FAILGUARD: AN OOD-SIMULATION APPROACH FOR PROACTIVE FAILURE DETECTION IN VISUOMOTOR POLICY LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Failure detection is a critical capability for ensuring the safety of robotic systems, which can anticipate and avoid irreversible harm to the environment, the robot itself, or to the human being during interacting with the physical world. However, failure detection in real-world robotic manipulation has long been challenged due to the inefficiency and potential risks when collecting diverse failure data through rollouts. In this paper, we introduce a method named **FailGuard** to achieve failure detection for robot visuomotor manipulation policy learning. To learn a failure detector, we only collect success rollouts from the pre-trained manipulation policy and augment the rollouts to simulate failure cases for detection learning. After trained with the collected data, the failure detector can achieve runtime failure detection and early stop the impending failure, improving the success rate and safety of the robot manipulation. We evaluate our method on both RoboSuite benchmark and real world tasks. The experimental results show that our proposed method outperforms several different kinds of baseline methods, and can effectively prevent failure cases during manipulation. Project Website: `https://zuyu3.github.io/failguard.github.io/`

1 INTRODUCTION

With recent advances in deep learning, researchers have developed different kinds of learning-based algorithms for visuomotor manipulation policy learning. Approaches such as imitation learning (Chi et al., 2023; Florence et al., 2022; Mandlekar et al., 2021) and reinforcement learning (Wang et al., 2019; Schulman et al., 2017; Casas, 2017; Haarnoja et al., 2018; Fujimoto et al., 2018) have enabled robots to acquire complex manipulation skills directly from visual or low-dim sensory input.

However, since current mainstream robotic learning methods typically train robot policies using only success demonstration trajectories, the sensors' input will inevitably encounter scenarios that were unseen during training—namely, out-of-distribution (OoD) situations—when the policy makes an incorrect action. In such OoD scenarios, the policy's output becomes undefined behavior, which can further lead the robotic system to perform erroneous actions and ultimately cause task failure. ***As a result, for real-world deployment, the ability of a robotic system to anticipate and avoid such failure is critical and essential to avert irreversible harm to the environment or humans.***

Compared to general out-of-distribution detection tasks in deep learning domain, robot manipulation failure detection requires consideration of temporal information in the detection process, as it is a decision-making process based on sequential input. Furthermore, given that failure scenarios are diverse and typically follow a long-tail distribution, collecting examples of all possible failure cases to cover the entire distribution is impossible. Second, permitting failures during real-world robotic experiments can cause irreversible harm to humans, the environment, or the robot itself, resulting in high costs and risks.

To address these issues, in this paper, we propose a method that learns failure detection via failure-simulation. Instead of collecting failures from real robot manipulation rollouts, our proposed system augments the success trajectories generated by the pre-train manipulation policy to simulate the failure-prone data. To better reduce the gap between the simulated failure-prone data and the ground truth failure data, we introduce a reasonable way to augment the success data, and provide a theoretical analysis. After gathering the data, we utilize supervise learning and contrastive learning

to train a failure detector. Finally, we deploy the trained failure detector in the robotic manipulation process to estimate potential risks of failure, thereby enhancing the safety and success rate.

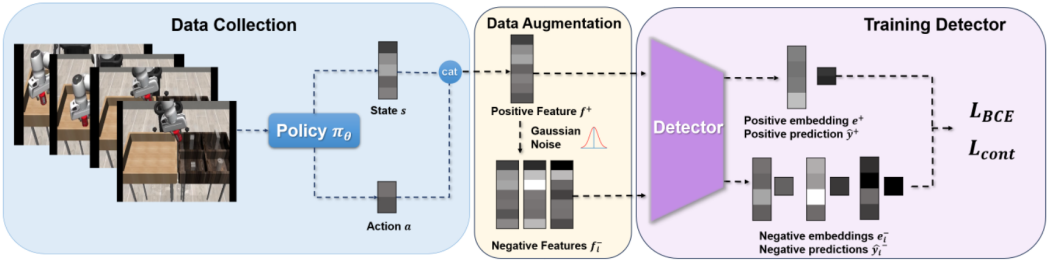

Figure 1: Architecture of the proposed algorithm. State s and action a are extracted by a pretrained policy π_θ from environment observation, and then concatenated to positive feature f^+. Then data augmentation is applied to f^+ to generate negative features f^- by adding dynamic scale Gaussian noise. The detector is trained using both contrastive loss and binary cross-entropy loss with f^+ and f^-.

To evaluate our proposed method, we conduct experiments under both simulation and real-world environment and report the AUROC and AUPRC metrics. The result shows that our method significantly outperforms other baselines. Furthermore, we conducted restart experiments, validating that our method has applicability during the deployment of robot manipulation policy and can assist in improving the success rate of policy in completing the task.

In summary, our work makes the following three contributions:

1. We propose a novel data augmentation method which generates failure-prone trajectories by adding adaptive Gaussian noise to only the collected success trajectories. With this approach, there is no need to collect dangerous failure examples in the real environment, enabling the acquisition of failure trajectory data even in settings with zero tolerance for failures.
2. We provide a theoretical analysis for the validity and effectiveness of our data augmentation method. The theorem shows that, by increasing the number of collected success trajectories, our chosen Gaussian-noise injection scheme can lower the upper bound of the failure detection error rate, thereby ensuring reliable failure detection performance of our model.
3. We train and evaluate our detector model under both simulated and real-world environments using multiple policy architectures. The experimental results demonstrate that our method – **FailGuard**'s detection capability significantly outperforms baselines across various tasks, and can substantially improve task success rates by guiding appropriate human-in-the-loop resets.

2 Related Work

2.1 Out-of-Distribution Detection

Reconstruction-based and embedding-based methods are two prominent approaches for out-of-distribution (OOD) detection. Reconstruction-based methods, such as autoencoders (Gong et al., 2019) and GANs (Goodfellow et al., 2014), are trained exclusively on normal data to learn a compressed representation. However, their applicability is limited in domains like robot manipulation, which are characterized by high-dimensional data and sample scarcity, making effective training impractical (Zavrtanik et al., 2021).

Conversely, embedding-based methods have demonstrated state-of-the-art (SotA) performance (Defard et al., 2021; Deng & Li, 2022; Liu et al., 2023). These techniques project input features into an embedding space where anomalies are identified by their distance from clusters of normal data. Nonetheless, they often incur significant computational and memory overhead due to their reliance on complex statistical algorithms (Defard et al., 2021; Roth et al., 2022).

2.2 Failure Detection in Robot Manipulation Learning

Currently, robots policy is trained mainly by reinforcement learning or imitation learning, aiming to complete given tasks. When task complexity increases or requires long-horizon decision, the

environment's reward signals tend to become sparse, which severely limits the effectiveness of reinforcement learning (Vecerik et al., 2017). Imitation learning, on the other hand, attempts to model the hidden action pattern behind successful trajectories to learn a policy, whose observation space is often lack of abnormal samples. Due their own weaknesses, these methods have a high likelihood of failure when encountering disturbance (Ak et al., 2023) during execution, further necessitates the need of reliable failure detection methods. Although many recent studies (Bharadhwaj et al., 2020; Cheong et al., 2018; Pinto et al., 2017; Eysenbach et al., 2017) attempt to mitigate risks by designing specific reward, these approaches often only address a single kind of risks, such as the affordance risks, and balancing risk and reward can confuse agent during training, ultimately resulting in a sub-optimal policy. ***As risks are inevitable in real robot deployments, it is necessary to consider the failure detection in robot manipulation learning.*** 1

3 Problem Formulation

We formulate the problem of robot manipulation as a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the continuous state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ denotes the state transition probability function, $\mathcal{P}(s'|s, a)$ is the probability of transitioning from state s to state s' after taking action a, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor.

In the setting of robot learning (Chi et al., 2023; Mandlekar et al., 2021; Fujimoto et al., 2018), the agent learns a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ from given data $\mathrm{D}_{\mathrm{train}}$ to accomplish a specific manipulation task. In general, a policy π_θ follows a two-stage architecture. An encoder $\mathcal{E} : S \to H$ extracts features from the observation space S and encodes them into a continuous hidden state space $H = \mathbb{R}^K$. Then a decoder $\mathcal{D} : S \to H$ uses the features produced by the encoder to generate the actions to be taken. We claim that this two-stage abstraction applies to the vast majority of policies, and build our method upon this structure; that is, our approach is applicable to any policy model conforming to this architecture.

During deployment, the agent may encounter states that differ significantly from those in D_{train}, which we refer to as out-of-distribution(OoD) states, and have a high tendency to induce erroneous actions, ultimately resulting in task failure. The objective of failure detection in this context is to detect such out-of-distribution states by learning a function $\phi : \mathcal{S} \to \{0, 1\}$, where 1 means failure-prone. This failure detection mechanism aims to identify states that deviate from the distribution of training data, enhancing the robustness and safety of the policy π_θ in real-world applications.

4 Method

In this paper, we introduce **FailGuard** to address failure detection problem for robotic manipulation. Based on the setting introduced in Sec. 3, our method, as illustrated in Fig. 1, consists of three main components: data collection (Sec. 4.1), data augmentation (Sec. 4.2), and detector learning (Sec. 4.3). Each component is designed to address specific challenges in failure detection for imitation learning in robotic manipulation. Below, we discuss the key design choices for each component, and in Section 4.4 we provide our theoretical analysis, which demonstrates that our data augmentation approach is sound and effective. Finally, we describe how to achieve failure distribution detection during deployment in Sec. 4.5.

4.1 Data Collection

To detect failure-prone states during robot manipulation, the proposed system first need to collect trajectory data for training. We execute a pre-trained policy π_θ in a specified environment and obtain trajectory data from success rollouts. We emphasize that our method does not depend on any particular type of policy; hence, the policy can be trained by any approach, such as imitation learning or reinforcement learning. The states from these successful trajectories are regarded as success samples $\mathrm{D}_{succ} = \{\{s_i^+, a_i^+\}_{i=1}^N\}$, since they guide the agent to complete the task successfully. We assume that these successful trajectories contain valid information about state transitions and state–action decisions, which can instruct the detector to learn what normal trajectories should look like.

For a binary detector ϕ, it is necessary to collect negative samples—i.e., failure-prone data $\mathrm{D}_{fail} = \{\{s_i^-, a_i^-\}_{i=1}^N\}$—for training. However, we do not directly obtain failure rollouts from the envi-

ronment as negative samples. Instead, we augment success trajectories to simulate failure-prone states.

This is because, first, when using failure samples, identifying the specific timestep that causes task failure is complex and ambiguous. A simple and intuitive approach would be to label the entire trajectory as anomalous; yet this is infeasible. The policy may only perform an anomalous action at a single critical moment that leads to failure, while all preceding states are perfectly correct. Labeling those normal states as failure-prone would confuse the detector, potentially yielding a suboptimal ϕ or even preventing convergence during training.

Second, for well-performing robotic manipulation policies, failure cases are relatively rare, and various types of failure-prone states often follow a long-tailed distribution. Directly using only the failures encountered during policy deployment risks insufficient sample quality and diversity.

Additionally, in the context of robot manipulation learning, particularly in real-world experiments, actively designing strategies to collect failure cases can be both costly and dangerous. In certain scenarios, policy failure may inflict irreversible damage on the environment or endanger human safety, which is unacceptable.

In light of these considerations, our system opts to generate failure-prone data by augmenting success trajectories rather than relying on real failure rollouts.

4.2 Data Augmentation for OoD Simulation

To simulate failure-prone states, we apply data augmentation to success trajectory data. We choose to add noise in the hidden state space H rather than in the raw observation space. The hidden state space typically has a relatively low dimensionality, making our noise injection more effective compared to the observation space. Moreover, although the image space is relatively sparse, performing data augmentation in the denser encoding space can generate a richer and more valuable set of failure-prone scenarios. This is because adding noise directly to images only simulates blurred conditions caused by camera issues such as defocus, which are very unlikely to occur with the high-performance cameras used in robotic manipulation, and therefore offer limited value. In contrast, true failure-prone trajectories are often clear, plausible images, but the robot's behavior is anomalous, leading to a failure to correctly manipulate the target object and complete the task. Such "plausible" images are difficult to produce through simple, intuitive augmentations and are also challenging to collect, as explained in Section 4.1.

To create failure-prone features, we add Gaussian noise to each dimension of the hidden state embedding vector h_i. Specifically, for each dimension $k \in \{1, \ldots, K\}$ of a success trajectory with T timesteps, we compute the variance σ_k^2 over the entire trajectory, i.e. $\sigma_k^2 = var\{e_k^1, e_k^2, \cdots, e_k^T\}$, and use it to scale the Gaussian noise $\epsilon_k \sim \mathcal{N}(0, \sigma_k^2)$ for the trajectory. This scaled noise is added to h_i to obtain an augmented embedding $\tilde{h}_i = h_i + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \ldots, \epsilon_K]$.

This augmentation process introduces controlled perturbations based on the natural variance in each embedding dimension, creating a richer set of failure-prone cases that closely simulate realistic cases. Here are the discussions on key design choices:

The Scale of the Gaussian Noise. When augmenting embedded features, adding excessive noise to each feature dimension can corrupt the original feature information, meaning that the corresponding observations will never be encountered during policy deployment. Although the detector may correctly classify such extreme cases as failures, this yields limited practical benefit. Conversely, if the noise scale is too small, the noise-augmented features will be nearly identical to the original features. Such overly similar positive and negative samples greatly increase the difficulty of model training and may prevent convergence. Moreover, these subtle perturbations may correspond to recoverable states in the environment, where the agent can return to normal operation after corrective actions. If the detector mistakenly labels these states as failures, the policy's robustness and stability will be compromised.

To generate noise with a reasonable scale, we employ the augmentation approach described previously in Sec. 4.2 to augment success trajectory data. We also provide theoretical analysis on this design choice in Sec. 4.4.

Why Not Use Variance at Each Timestep. To generate failure-prone data samples, a seemingly intuitive approach is to compute the variance at each timestep across all trajectories and use it to

set the noise variance for that timestep during data augmentation. However, trajectories in robotic manipulation typically vary in length, so features at the same timestep are misaligned across trajectories. For example, at timestep 10, one trajectory may have already closed the gripper to attempt an object grasp, whereas another may still be positioning the gripper and has not yet closed it. Even if we can identify each critical timestep, handling the states between these key points remains challenging: how to pad or compress the states between keypoints is an unresolved problem. Simply discarding these intervals would lose essential information about state transitions.

4.3 Failure-Prone Detection Learning

After collecting and augmenting the data, we train a detector using the original and augmented features through contrastive learning, aiming for the detector to correctly distinguish positive and negative samples and learn how to determine whether the states are failure-prone in the environment.

The detector consists of two parts: an feature extractor that generates embeddings from the features to be detected, and an MLP that maps these embeddings to a failure tendency score. Given the sequence of the embedding features and actions $\{h_i, a_i\}_{i=M}^{N}$, the extractor generates a feature f that embed the sequential manipulation information. To learn the failure detection, we input paired features, including one original feature f^+ and c noise-augmented features f^-. The encoder in the detector generates embeddings e^+ and e_i^-, where $i \in \{1, 2, \ldots, c\}$. The detector's MLP layer then generates the failure tendency score $\hat{y}$ for each embedding.

Our loss function has two components to compensate each other: a contrastive learning loss for the embeddings and a binary cross-entropy (BCE) loss for $\hat{y}$.

$$\mathcal{L} = \mathcal{L}_{cont} + \alpha \cdot \mathcal{L}_{BCE}$$

where y is the ground truth label, and $\hat{y}$ is the prediction of the detector

4.4 Theoretical Analysis

In this section, we provide a theoretical analysis of our method. We consider an augmented dataset, $\widetilde{D}$, containing positive (success) trajectories $\{x^+\}$ and negative (failure) trajectories $\{x^-\}$. The negative samples are generated by adding Gaussian noise to positive trajectories. This process may lead to some perturbed negatives remaining in positive data domain, resulting in mislabeling. We denote the probability of this label noise by η, and we assume the mild condition that $\eta < \frac{1}{2}$. This setup frames our problem as a binary classification task with label noise.

Our main theoretical result establishes a generalization bound for our method. For the full derivation, please see Appendix A.

Theorem 1. *Assume our loss function l is G-Lipschitz continuous w.r.t. $\|\cdot\|_\infty$ and $\|f(x)\|_2 \leq R$. For $\delta \in (0, 1)$, with probability at least $1 - \delta$ for all $f \in \mathcal{F}$ we have:*

$$R_D(f) = \frac{1}{1-2\eta}\left(\hat{R}_{\widetilde{D}}(f) + O\left(\frac{GR\mathcal{B}\log^2(nRk)}{n\sqrt{k}} + \mathcal{B}\sqrt{\frac{\log(1/\delta)}{n}}\right) - \eta\right) \tag{1}$$

where n is the number of positive samples, k is the number of negative samples, and $\mathcal{B}$ is a global upper bound on the contrastive score.

Remark: This theorem demonstrates two key points:

1. The failure detection error decreases as the number of demonstration data points (n) grows.
2. Reducing the label noise rate η—that is, narrowing the gap between augmented failure-prone data and ground-truth failure data—can significantly reduce the error rate.

4.5 Failure Detection during Deployment

When deploy the manipulation policy, the pre-trained policy network π_θ interacts with the environment, generating actions a_i based on observation o_i. At each timestep, the detector evaluates the state using the feature embedding h_i, which is extracted by the policy network from the observation, and the corresponding action. The detector is designed to estimate whether the current state exhibits any signs of "out-of-distribution" which could lead to failure risk.

If the detector determines that the likelihood of failure-prone is high, indicated by an output exceeding a predefined threshold τ, an early-stop mechanism is triggered. This early-stop strategy

terminates the current rollout and reset the robot to the initial state in order to prevent the robot from causing further potential disruptions or damage to the environment. After an early-stop occurs, human being can intervene to restore the environment to an ideal state as needed, allowing the robot to resume the task. The reason why we choose human-in-the-loop error correction here is that, the robot may have already caused irreversible changes to the environment unable for the robot alone to reset to a valid state. For example, the object to be manipulated may have fallen off the work platform, beyond the robot's reach. In these situations, human's intervention is necessary.

5 EXPERIMENTS

FailGuard aims to effectively detect failure-prone occurrences during agent execution and, by guiding timely restarts, ensure the safe and normal execution of the Policy, thereby improving success rates. To evaluate the effectiveness of our method, we conducted experiments in both simulation (Sec. 5.2) and real-world environments (Sec. 5.3). The results show that our method significantly outperforms the Baseline model and yields a considerable improvement in success rates through Restarts. In this section, we will describe our implementation details, experimental settings, and results.

Table 1: AUROC and AUPRC on low-dimensional and image tasks

Method	AUROC (low-dimensional tasks)					AUROC (image tasks)				
	lift	square	can	transport	tool hang	lift	square	can	transport	tool hang
SVDDDEED	0.950	0.696	0.885	0.420	0.781	0.392	0.796	0.749	0.851	0.893
LSTMED	1.000	0.747	0.823	0.959	0.750	1.000	0.799	0.877	0.909	0.881
SimpleNet	1.000	0.874	0.855	0.944	0.739	1.000	0.928	0.944	0.937	0.912
SAA+	0.777	0.479	0.449	0.556	0.339	0.050	0.392	0.614	0.804	0.475
AnomalyCLIP	0.917	0.670	0.955	0.386	0.676	0.103	0.845	0.796	0.868	0.942
Ours	**1.000**	**0.999**	**1.000**	**0.968**	**0.849**	**1.000**	**1.000**	**1.000**	**0.999**	**1.000**
Method	AUPRC (low-dimensional tasks)					AUPRC (image tasks)				
	lift	square	can	transport	tool hang	lift	square	can	transport	tool hang
SVDDDEED	0.914	0.743	0.602	0.319	0.583	0.952	0.758	0.629	0.985	0.830
LSTMED	0.806	0.859	0.799	0.933	0.915	0.983	0.988	0.977	0.985	0.974
SimpleNet	0.921	0.507	0.507	0.437	0.578	0.937	0.982	0.698	0.975	0.876
SAA+	0.454	0.585	0.497	0.498	0.594	0.931	0.781	0.675	0.934	0.816
AnomalyCLIP	0.813	0.627	0.771	0.655	0.500	0.967	0.640	0.831	0.967	0.822
Ours	**1.000**	**0.996**	**1.000**	**0.939**	**0.993**	**1.000**	**0.990**	**0.970**	**0.990**	**0.998**

5.1 EXPERIMENTAL SETTING AND METRICS

We evaluate failure detection performance using the Area Under the Receiver Operating Characteristic (AUROC) and the Area Under the Precision-Recall curve (AUPRC). The anomaly score for each trajectory is determined by the maximum detector output observed across all timesteps, with labels assigned based on final task success.

To assess our restart mechanism, **FailGuard**, we measure the improvement in the policy's success rate under simulated disturbances. These disturbances, applied with a 0.3 probability per rollout, include either injected action noise or an abrupt environmental shift to an OoD state. If the detector's output exceeds a predefined threshold, the current attempt is terminated, the environment is reset to its nominal state, and the policy is restarted. We permit up to two restarts, meaning a task only fails if all three attempts are unsuccessful.

We argue that our restart experiment setting is realistic. In real production environments, a robot's actions may encounter disturbances, or the environmental state may change due to various factors such as wind, accidental human collisions, or even malicious interference. Under such circumstances, the robot's task is likely to fail and may even cause irreversible consequences. Therefore, allowing for early-stop and restart is essential. We chose to reset the environment to a normal state because, in many situations, changes in the environment exceed the robot's ability to self-correct, necessitating human intervention to restore normalcy.

We compare our method against five strong baselines with distinct characteristics, including reconstruction-based and zero-shot methods. They are **SVDDED** (Ruff et al., 2018),

LSTMED (Deng et al., 2023), **SimpleNet** (Liu et al., 2023), **AnomalyClip** (Zhou et al., 2023) and **SAA+** (Cao et al., 2023). Detailed introduction for all baselines can be found in Appendix B

5.2 Simulation Experiments

5.2.1 Simulation Environment

Robomimic is a large-scale benchmark designed to study imitation learning and offline reinforcement learning in robotic manipulation consisting of 5 tasks. In the robomimic simulation, we created an out-of-distribution environment by altering certain parameters, such as object initial locations and object colors, in addition to the original data collection setting. We then tested our method's performance in both in-domain and out-of-distribution environments, evaluating its effectiveness on policies based on low-dim states or image observations. Detailed OoD settings can be found in Table 5

5.2.2 Detecting failure-prone states

Our primary experiment evaluates the method's effectiveness in detecting failure states. As shown in Table 1, our method achieves SotA results across all tasks and metrics, significantly outperforming the baselines. This demonstrates its robustness and effectiveness for failure detection across diverse manipulation policies.

Furthermore, the results underscore the importance of temporal modeling. Methods incorporating temporal context, like LSTMED, excel on long-horizon tasks (e.g., **transport image**, **square image**), whereas single-frame methods consistently underperform.

5.2.3 Restart Experiment

To assess if our method improves task completion, we performed rollouts allowing our detector to trigger policy restarts, per the protocol in Sec. 5.1. As shown in Table 2, this restart mechanism significantly boosts success rates across all tasks, confirming the method's practical utility.

Task	Original	With restart	Improvement
Lift image	56.0%	83.0%	↑ **27.0%**
Can image	26.0%	50.8%	↑ **24.8%**
Square image	7.5%	17.0%	↑ **9.5%**
Tool Hang image	31.5%	41.3%	↑ **9.8%**
Transport image	24.5%	33.0%	↑ **8.5%**

Table 2: **Success rate of restart experiment.** In the 5 image-based settings, restart can significantly improve the policy's performance, which proves our detector is of realistic meaning and importance.

Figure 2 visualizes this process on the **square image** and **can image** tasks. The plots illustrate how the detected failure-prone score remains low during normal execution, spikes upon a disturbance to trigger a restart, and normalizes after the policy successfully completes the task post-restart.

5.2.4 Rollout timing accuracy

We conducted a further experiment on the square image task to assess our method's timeliness—its ability to react swiftly to OoD states. In this test, we induced an OoD environmental shift at a predetermined time and tracked the outputs of all methods. As shown in Figure 3, our method immediately responds to the disturbance, with the failure-prone probability rising sharply to a high, stable level, accurately reflecting the ground truth.

Notably, our method also excels at interpreting transient anomalies. At timestep 25, the policy makes a brief, self-corrected error. While baselines flag this initial deviation and remain in a high-probability state (failing to recognize the recovery and risking a premature termination), our method correctly identifies the event's transient nature. This highlights its capacity to differentiate between momentary, recoverable slips and genuine, persistent failures.

5.3 Real-World Experiments

5.3.1 Real-World Environment

We evaluated FailGuard's real-world performance across three tabletop manipulation tasks. In these experiments, our detector triggers a policy restart upon identifying a task failure-prone. The tasks and their disturbance settings can be found in Appendix G.

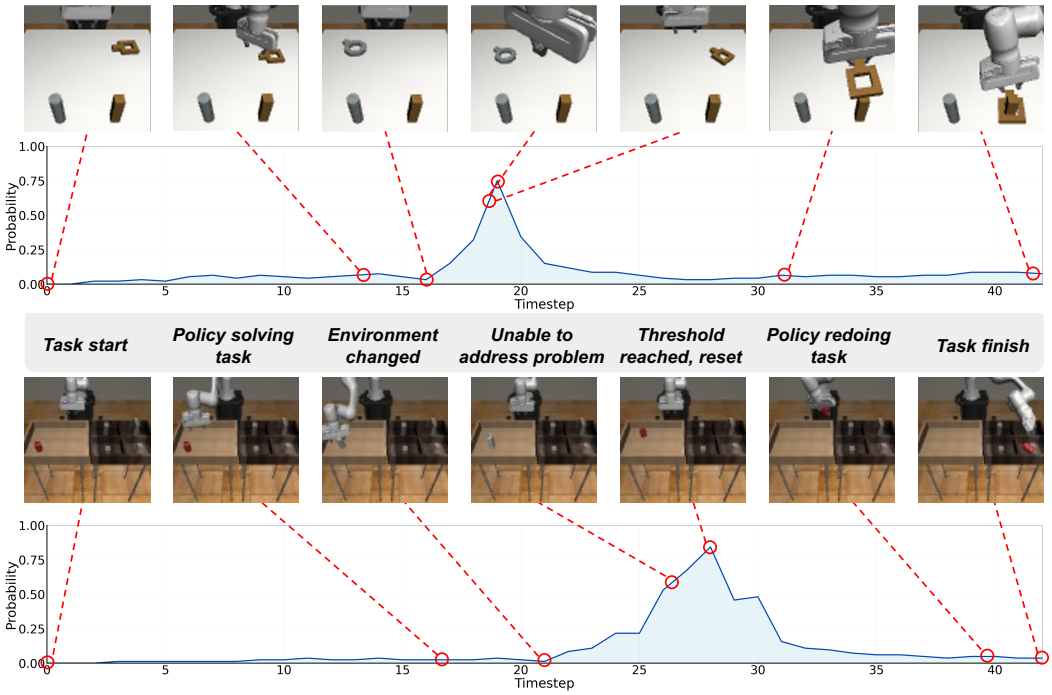

Figure 2: **Visualization of the process of restart experiment.** The context of detector's output when early stop and restart triggered is shown, as well as some key timestep observation image. It is shown that the our protocol is in line with expectations, and can be used in realistic scenario.

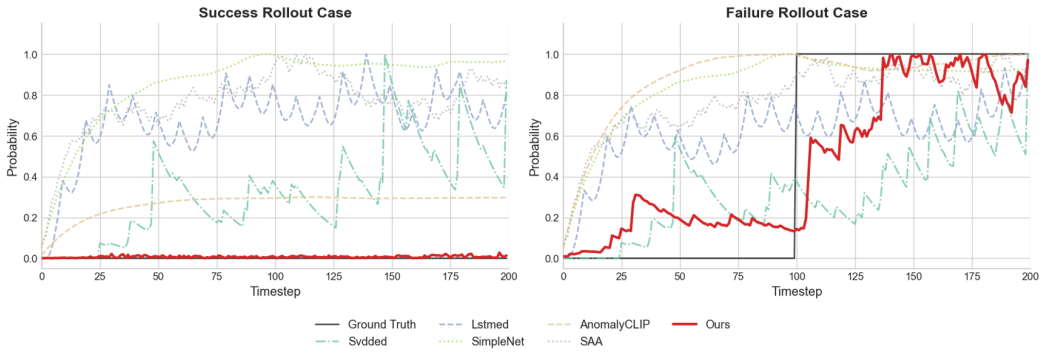

Figure 3: **failure-prone score predictions.** We plot the output failure-prone score over timesteps for both success and failed rollout cases. These curves show how the model's failure detection evolves as the rollout progresses. Our method demonstrates a clear advantage over the other baselines in both accuracy and response time, effectively identifying failure-prone score states faster and more reliably.

As illustrated in figure 4, our method outperforms all baselines in the real-world setting, consistent with the results obtained in simulation.

5.3.2 Restart Experiment

We performed restart experiments on the real robotic arm. As shown in table 3, our method provides substantial improvements to the Diffusion Policy, achieving from 9% to 18% success rate increase.

5.4 Ablation Study

We experimented with training our detector model using multiple different settings, including data collection choice, noise variance scale, and loss function choices. The result is shown in table

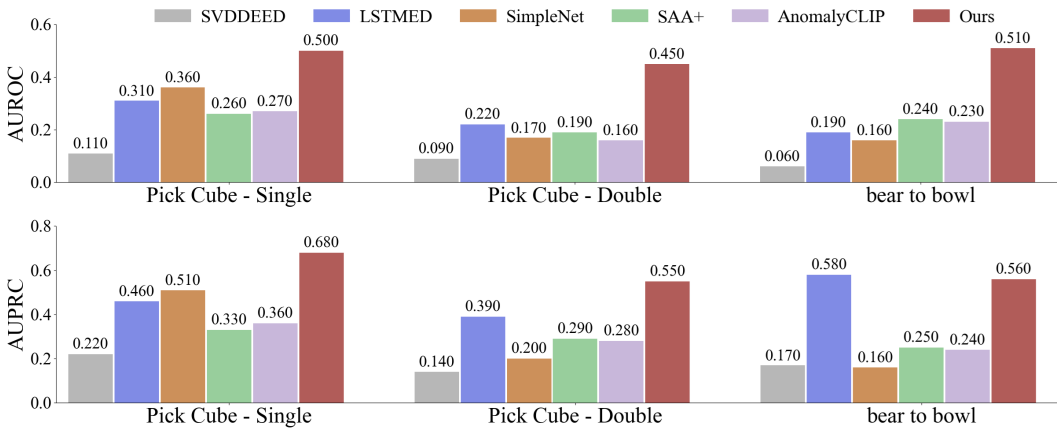

Figure 4: **Performance comparison of our method on real-world robotic tasks.** The experimental results show that our method achieves the best results on most tasks, demonstrating our FailGuard can work well in real world.

Task	Original	With restart	Improvement
Pick Cube – Single	63.6%	72.7%	↑ **9.1%**
Pick Cube – Double	45.5%	63.6%	↑ **18.1%**
Bear to Bowl	18.2%	36.4%	↑ **18.2%**

Table 3: **Success rate of restart experiment in real world.** In all the 3 tasks, restart can improve the policy's performance.

4. According to the ablation study results, We validate the effectiveness of our method and key choices, demonstrating the advantages of our approach.

Metric	fixed large noise	fixed small noise	timestep-wise noise	Uniform Noise	Laplace Noise	all trajectories	without contrastive loss	without BCE loss	Mixed Data (Oracle)	**Ours**
AUROC	0.821	0.803	0.903	0.777	0.831	0.625	0.863	0.803	0.957	**0.996**
AUPRC	0.783	0.877	0.949	0.890	0.896	0.688	0.902	0.890	0.988	**0.999**

Table 4: **Ablation study (transposed) on the square image task.** AUROC and AUPRC across different settings. Our method achieves the best performance.

6 Conclusion

We introduced FailGuard, a method for failure detection in robotic visuomotor policy learning. Instead of collecting diverse failure data, which is impractical and hazardous in real-world scenarios, FailGuard augments success rollouts to simulate failure-prone cases for failure detection training. By leveraging contrastive learning and supervised methods, our approach ensures robust detection performance. Evaluations on the RoboMimic benchmark and Real-world tasks demonstrated FailGuard's effectiveness, outperforming baselines across various tasks and improving task success rates through an early-stop and restart mechanism. These results highlight the method's potential to enhance safety and reliability in robotic manipulation.

We have made every effort to ensure that the results presented in this paper are reproducible. The cxperimental sctup, including training steps, model configurations, andhardware details, is deseribed in detail in the paper. We believe these measures will enable other researchers to reproduce our work and further advancethe feld.

REFERENCES

Abdullah Cihan Ak, Eren Erdal Aksoy, and Sanem Sariel. Learning failure prevention skills for safe robot manipulation. *IEEE Robotics and Automation Letters*, 2023.

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine learning*, 2:343–370, 1988.

Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

Yunkang Cao, Xiaohao Xu, Chen Sun, Yuqi Cheng, Zongwei Du, Liang Gao, and Weiming Shen. Segment any anomaly without training via hybrid prompt regularization. *arXiv preprint arXiv:2305.10724*, 2023.

Noe Casas. Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*, 2017.

SH Cheong, JH Lee, and CH Kim. A new concept of safety affordance map for robots object manipulation. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 565–570. IEEE, 2018.

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*, pp. 02783649241273668, 2023.

Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, pp. 475–489. Springer, 2021.

Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9737–9746, 2022.

Wenfeng Deng, Yuxuan Li, Keke Huang, Dehao Wu, Chunhua Yang, and Weihua Gui. Lstmed: An uneven dynamic process monitoring method based on lstm and autoencoder neural network. *Neural Networks*, 158:30–41, 2023.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pp. 158–168. PMLR, 2022.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1705–1714, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

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*, pp. 1861–1870. PMLR, 2018.

Yunwen Lei, Tianbao Yang, Yiming Ying, and Ding-Xuan Zhou. Generalization analysis for contrastive representation learning. In *International Conference on Machine Learning*, pp. 19200–19227. PMLR, 2023.

Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. Simplenet: A simple network for image anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20402–20411, 2023.

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. *arXiv preprint arXiv:2108.03298*, 2021.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pp. 2817–2826. PMLR, 2017.

Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14318–14328, 2022.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pp. 4393–4402. PMLR, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. ISBN 978-1-108-41519-4. doi: 10.1017/9781108231596. URL https://www.cambridge.org/core/books/highdimensional-probability/797C466DA29743D2C8213493BD2D2102.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Reconstruction by inpainting for visual anomaly detection. *Pattern Recognition*, 112:107706, 2021.

Qihang Zhou, Guansong Pang, Yu Tian, Shibo He, and Jiming Chen. Anomalyclip: Object-agnostic prompt learning for zero-shot anomaly detection. *arXiv preprint arXiv:2310.18961*, 2023.

A Proof of Theorem 1

Here, we provide the detailed derivation for the generalization bound presented in Section 4.4.

Our problem is equivalent to a binary classification task with noisy 0–1 labels, where the label noise rate is η. By invoking the relationship between the risk on a noisy dataset and the risk on a clean dataset from (Angluin & Laird, 1988), we can write:

$$R_{\widetilde{D}}(f) = (1-\eta)R_D(f) + \eta(1 - R_D(f))$$

Rearranging this equation gives the risk on the true data distribution, $R_D(f)$, in terms of the noisy risk, $R_{\widetilde{D}}(f)$:

$$R_D(f) = \frac{R_{\widetilde{D}}(f) - \eta}{1-2\eta}, \quad (\text{for } \eta < \frac{1}{2}) \tag{2}$$

From this relation, one sees that a larger label-noise rate η incurs a higher true risk, which aligns with intuition.

Next, we establish the generalization bound for applying contrastive learning methods on our noisy dataset $\widetilde{D}$. We assume our loss function l is G-Lipschitz continuous w.r.t. $\|\cdot\|_\infty$ and $\|f(x)\|_2 \leq R$. Then, according to Theorem 4.9 in (Lei et al., 2023), we have the following bound. For $\delta \in (0,1)$, with probability at least $1-\delta$ for all $f \in \mathcal{F}$:

$$R_{\widetilde{D}}(f) \leq \hat{R}_{\widetilde{D}}(f) + O\left(\frac{GR\mathcal{B}\log^2(nRk)}{n\sqrt{k}} + \mathcal{B}\sqrt{\frac{\log(1/\delta)}{n}}\right) \tag{3}$$

where n is the number of positive samples, k is the number of negative examples, and $\mathcal{B}$ denotes a global upper bound on the "contrastive score"—that is, the difference in similarity between a positive sample and a negative sample.

By substituting the generalization bound from Equation 3 into the risk relationship from Equation 2, we derive the final bound on the generalization error of our method:

$$\begin{aligned} R_D(f) &= \frac{R_{\widetilde{D}}(f) - \eta}{1-2\eta} \\ &\leq \frac{1}{1-2\eta}\left(\hat{R}_{\widetilde{D}}(f) + O\left(\frac{GR\mathcal{B}\log^2(nRk)}{n\sqrt{k}} + \mathcal{B}\sqrt{\frac{\log(1/\delta)}{n}}\right) - \eta\right) \end{aligned}$$

This is the result presented in Theorem 1 in the main text.

B Detailed Baseline Introduction

Among the reconstruction-based approaches, **SVDDED** (Ruff et al., 2018) is a Deep One-Class autoencoder that identifies outliers by measuring the L_2 distance of an input's embedding to a learned center; however, this method relies on single-frame decisions and ignores temporal information. In contrast, **LSTMED** (Deng et al., 2023) is an LSTM-based encoder-decoder method that leverages temporal context, using the reconstruction error (the L_2 distance between input and output) as the anomaly score. Another baseline, **SimpleNet** (Liu et al., 2023), is a classifier trained with Gaussian noise augmentation and contrastive learning that, unlike our method, uses fixed-variance noise and operates on single frames. For the zero-shot methods, **AnomalyClip** (Zhou et al., 2023) is a CLIP-based approach using text prompts to detect anomalies in single images, which is effective for attribute changes but cannot process temporal sequences. Similarly, **SAA+** (Cao et al., 2023) is a zero-shot method based on Segment Anything (SAM) that uses prompts to segment anomalous regions, exhibiting principles and performance similar to AnomalyClip.

C Implementation Details Of Simulation Environment

We use a transformer-based network as our detector. First, we apply a feature mapping layer to project the input detection features into hidden layer features with a specified size of $hidden\_dim$. We then add position embedding to the hidden layer features, enabling the transformer to leverage the temporal information from the rollout trajectory for decision-making. This is a key distinction from many methods that rely solely on single-frame images. The hidden layer features with position

embeddings are then fed into a decoder-only transformer, which outputs the embedding e. Finally, fully-connected output mapping layer is applied on embedding e to generate the anomaly logits $\hat{y}$. The embedding e and logit $\hat{y}$ are used to compute loss later.

In our simulation experiments, we set $hidden\_dim = 512$. The decoder-only transformer has 4 layers, each with 8 attention heads. For the optimizer, we use the Adam optimizer with a learning rate of $1e^{-4}$ and a weight decay of 0.01.

D Implementation Details of Real-World Environment

We use a transformer-based network as our detector similar to the one introduced in Sec. C. However, due to the complexity of real-world tasks, we adopt Diffusion Policy (Chi et al., 2023) as the agent's policy to control the robotic arm. For the Diffusion Policy, we pass the MLP-mapped encoded observation features to the detector to determine whether the current state is out-of-distribution. Here we set $hidden\_dim = 512$. The decoder-only transformer has 6 layers, each with 8 attention heads.

E Visualization

To better understand the reasons behind our method's effectiveness, we conducted a series of visualizations.

E.1 Embedding Visualization

we applied t-SNE to embeddings e from a successful and a failed rollout in the square image task. The visualization in Figure 5 reveals two distinct clusters for success (green) and failure (red) states. This high degree of separability indicates our model learns effective representations to differentiate between nominal and anomalous trajectories.

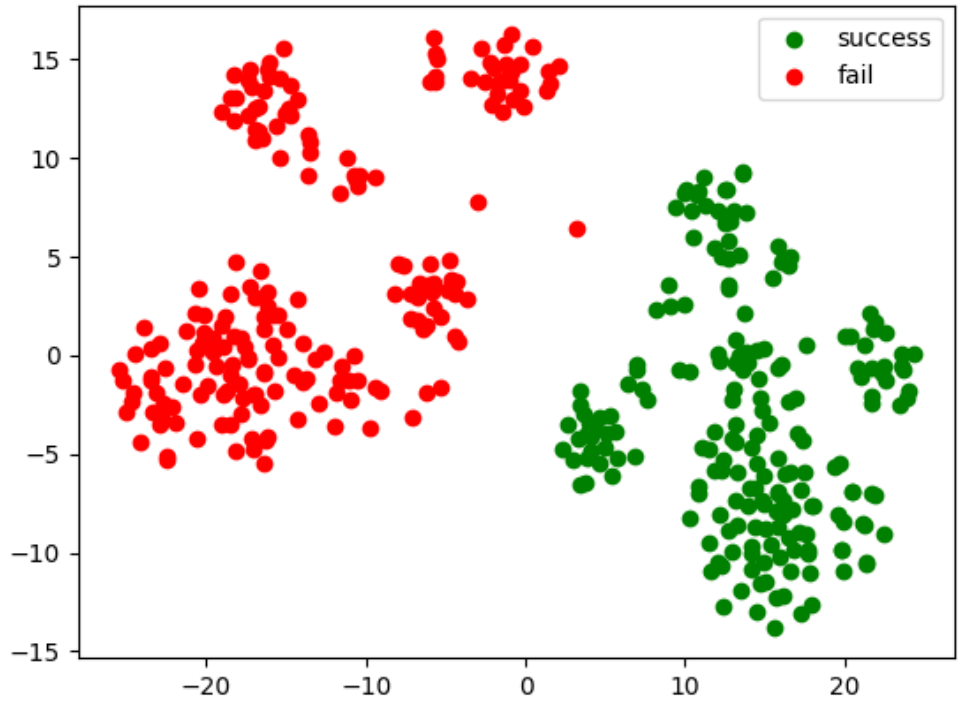

Figure 5: **T-SNE visualization of learned embeddings(representations).** We extract the embeddings e) output by the model at each timestep of a trajectory and apply the t-SNE method to reduce the embeddings to a 2D plane. The red and green points represent embeddings from trajectory of from failed and successful cases, respectively. The results show that our method effectively distinguishes between the features of failed and successful trajectories, with clear separation in the 2D space.

E.2 SALIENCY MAP

We use Grad-CAM to visualize which observational features drive our detector's anomaly score. Figure 6 shows these saliency maps for three tasks—**lift image**, **can image**, and **square image**—under three conditions: successful completion, disturbance, and an OoD environment.

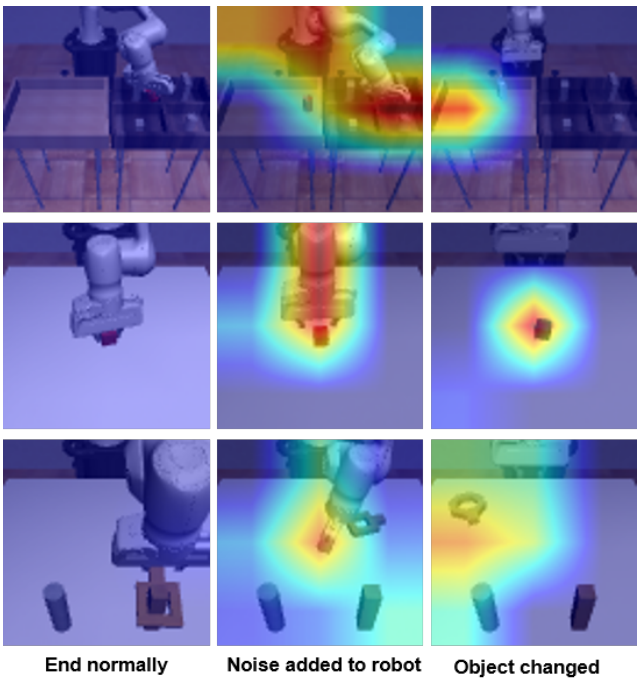

Figure 6: **Saliency map of the detector's output relative to the input observation image.** The "End normally" column displays the saliency map when the task is completed with no failure. The "Noise added to robot" column mainly focuses on the robot's posture and the absence of an object being grasped by the gripper. For the "Object changed" column, the top image marks the object's original position, the middle image highlights the cube's texture change, and the bottom image shows the nut's shape changes from square to circle.

The visualizations reveal that our detector correctly localizes the source of failure. While attention is diffuse in successful runs, it sharply focuses on an abnormal robot posture during a disturbance or a modified target object in an OoD state. This demonstrates that our method effectively identifies the key out-of-distribution components in the input observation.

E.3 NOISE TRAJECTORY VISUALIZATION

We also visualized the a rollout and its augmented trajectories. We generated one hundred Gaussian-noise-augmented trajectories and reduced the dimensionality of their feature data to two dimensions using t-SNE. In Figure 7, the original trajectories are represented in blue, while the noise-augmented points are shown in red.

From the visualization, it is evident that the noise-augmented trajectories form a "protective shell" around the original trajectories, akin to the relationship between the copper core and the insulation layer of a wire. The original trajectory data, resembling the core, represents the normal states, while the noise-augmented trajectories, analogous to the insulation layer, encapsulate it. This is indeed referred to as the Gaussian annulus theorem in high-dimensional statisticsVershynin. This structure underscores the effectiveness of our noise augmentation method, as it provides a robust and comprehensive coverage of the data space, enabling the detector to generalize better to potential anomalies

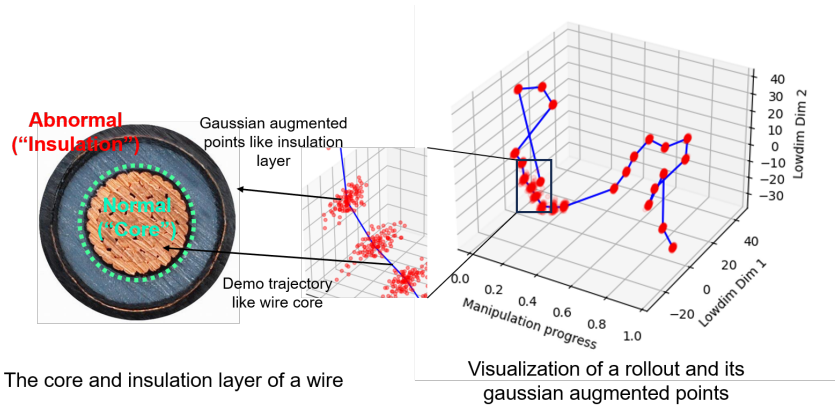

Figure 7: **Visualization of the original and Gaussian-noise-augmented trajectories.** The t-SNE algorithm was used to reduce the dimensionality of trajectory feature data to two dimensions. The original trajectories are plotted in blue, while the noise-augmented trajectory points are shown in red. The augmented trajectories form a "insulation layer", encapsulating the original trajectories "core" within.

F Robomimic Simulation Experiment Settings

Environment name	Property	In domain setting	Out of domain setting
Lift	specular	0.4	0.7
	shininess	0.1	0.2
	texture	red wood	blue wood
	initial x range	[-0.03, 0.03]	[-0.05, 0.03]
	initial y range	[-0.03, 0.03]	[-0.01, 0.03]
can	object	can	milk carton
Square	nut shape	square	round
	initial x range	[-0.115, -0.11]	[-0.125, -0.12]
Tool Hang	frame initial position	(-0.08, 0.0)	(-0.10, 0.0)
	hook initial position	(-0.04, -0.24)	(-0.02, -0.22)
	hook initial rotation	$\frac{1}{3}\pi$	$-\frac{5}{18}\pi$
	wrench initial position	(0.04, -0.2)	(0.04, -0.18)
	wrench initial rotation	$-\frac{11}{18}\pi$	$\frac{1}{3}\pi$
Transport	cube texture	red wood	blue wood
	left box initial X-Axis position center	0.2	0.21
	top right box initial X-Axis position	0.2	0.21
	lower right box initial X-Axis position	-0.2	-0.21
	hammer initial rotation	$\frac{1}{2}\pi$	$-\frac{1}{2}\pi$

Table 5: The detailed OOD parameters compared to in domain settings for all tasks in our experimental environments. Only the different properties between the two are listed.

G Real-World Experiment Task and Disturbance Setting

- **Pick Cube - Single**. The robot picks a single cube and places it in a plate. OoD conditions include: 1) human perturbation of the cube during grasp, 2) the cube's initial absence, or 3) human removal of the cube mid-air.

- **Pick Cube - Double**. The robot must pick a cube of a specified color from two options. OoD conditions include: 1) the absence of the target-colored cube, or 2) shifted initial positions.
- **Bear to Bowl**. The robot picks a toy bear and places it in a bowl, a task where failure can occur if the bowl tips. The primary OoD condition is the bear's initial absence.

In all three tasks, whenever an out-of-distribution condition is detected (or the task fails), a restart should be triggered , resetting the robot to its initial position and allowing the policy to attempt the task again.

H LLM USAGE

Large Language Models (LLMs) were employed to support the writing and refinement of this manuscript. In particular, we used an LLM to enhance the clarity, readability, and overall flow of the text, including tasks such as sentence rephrasing, grammar checking, and language polishing.

The LLM was not involved in generating research ideas, designing the methodology, or conducting experiments. All concepts, approaches, and analyses were independently developed and carried out by the authors. The LLM's role was limited exclusively to improving the linguistic presentation of the paper, without contributing to its scientific content or data analysis.

The authors retain full responsibility for the manuscript, including any portions of text refined with the assistance of the LLM. We also ensured that the use of the LLM complied with ethical standards and did not result in plagiarism or scientific misconduct.