From Mystery to Mastery: Failure Diagnosis for Improving Manipulation Policies

Som Sagar[1], Jiafei Duan[2], Sreevishakh Vasudevan[1], Yifan Zhou[1], Heni Ben Amor[1], Dieter Fox[2,3], and Ransalu Senanayake[1]
[1]Arizona State University, [2]University of Washington, [3]NVIDIA

***Abstract*—Robot manipulation policies often fail for unknown reasons, posing significant challenges for real-world deployment. Researchers and engineers typically address these failures using heuristic approaches, which are not only labor-intensive and costly but also prone to overlooking critical failure modes (FMs). This paper introduces Robot Manipulation Diagnosis (RoboMD), a systematic framework designed to automatically identify FMs arising from unanticipated changes in the environment. Considering the vast space of potential FMs in a pre-trained manipulation policy, we leverage deep reinforcement learning (deep RL) to explore and uncover these FMs using a specially trained vision-language embedding that encodes a notion of failures. This approach enables users to probabilistically quantify and rank failures in previously unseen environmental conditions. Through extensive experiments across various manipulation tasks and algorithms, we demonstrate RoboMD's effectiveness in diagnosing unknown failures in unstructured environments, providing a systematic pathway to improve the robustness of manipulation policies. Project Page: somsagar07.github.io/RoboMD/.**

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. The intricate interactions between policies and their environments produce a vast range of potential failure modes (FMs) [1, 2]. Relying on intuition proves unreliable, and evaluations across all possible environment variations is equally intractable. Hence, efficient methods are essential for systematically uncovering these FMs.

Researchers in robotics employ various methods to understand the behavior of manipulation policies. While techniques such as uncertainty quantification tell us where the model is confident about its own performance, that information is hardly actionable for further policy improvement. On the other hand, quantifying the epistemic uncertainty—the unknown unknowns—in large models is almost impossible [4]. Taking a different route, recent attempts have also tried to quantify failures using brute-force methods [1]. Considering the large amount of potential FMs as well as the complex relationship between policies and their operating environment, we propose using deep reinforcement learning (deep RL) to efficiently evaluate policy performance across diverse environment variations. We name this framework Robot Manipulation Diagnosis (RoboMD).

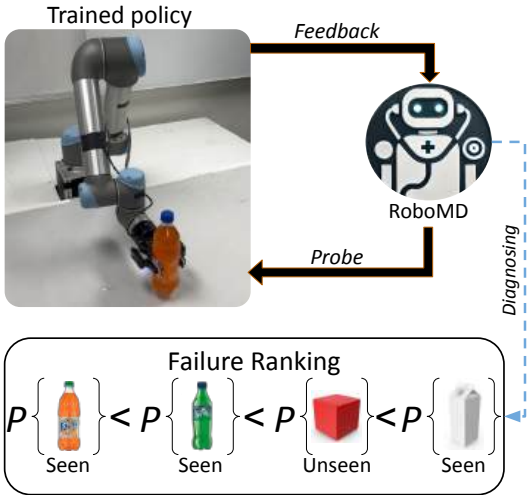

Fig. 1: RoboMD diagnoses failure modes in pre-trained manipulation policies by interacting with the policy and its environment to quantify and rank failure probabilities across both seen and unseen environmental variations (e.g., different object types in this case). This highlights RoboMD's ability to generalize failure diagnosis beyond known environments.

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 that—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 policy performance. This embedding is then used by our deep RL agent to explore failure modes within a continuous action space, rather than being constrained to discrete, *seen* environment conditions. Unlike prior methods that simply query standard VLMs to determine task failure [5], our framework integrates VLM embeddings directly into the RL exploration process. 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 to efficiently diagnose potential failures in pre-trained manipulation policies.
2) Generalizing this method to enable continuous exploration of a specially trained vision-language embedding, which encodes policy performance information, allowing for the discovery and quantification of failures across diverse and previously unseen environmental variations.
3) Demonstrating how failures diagnosed by RoboMD can be used to improve manipulation policies systematically.

Our experimental results demonstrate the effectiveness of RoboMD in identifying and generalizing FMs of four commonly used manipulation tasks on five different manipulation policy training methods, ranging from behavioral cloning to diffusion policies [6]. Furthermore, real-world experiments validate the framework's ability to run in physical settings. As benchmarks and ablations, we compare various diagnosis methods, deep RL methods, and the effect of VLM-backed continuous action spaces.

II. Related work

One way to understand the failures in robot learning models is through the eyes of uncertainty. Considering aleatoric uncertainty—the known unknowns—through probabilistic models is ubiquitous in classical robotic systems [7]. While such techniques make the robots robust to measurement and action noise, they do not inform if the model works or fails. In contrast, the epistemic uncertainty—the unknown unknowns—can be used to understand where we are confident that the model will work [4]. While many attempts have been made to characterize the epistemic uncertainty in robot perception systems [8, 9], only a few attempts have been made to address this challenge in deep reinforcement learning [10] and imitation learning [11, 12, 13]. 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.

Failure detection in large models can be characterized by querying vision-language foundation models [14, 5, 15, 16, 17] or searching for failures [18]. 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. Further, VLM models are not yet capable of making highly accurate quantitative predictions such as probabilities. In the latter approach, outside of robotics, deep reinforcement learning has recently been employed in machine learning to identify errors in image classification and generation [18]. [19, 20] utilized RL to explore adversarial rainy conditions and stress test model robustness. [21] 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 have been extensively studied to address the challenge of robots encountering data outside their training distributions. For example, [22] proposed OOD detection methods for automotive perception without requiring additional training or inference costs. Similarly, [23] introduced sensitivity-aware features to enhance OOD object detection performance. Tools such as PyTorch-OOD have further streamlined the evaluation and implementation of OOD detection methods [24]. [14] introduced a runtime OOD monitoring system for generative policies. Thiagarajan et al. [25] looked at OOD and failures in regression models. Note that failures are more likely when operating outside the training distribution. However, not all out-of-distribution instances result in failures, and failures can also occur within the training distribution. Our goal is to characterize failures both within and beyond the training distribution, rather than simply identifying OOD samples.

Another premise is that 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, 26]. Vision-language-action models trained on multimodal datasets have demonstrated significant advancements in improving adaptability to real-world scenarios [27, 28]. Additionally, approaches such as curriculum learning and domain randomization have proven effective in enhancing generalization by exposing models to progressively complex or randomized environments [29]. These methodologies collectively address the challenges of policy robustness. Our method, being peripheral to these techniques, can be used to identify failures in models trained using any of these methods.

There are numerous other work on characterizing safety from a controls theory perspective [30, 31], human factors perspective [32], etc. Work on providing theoretical certificates through statistical methods [33, 34, 35, 36] or formal methods [37] are also highly valuable. While these diverse approaches are aimed at solving the problem collectively

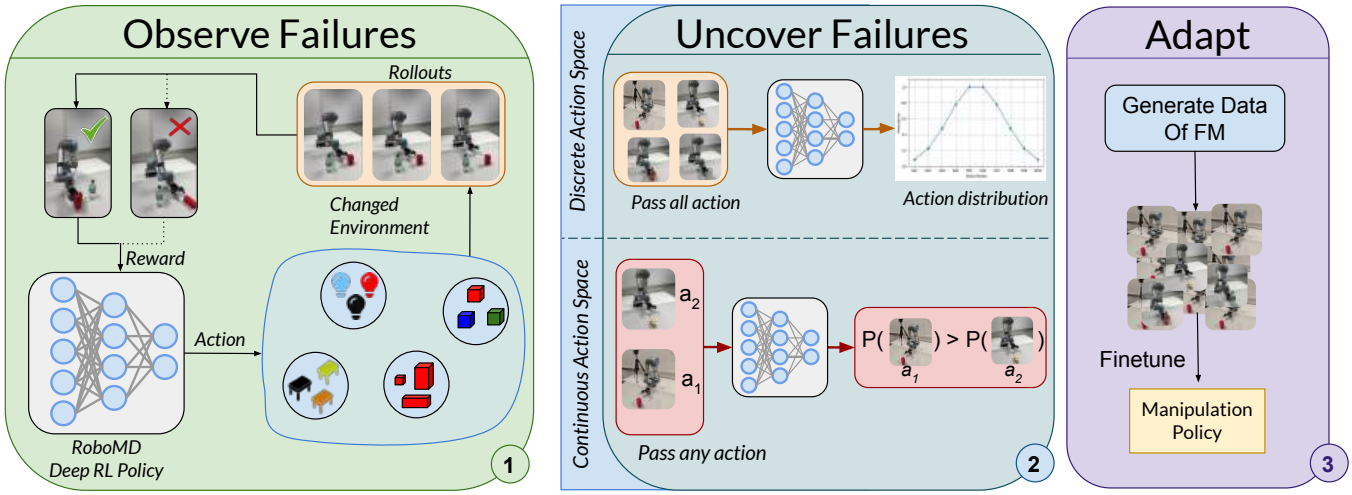

Fig. 2: RoboMD Framework: (1) A PPO-based deep RL agent identifies configurations most likely to induce failures by changing the environment and rolling out the pre-trained manipulation policy. (2) Once PPO training is complete, its output distribution, given an input image of the environment, is analyzed to derive probabilities for each failure mode (FM), quantifying the likelihood of failure. The simpler case is the discrete action space that directly quantifies failure probabilities for candidate FMs. With the continuous action space, we can quantify the likelihood for unseen environment changes. (3) FM likelihoods can be used to fine-tune the policy.

contribute to more robust and safer robot deployment, our framework specifically proposes a method to diagnose failures before deployment, that we empirically show can be used to further improve policies.

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, this candidate set, $\mathcal{C}$, 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 reinforcement learning. 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 for this framework.

The failure diagnosis process is modeled as an MDP, represented by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$, where,

1) State Space ($\mathcal{S}$): The state space consists of the visual input of the robotic environment before the manipulation policy rollout, encapsulating the environment and robot configuration information necessary for detecting failures. This visual input is the same visual input provided as the first time frame of the manipulation policy.
2) Action Space ($\mathcal{A}$): In this subsection, actions are defined as discrete changes to environmental conditions or robot configurations. That is, $\mathcal{A} = \mathcal{C}$. For instance, changing a red table to blue is an action. A sequence of actions, can change various aspects of the environment. In section III-B, we generalize this discrete action space to continuous action space on a specially trained embedding for searching beyond this candidate set.
3) Reward Function (R): The reward is designed to promote failure discovery by assigning positive rewards to failure outcomes and penalizing successful rollouts based on the time required. Formally,

$$R(s,a) = \begin{cases} C_{\text{failure}}, & \text{if failure,} \\ -C_{\text{success}} \times t, & \text{if success,} \end{cases}$$

where C_{success} and C_{failure} are constants and t is the time horizon of the rollout of the manipulation policy, for states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$.
4) Transition Dynamics ($\mathcal{P}$): Transition probabilities, $\mathcal{P}(s'|s,a)$, are governed by the underlying physics engine or the real-world of the robot environment, incorporating stochastic elements such as noise and uncertainties for realistic variability.
5) Discount Factor (γ): A discount factor, $\gamma = 0.99$, is used to balance immediate and future rewards, prioritizing long-term exploration strategies.

Algorithm 1 Failure Diagnosis with Discrete Actions (*Seen*)

Initialize: Number of steps N, previous action $a_{\text{old}} \leftarrow \emptyset$
for $i = 1$ to N **do**
  Sample action $a \in \mathcal{A} = \mathcal{C}$ from RL policy $\pi^{\text{MD}}(a|s)$
  Modify environment with $a_{\text{new}} = a_{\text{old}} + a$
  Execute manipulation policy π^{R}
  if failure detected **then**
    Reset environment
    $a_{\text{old}} \leftarrow \emptyset$ ▷ Discard accumulated actions on failure
  else
    $a_{\text{old}} \leftarrow a_{\text{new}}$ ▷ Accumulate successful actions
  end if
end for

Algorithm 2 Failure Diagnosis with Contin. Actions (*Unseen*)

Initialize: Set of known embeddings $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$, previous action $a_{\text{old}} \leftarrow \emptyset$, number of steps N
for $i = 1$ to N **do**
  Sample action $a \in \mathcal{A}$ from RL policy $\pi^{\text{MD}}(a|s)$
  Find closest embedding $e^* = \arg\min_{e \in \mathcal{E}} \|a - e\|$
  Modify environment with $a_{\text{new}} = e^*$
  Execute manipulation policy π^{R}
  if failure detected **then**
    Reset environment
    $a_{\text{old}} \leftarrow \emptyset$ ▷ Discard accumulated actions on failure
  else
    $a_{\text{old}} \leftarrow a_{\text{new}}$ ▷ Accumulate successful actions
  end if
end for

We expect the RL agent to gradually modify the environment by applying a finite number of predefined actions $(a_1, a_2, \ldots, a_n)$ in a way that induces failures. Each action $a \in \mathcal{A}$ corresponds to a specific environmental change such as action a_1 changes the position of an object by a fixed offset and action a_2 changes the color of an object. 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 configuration with a black table of size X under 50% lighting conditions. Algorithm 1 outlines the workflow for iteratively applying these discrete actions and evaluating the policy's performance in the perturbed environment.

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) [38] as the learning algorithm. Additionally, PPO provides stability and adaptability in high-dimensional environments. PPO [38] optimizes a clipped surrogate objective,

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio of current and old policies, and $\hat{A}_t$ is the advantage estimator at time t. For stability, the objective is clipped with the parameter ϵ. RoboMD policy, π^{MD}, is learned by interacting with the environment by changing the environment with $a_t \sim \pi^{\text{MD}}(a_t|s_t)$, receiving rewards for detecting a failure, and transitioning to subsequent states s_{t+1}. By iteratively refining π^{MD}, PPO guides the agent toward failure-inducing environment variations. After learning, RoboMD can tell us the probability of failure for each candidate FM in $\mathcal{C}$. We will further discuss this in 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. The limitation of this approach is its inability to diagnose failures beyond $\mathcal{C}$. We argue that in order to predict failures of such 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 special vision-language embedding that can be used to learn failures. With these two, we train RoboMD policy on the *continuous action space of the trained embedding*. Even if $\mathcal{C}$ does not fully cover the entire space of failures, the RoboMD deep RL algorithm 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 as a 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 RoboMD model on this continuous space to predict failures beyond the observed environment.

Training the Vision-Language Embedding. The objective of training the embedding is providing a continuous space that encodes some information about success-failures for the RL algorithm 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 $y \in \{\text{failure, success}\}$. 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 [39] to convert x_i^{vision} to visual features. Since inputs are images, vision transformer effectively captures spatial and contextual relationships within the visual input.
2) A CLIP encoder [40] to process semantic descriptions - Since robots operate in complex environments, we

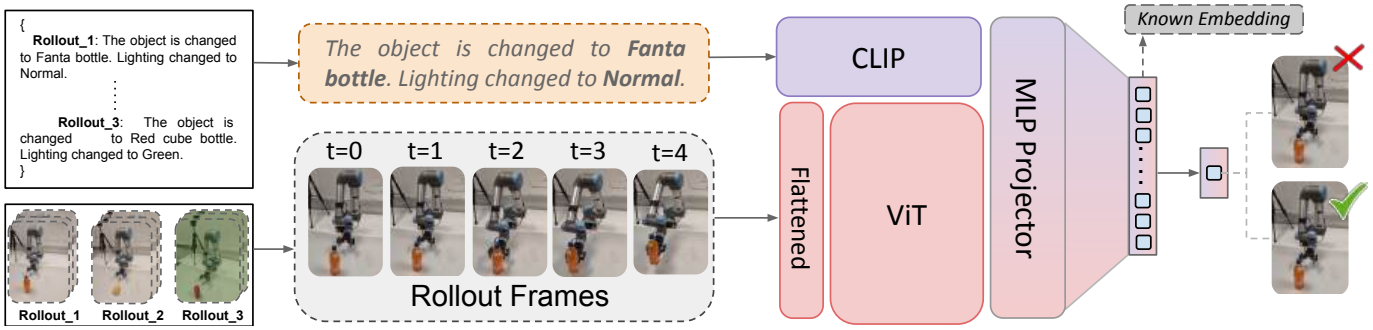

Fig. 3: The pipeline illustrates how rollouts with disruptions (e.g., object or lighting changes) are processed to learn meaningful embeddings. Text and visual data from the rollouts are embedded using CLIP and ViT, then projected through an MLP to generate text, image to failure aligned representations.

empirically found that providing language inputs of the task description helps the vision transformer focus on necessary features of the vision input, thus leading to a good final embedding with a few samples.

The outputs of the two backbones are concatenated and passed through Multi-Layer Perceptron (MLP) layers to form an embedding in $\mathbb{R}^{512}$,

$$\mathbf{e} = \text{MLP}_{\text{final}}\left[\text{Concatenate}(\text{ViT}(x^{\text{vision}}), \text{CLIP}(x^{\text{lang}}))\right].$$

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., different 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 a contrastive loss,

$$\sum_{i,j\in\mathcal{D}} \left[\mathbb{1}_{y_i=y_j} \cdot d_{ij} + \mathbb{1}_{y_i\neq y_j} \cdot \max(0, \text{margin} - d_{ij})\right], \quad (1)$$

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 margin is a hyperparameter that defines the minimum separation distance between embeddings of different classes.

Here, for each $(x^{\text{vision}}, x^{\text{lang}})$ pair, the model is trained to maximize the similarity between embeddings of similar actions, while minimizing the similarity between dissimilar actions. This contrastive learning objective ensures the embedding space reflects semantic relationships among actions, grouping similar actions while separating unrelated ones. $\mathbf{e}$ is used to predict the outcome (success or failure) based on the combined visual and textual inputs. This ensures that the model not only aligns visual and textual data but also captures task-specific information related to outcomes.

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 2, 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 $\mathcal{E}$.

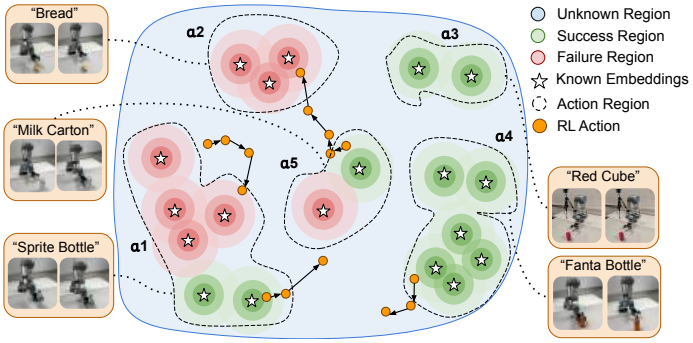

Fig. 4: Continuous Action Space Exploration. The diagram illustrates three types of regions in the action space: Unknown (blue), Success (green), and Failure (red). Known embeddings (stars) represent pre-computed reference points, which guide the exploration process. Orange circles depict actions taken by the RoboMD RL agent, with arrows indicating the sequence of transitions during exploration. Dashed boundaries indicate naturally formed action regions, grouping similar outcomes (e.g., all stars within an action region represent the same action, such as changing the cube color to red). The RoboMD RL agent systematically navigates the action space, transitioning across different regions and identifying failure modes. Since these traversals are always directed toward failures, the learned policy, π^{MD}, represents a failure distribution.

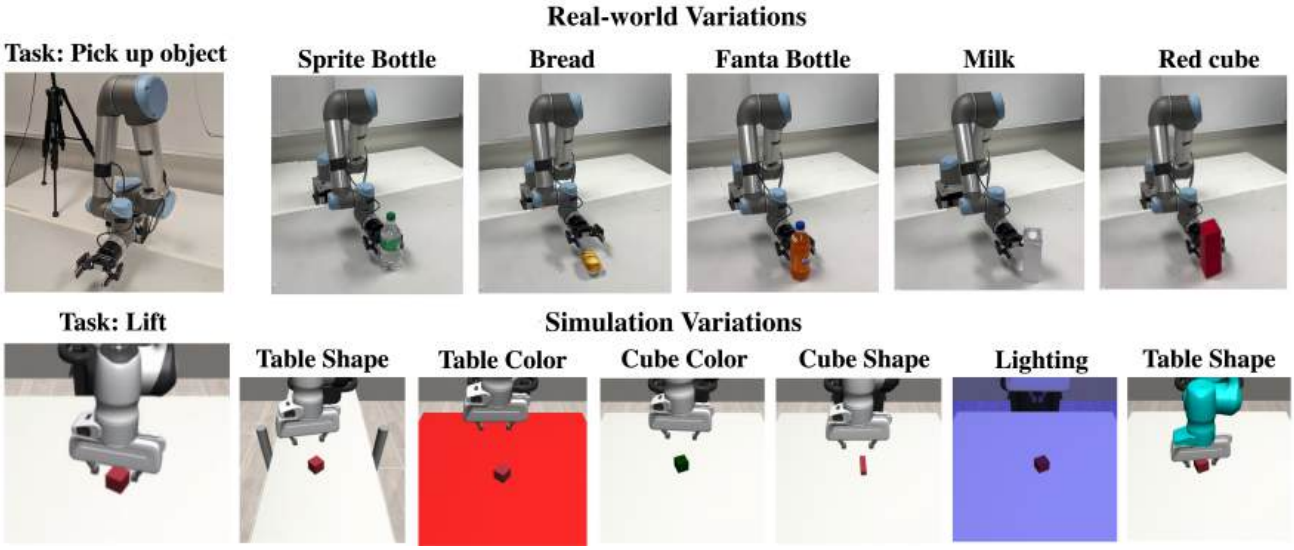

Fig. 5: Some environment variations for both simulation and real-world evaluation.

This is because significant deviations from $\mathcal{E}$ lead to uncertain or poorly understood regions. This rewards mechanism can be capture by,

$$R(s,a) = \begin{cases} \frac{C_{\text{failure}}}{\text{penalty}+1} - k \cdot \mathcal{N}(a), & \text{if failure,} \\ -\frac{C_{\text{success}}}{\text{horizon} \times (\text{penalty}+1)}, & \text{if success.} \end{cases} \tag{2}$$

Here, horizon represents the total episode length, which is the number of timesteps in a single rollout or trial. The penalty is proportional to the Euclidean distance between the current action a and the nearest known embedding $\mathcal{E}$. The action frequency penalty, $\mathcal{N}(a)$, counts the number of times the same action a has been taken consecutively within the episode. This approach allows the agent to explore adaptively while using prior knowledge about the surrounding regions, such as whether they are likely to lead to success or failure.

As illustrated in Figure 4, this process can be conceptualized as projecting a few failure-success demonstrations into a semantically meaningful embedding, which is then used to learn a generalized failure detection policy. After the first step, π^{MD} is learned by dynamically sampling the continuous embedding space, with many RL action sequences that direct toward failure-prone regions due to the reward. Note that, this process only requires sampling from the embedding space, which does not require policy roll-outs, making it a low resource task. Since the policy that it learns indicate how to take an action (i.e., change the environment) in such a way that the manipulation policy fails, the learned policy, π^{MD}, indicates the failure distribution.

C. Uncovering Failures

Because the RoboMD policy in previous sections has learned to go to high probability failure regions, we can directly utilize the policy distribution to uncover failures. In discrete action spaces, introduced in Section III-A, the RoboMD RL policy, $\pi^{\text{MD}}(a|s)$, maps states s to a probability distribution over the action set $\mathcal{A}$. This distribution reflects the likelihood of selecting each action under a given state s (i.e., image of the environment). For a given observation, the probability of selecting an action, a, (i.e., failure mode, given input) is defined as the softmax of RoboMD policy probability mass function (PMF),

$$\pi^{\text{MD}}(a \mid s) = \frac{\exp(f_a(s))}{\sum_{a' \in \mathcal{A}} \exp(f_{a'}(s))}, \tag{3}$$

where $f_a(s)$ is the unnormalized logit (score) for action a of the RoboMD policy.

In continuous spaces, described in Section III-B, the policy $\pi^{\text{MD}}(a \mid s)$ is represented as a probability density function (PDF), $p(a \mid s)$, parameterized by a multivariate Gaussian distribution. This PDF assigns density values to actions $a \in \mathbb{R}^d$ for a given state s. Continuous action spaces offer greater flexibility and granularity, allowing exploration beyond predefined actions. However, the probability of any single real-valued action point is zero because $P(X = x_0) = \lim_{\epsilon \to 0} \int_{x_0}^{x_0+\epsilon} f(x)\,dx = 0$. Although individual probabilities are zero, the ratios always have a value, making this formulation computationally tractable and stable, similar to how PPO maximizes a probability ratio, shown in Eq. (4). Similarly, we can compute which perturbation is more likely to result in a failure than another by computing,

$$\frac{p^{\text{MD}}(a = a_1 \mid s)}{p^{\text{MD}}(a = a_2 \mid s)}, \tag{4}$$

where s is the arbitrary observation environment, $s \in \mathcal{C}$ or $s \notin \mathcal{C}$. Note that this is different to typical RL rollouts, in which the objective is finding the most likely action $\arg\max p(a|s)$.

Since two actions are comparable in this setup, their embeddings can be passed into the policy distribution to directly compare log probabilities. These s can be arbitrary environments, $s \in \mathcal{C}$ or $s \notin \mathcal{C}$. However, if the observation is far away from $\mathcal{D}$, the reliability is reduced. The reliability, as a metric can be computed as $\min_{\mathbf{e} \in \mathcal{E}} \|\mathbf{e} - \mathbf{e}_s\|$.

TABLE I: Benchmark results of RL models and VLMs across different tasks for a BC MLP policy. RoboMD (PPO Continuous) consistently outperforms other baselines.

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**

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, 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. Experiments

In this section, we investigate the following questions:

1) How well does RoboMD detect FMs in seen scenarios compared to other RL approaches 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 [41] using datasets from RoboMimic [42] and MimicGen [43]. Benchmarking is performed across Lift, Stack, Threading, and Pick & Place tasks, which represent a range of common manipulation challenges with varying levels of difficulty.

RoboMD, which utilizes PPO, is evaluated alongside other RL models such as A2C [44] and SAC [45]. 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 [14, 5, 15, 16, 17], 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. As shown in Table I, ICL improves the performance of GPT-4o, particularly in the *Square* task. However, overall VLM performance remains below 60%, indicating that these models struggle with reliably predicting environment configurations.

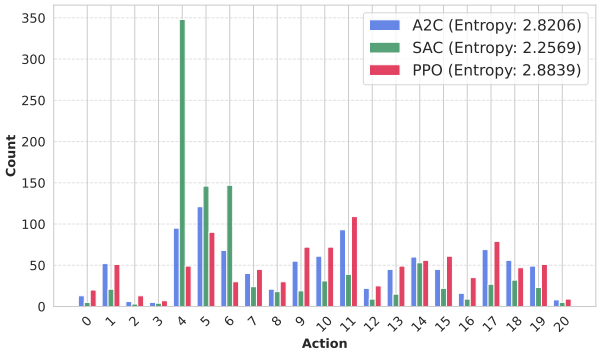

Fig. 6: Action diversity across RL algorithms. The X-axis represents individual actions applied to the task. See Fig. 18 for details on specific action perturbations.

TABLE II: Failure characteristics for discrete action spaces. Lower FSI values indicate more robust models.

Model	**Entropy** (↓)	**NFM** (↓)	**FSI** (↓)
IQL [46]	2.49	4	0.72
BCQ [47]	2.79	6	1.15
BC Transformer	2.47	5	0.98
HBC [48]	2.11	4	0.68
BC (Two-Image Input)	2.14	3	0.63
BC (Proprioceptive + Image)	2.58	3	0.75

To further evaluate the exploration characteristics of different RL algorithms, we analyze the action distributions of A2C, SAC, and PPO over 21 actions (environment variations). Each algorithm is trained on the same task for the same amount of timesteps, and we compute the entropy of their action distributions to quantify how diversely they explore the state-action space. Figure 6 illustrates the action distribution. Where entropy values are used as a measure of action diversity in Table II, PPO exhibits the highest entropy (2.8839), indicating a more diverse exploration, which is necessary for discovering a wide range of failure modes. In contrast, SAC (entropy: 2.2569) tends to be less exploratory, limiting its ability to uncover rare failure cases. A2C, while more exploratory than SAC, still falls short of PPO's broad coverage.

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, including Behavior Cloning (BC), Batch-Constrained Q-learning (BCQ), Conservative Q-Learning (CQL), and BC Transformer. 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.

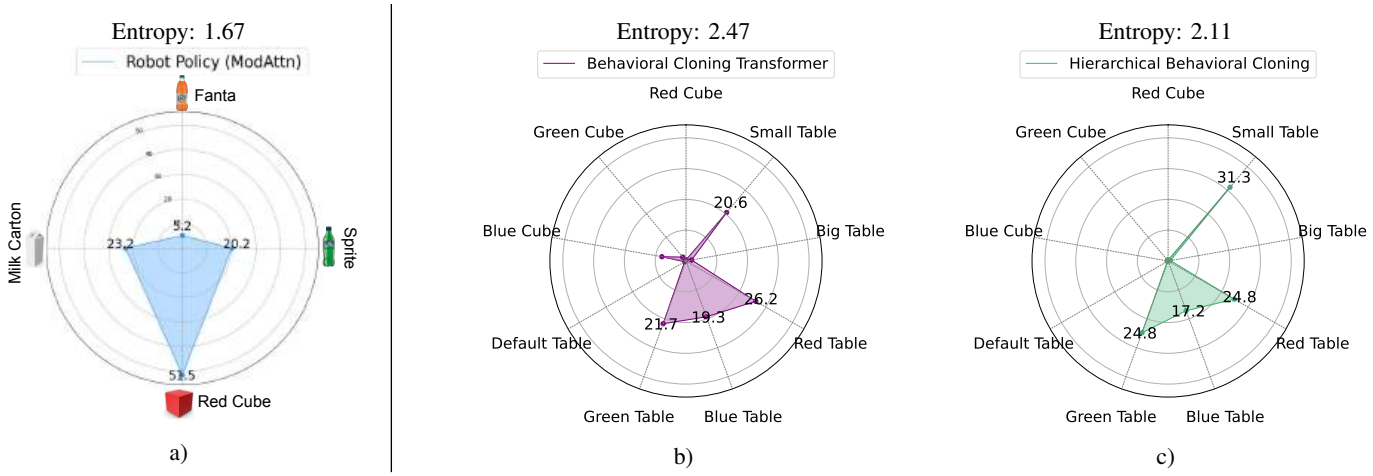

Fig. 7: Individual FM analysis of multiple models. Each radar plot represents the failure likelihood of a specific actions. The axes correspond to different environmental setups (e.g., Red Cube, Green Table, Blue Table) (a) for real-world setup and (b, c) for simulation, and the numbers indicate the probability of failure for actions under each configuration.

TABLE III: RoboMD failure detection accuracy across different tasks using Behavior Cloning (BC), Batch-Constrained Q-learning (BCQ), HBC, BC Transformer, and Diffusion.

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%

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. The goal is to determine whether failure detection and ranking remain consistent when encountering novel actions beyond the training distribution. We conduct experiments in both real robot and simulated environments. For simulated environments, we use an identical setup as described in Section IV-A. In the real world, we employ a UR5e robot with a camera mounted next to it, providing a side view of both the robot and the objects in the environment.

Seen Environments: Figure 7 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. In contrast, higher entropy indicates a broader distribution of failures

To quantify the failure characteristics of each model, Table II summarizes the entropy values and the number of failure modes (FM) identified for each model. Entropy measures the diversity of failure likelihoods across configurations, while fewer failure modes indicate a more robust 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 $P_{\text{failure}})$ represents the probability of failure for action a_i, and W_i is the normalized weight such that the failure mode with the highest probability is assigned a weight of 1, while others are scaled proportionally. Models such as *HBC* demonstrate lower entropy, FSI and fewer failure modes, 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. Accuracy on this dataset tends to be similar to seen actions, indicating robustness. 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. 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. 8 illustrates the failure distribution of the manipulation policy on the Lift task before and after fine-tuning with failure-inducing samples. The pretrained policy exhibits higher failure probabilities across multiple actions (environment conditions), deviating significantly from the ideal distribution (dashed black). Fine-tuning reduces failures (details in Appendix IV), particularly in previously high-failure regions (A6, A5, A7), which correspond to table colors red, green, and blue. 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.

TABLE IV: Comparison of rankings for failure-inducing actions in continuous and discrete action spaces. a_r represent actions performed in the real robot environment a_r1 = "Bread" (Unseen), a_r2 = "Red Cube", a_r3 = "Milk Carton", a_r4 = "Sprite". a_s represent the actions performed in simluated environment a_s1 = "Red Table", a_s2 = "Black Table" (Unseen), a_s3 = "Green Lighting." Rank consistency indicates whether the rankings are preserved across the two formulations. The accuracy is computed over 21 unseen environment variations.

Task ID	Algorithm	Continuous Rank	Ground Truth Rank	Consistency	Accuracy
Real Robot (UR5e)	ModAttn [49]	$a_r1 > a_r2 > a_r3 > a_r4$	$a_r1 > a_r2 > a_r3 > a_r4$	✓	-
Sim. Can	HBC [48]	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	61%
Sim. Square	Diffusion [6]	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	68%
Sim. Stack	BCQ [47]	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	80%
Sim. Threading	BC Transformer	$a_s1 > a_s2 > a_s3$	$a_s1 = a_s2 > a_s3$	✓	74%

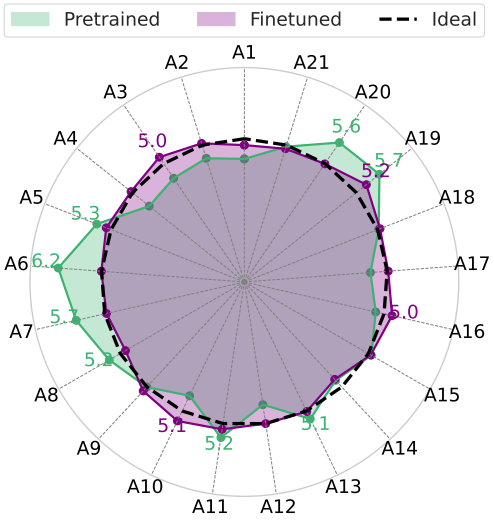

Fig. 8: Failure distribution before and after fine-tuning "Lift" behavior cloning policy on failure modes chosen by RoboMD. The radar plot illustrates failure probabilities across different actions, where the ideal distribution (dashed black) represents zero failure across all actions. For action list refer Appendix III-A.

D. 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. 9: (1) embeddings trained with only BCE loss, (2) embeddings trained with BCE and Contrastive loss, and (3) embeddings trained using only an image encoder. 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.

V. Limitations and Conclusions

We introduced RoboMD, a framework designed to diagnose failure modes in robot manipulation policies. By leveraging deep RL with both discrete and continuous action spaces,

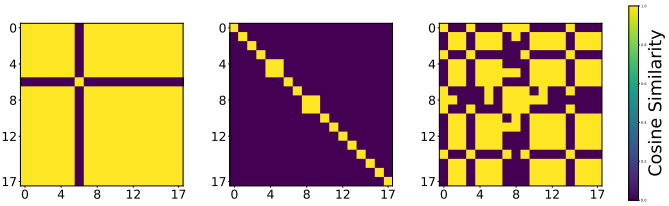

Fig. 9: Confusion matrices of embeddings trained using a) Binary Cross-Entropy (BCE) loss, b) BCE and Contrastive Loss, and c) both losses but no text encoder. Diagonal is better.

TABLE V: Deviation scores measuring embedding quality across different loss functions BCE and Contr (contrastive) also with (Image + Textual) backbone and only Image backbone. Lower MSE and Frobenius norm distance (between the confusion matrix and identity matrix) values indicate embeddings that are closer to the ideal diagonal structure (i.e., better separation between actions).

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

RoboMD identifies vulnerabilities across diverse configurations, offering actionable insights for targeted policy improvements. The experimental results demonstrate that RoboMD, particularly with PPO in continuous action spaces, consistently outperforms baseline RL models and VLMs. It captures nuances in failure scenarios, ranks failure-inducing actions, and generalizes diagnosis to unseen environment variations. Despite its promising performance, RoboMD is not without limitations. While RoboMD demonstrates significant improvements in diagnosing FMs, as any other method, its performance declines when querying about the failure likelihood of completely new environments that are farther away from known environments. Future work will focus on training a generalist PPO model for RoboMD for combined tasks and environment variations. In summary, RoboMD proposes a new framework for failure analysis in manipulation policies, offering a foundation for improving their robustness in unstructured environments.

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] 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.

[5] 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.

[6] 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.

[7] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. URL https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Samuel Wilson, Tobias Fischer, Feras Dayoub, Dimity Miller, and Niko Sünderhauf. SAFE: Sensitivity-aware features for out-of-distribution object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23565–23576, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/papers/Wilson_SAFE_Sensitivity-Aware_Features_for_Out-of-Distribution_Object_Detection_ICCV_2023_paper.pdf.

[24] Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. Pytorch-ood: A library for out-of-distribution detection based on pytorch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4351–4360, 2022. URL https://openaccess.thecvf.com/content/CVPR2022W/HCIS/papers/Kirchheim_PyTorch-OOD_A_Library_for_Out-of-Distribution_Detection_Based_on_PyTorch_CVPRW_2022_paper.pdf.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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/.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation frame-

work and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. URL https://arxiv.org/abs/2009.12293.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

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 10.

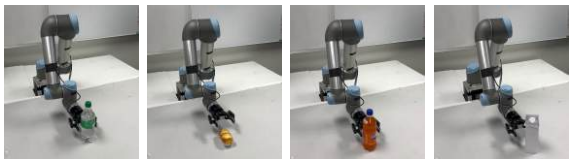

Fig. 10: Scenes from experiments on real world robot

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 10.

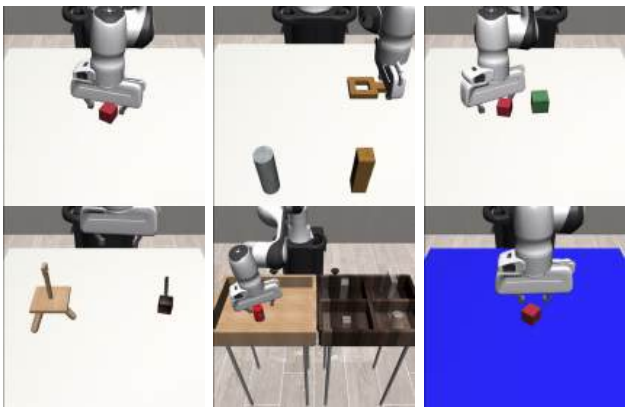

Fig. 11: Scenes from experiments on Robosuite

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

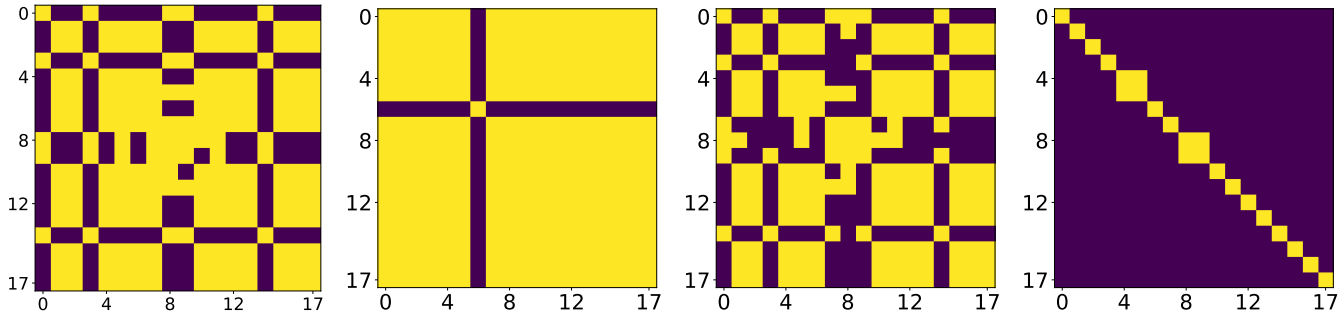

Fig. 12: The order in which the confusion matrix is a) Image Ecoder + BCE b) Image + Text Encoder + BCE loss c) Image Encoder + BCE + Contrastive loss d) Image + Text Encoder + BCE + Contrastive loss

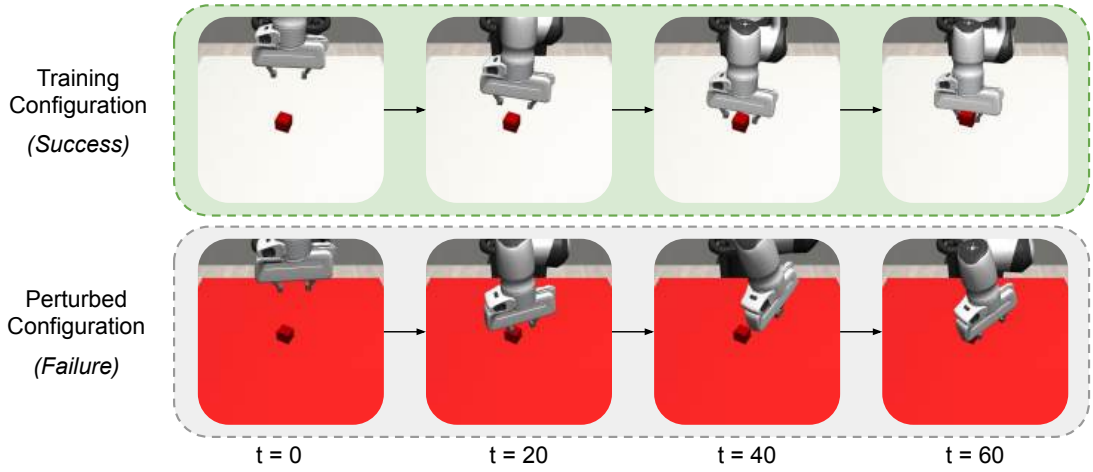

Fig. 13: Testing Robustness Under Visual Perturbations: Successful Rollout in Training vs. Failure Induced by Red Table Distraction

exploration by learning an agent that actively seeks out environmental configurations likely to induce policy failures. This 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 13 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.

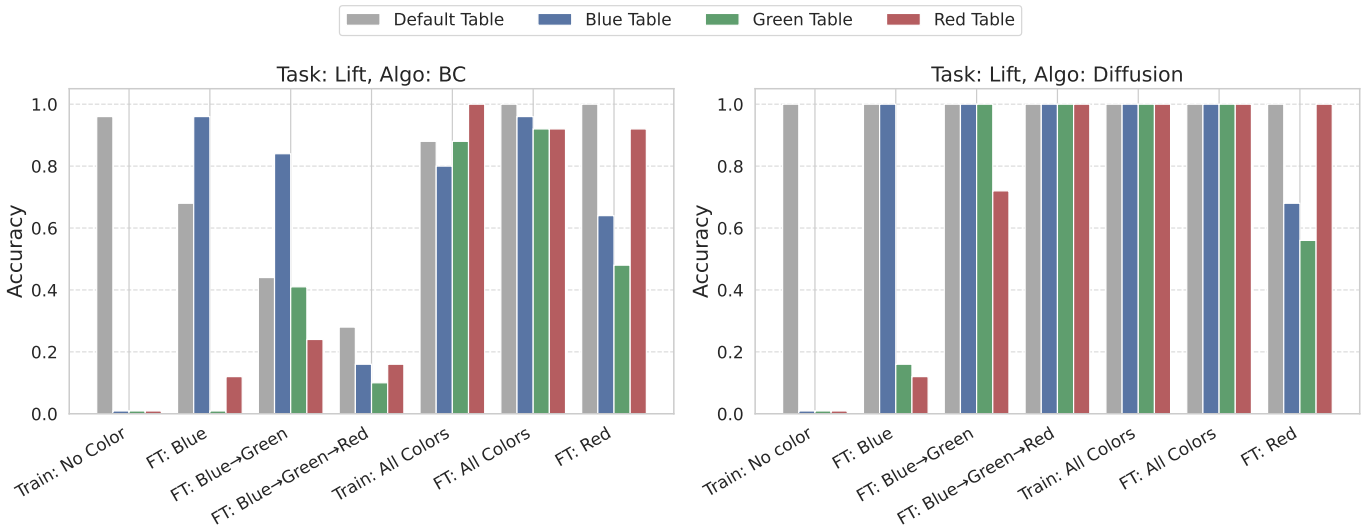

Fig. 14: Performance comparison of behavior cloning (BC) and diffusion-based policies on the Lift task before and after fine-tuning with failure-inducing samples. Each bar represents the success rate of the policy across different **table colors**.

III. Continuous Action Space Embedding

Embedding actions in a continuous space is crucial for efficiently capturing the underlying structure of decision-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 8:

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 12 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 embed-

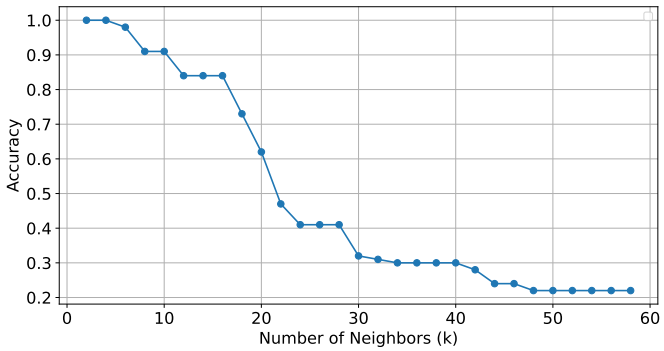

Fig. 15: kNN Accuracy Drop with Increasing k in Continuous Action Space Embeddings

dings should be locally consistent, meaning that a small k (considering only close neighbors) should yield high accuracy, while increasing k (incorporating more distant neighbors) may introduce noise and reduce accuracy as shown in Fig 15.

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.

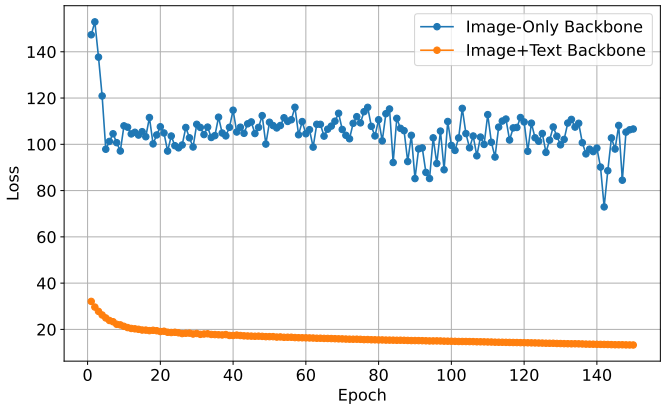

Fig. 16: Training loss for training action representations

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 14. 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 17, 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.

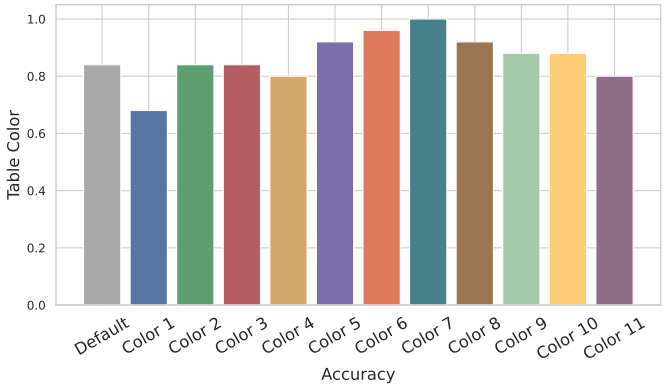

Fig. 17: BC lift finetuned on a combined dataset of 12 different Table colors

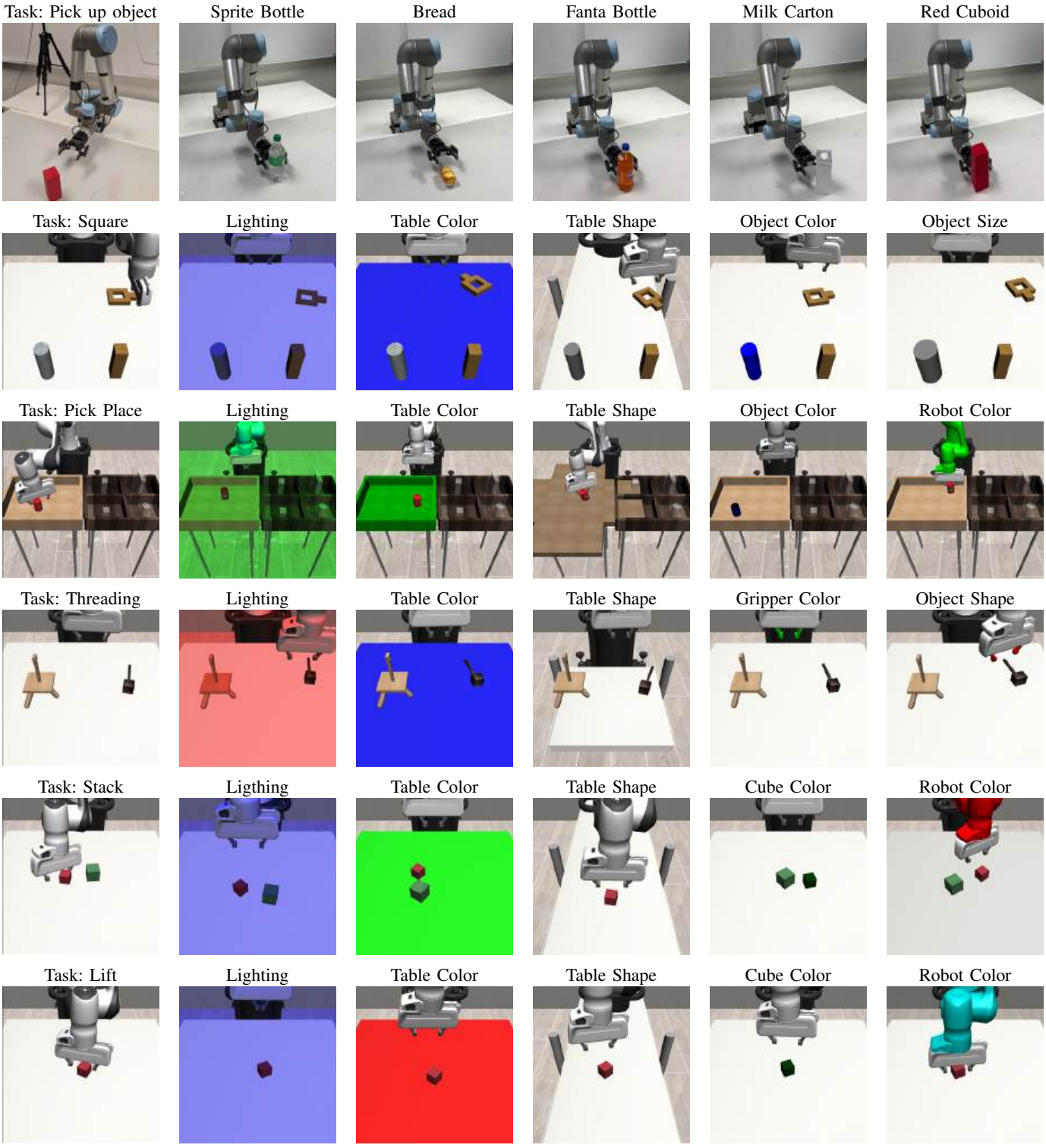

Fig. 18: Environmental and Object Perturbations on Manipulation Tasks