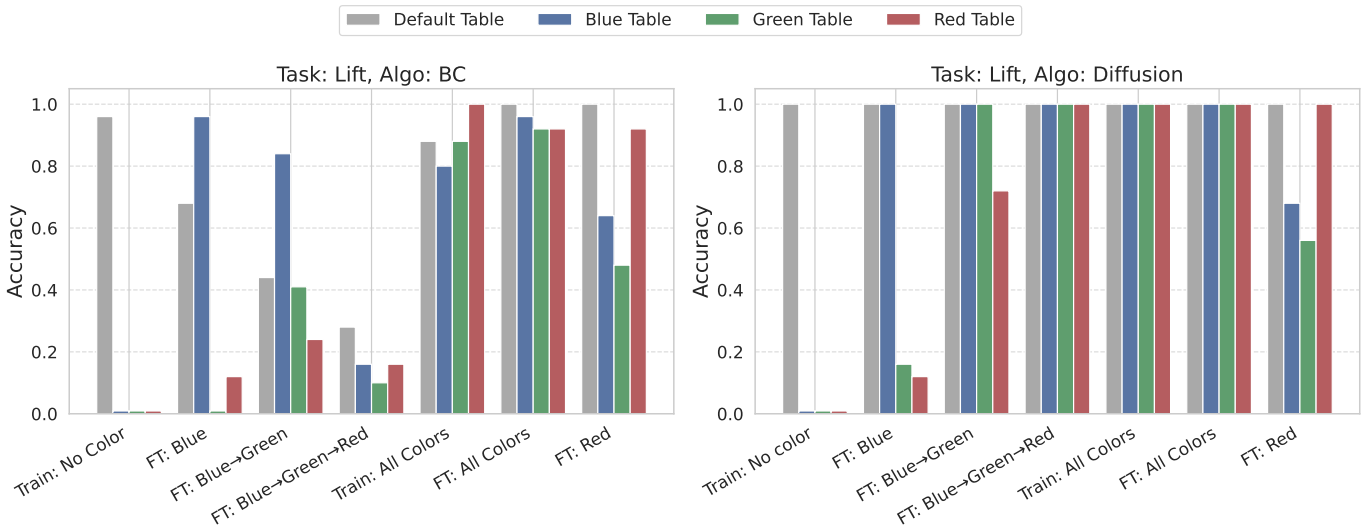

Fig. 12: Performance comparison of behavior cloning (BC) and diffusion-based policies on the Lift task before and after fine-tuning with failure-inducing samples. Each bar represents the success rate of the policy across different **table colors**.

making processes. Unlike discrete action spaces, where each action is treated as an independent category, continuous action space embeddings aim to encode similarities and relationships between actions in a structured space.

TABLE VI: Actions for Can and Box tasks.

Task	Action Description
Can	Change the can color to red.
	Change the can color to green.
	Change the can color to blue.
	Change the can color to grey.
Box	Change the box color to green.
	Change the box color to blue.
	Change the box color to red.
Box Sizes	Resize the box to $0.3 \times 0.3 \times 0.02$ (L, B, H).
	Resize the box to $0.2 \times 0.2 \times 0.02$ (L, B, H).
	Resize the box to $0.1 \times 0.1 \times 0.02$ (L, B, H).

A. Action Description Mapping for CLIP Language Input

To generate language inputs for CLIP, we use a mapped dictionary that encodes the action being applied to the image. The action descriptions for different tasks are detailed in Table VI. This table represents only a subset of possible actions, and users are free to modify the language as needed. The descriptions are not strict requirements, as the model learns over time to associate text and images with failure patterns, allowing for flexibility in phrasing while maintaining the underlying semantic meaning. The actions used for Lift task is as follows which was also shown as (A1,A2...A21) in Fig 7:

1) Change cube color to red
2) Change cube color to green
3) Change cube color to blue
4) Change cube color to gray
5) Change table color to green
6) Change table color to blue
7) Change table color to red
8) Change table color to gray
9) Resize table to (0.8, 0.2, 0.025)
10) Resize table to (0.2, 0.8, 0.025)
11) Resize cube to (0.04, 0.04, 0.04)
12) Resize cube to (0.01, 0.01, 0.01)
13) Resize cube to (0.04, 0.01, 0.01)
14) Change robot color to red
15) Change robot color to green
16) Change robot color to cyan
17) Change robot color to gray
18) Change lighting color to red
19) Change lighting color to green
20) Change lighting color to blue
21) Change lighting color to gray

B. Evaluation

Fig 10 illustrates the similarity structure of embeddings trained using only Binary Cross-Entropy (BCE) loss, resulting in highly correlated representations. In contrast, the right matrix, trained with a combination of BCE and Contrastive Loss, demonstrates improved separation, as evidenced by the stronger diagonal structure and reduced off-diagonal similarities.

To assess the quality of the learned embeddings, we conduct an evaluation using a k-Nearest Neighbors (kNN) classifier. Specifically, we train kNN on a subset of the embeddings and analyze the impact of increasing k on test accuracy. The intuition behind this evaluation is that well-separated embeddings should be locally consistent, meaning that a small k (considering only close neighbors) should yield high accuracy,

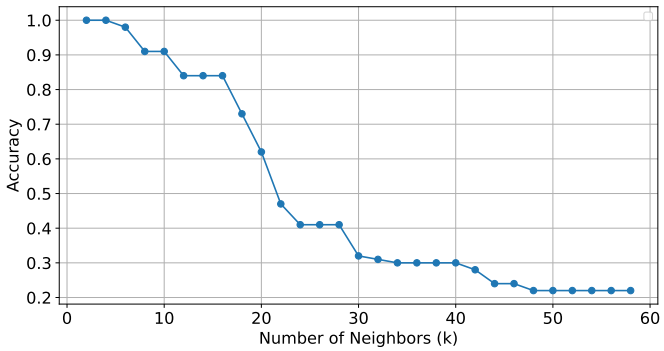

Fig. 13: kNN Accuracy Drop with Increasing k in Continuous Action Space Embeddings

while increasing k (incorporating more distant neighbors) may introduce noise and reduce accuracy as shown in Fig 13.

C. Integrating Visual and Textual Representations

Incorporating a textual backbone alongside the image backbone yielded significantly lower loss values and faster convergence compared to using an image-only backbone.

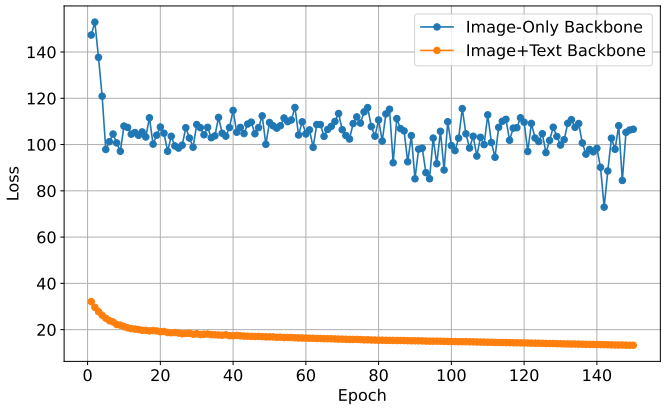

Fig. 14: Training loss for training action representations

This improvement can be attributed to several factors:

1) Semantic Guidance: Textual representations carry rich semantic information that can guide the image backbone. Instead of relying solely on visual cues, the model gains an additional perspective on the underlying concepts (e.g., object names, attributes, or relations).
2) Improved Discriminative Power: With access to text-based information, the model can differentiate between visually similar classes by leveraging linguistic differences in their corresponding textual descriptions.
3) Faster Convergence: Because textual features often come from large, pretrained language models, they are already highly informative. Injecting these features into the training pipeline accelerates the learning process, reducing the number of iterations needed to reach a satisfactory level of performance.

IV. Fine-tuning

Once failure modes are identified. The most effective strategy is fine-tuning the manipulation policy, π^{R}, using all selected failure samples together, rather than iteratively adapting to subsets as shown in Fig 12. To adapt the policy π^{R} against identified failures, we select a subset $\mathcal{C}_{\mathrm{sub}} \subseteq \mathcal{C}$ by choosing samples of area a user wants to improve. Finally, we *fine-tune* π^{R} on the combined dataset $\mathcal{C}_{\mathrm{sub}}$, thereby ensuring targeted corrections for critical failures as shown in Fig 15, where a large FM finetune also lead to accuracy improvement. In scenarios where computational resources allow, fine-tuning on the *entire* set $\mathcal{C}$ may be more effective; however, when resources are constrained, leveraging RoboMD to identify an optimal *subset* of $\mathcal{C}$ is an efficient and robust strategy for policy adaptation.

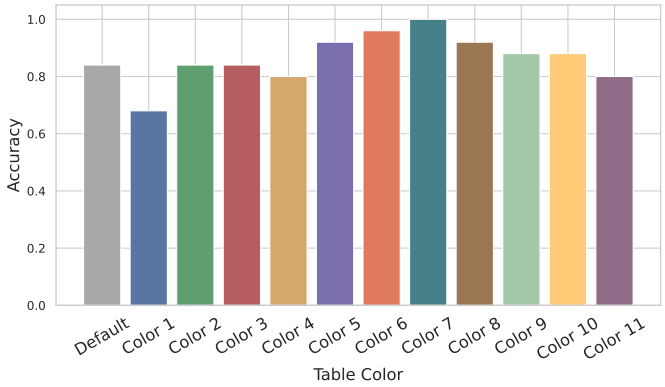

Fig. 15: BC lift finetuned on a combined dataset of 12 different Table colors

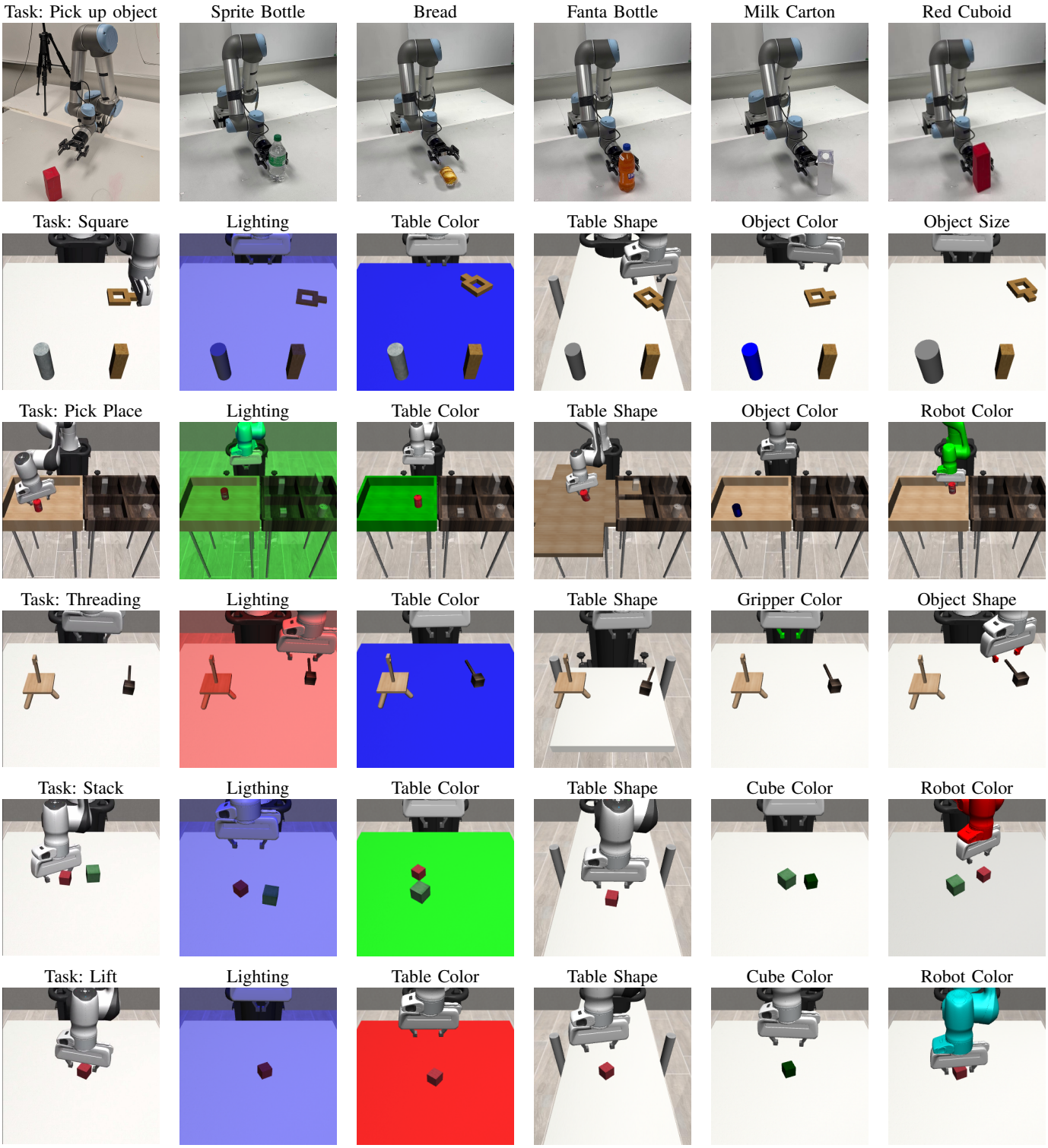

Fig. 16: Environmental and Object Perturbations on Manipulation Tasks