Cost-aware Discovery of Contextual Failures using Bayesian Active Learning

Anjali Parashar **Joseph Zhang** **Yingke Li** **Chuchu Fan**
Laboratory for Information & Decision Systems (LIDS)
Massachusetts Institute of Technology, United States
{anjalip, jzha, yingkeli, chuchu}@mit.edu

Abstract: Ensuring the robustness of robotic systems is crucial for their deployment in safety-critical domains. Failure discovery, or falsification, is a widely used approach for evaluating robustness, with recent advancements focusing on improving sample efficiency and generalization through probabilistic sampling techniques and learning-theoretic approaches. However, existing methods rely on explicitly defined analytical cost functions to characterize failures, often overlooking the underlying causes and diversity of discovered failure scenarios. In this work, we propose a novel failure discovery framework that integrates contextual reasoning in the falsification process, specifically tailored for high evaluation-cost applications. Our method incorporates expert-in-the-loop feedback to construct a probabilistic surrogate model of failures using Bayesian inference. This model is iteratively refined and leveraged to guide an active learning strategy that prioritizes the discovery of diverse failure cases. We empirically validate our approach across a range of tasks for high-cost contextual falsification in robotic manipulation and autonomous driving. The project website is at https://mit-realm.github.io/contextualfailures-website/.

Keywords: Failure discovery, Testing, Contextual failures

1 Introduction

Failure discovery, or falsification, of complex systems is the process of demonstrating that the system fails to behave as expected [1, 2, 3, 4, 5]. It is a critical task to ensure the safety and reliability of autonomous systems [6], but often labor-intensive and time-consuming due to system complexity and the high cost associated with evaluations [7, 8]. Existing failure discovery methods based on probabilistic and optimization fundamentals require at least one of the following: an accurate model of the system (model-based methods) [9, 10, 11, 12, 5], a well-defined cost function to guide optimization (optimization-based methods) [13, 14, 15, 16], or the ability to run the system extensively (sampling-based methods) [9, 10, 11]. However, these requirements can be challenging to meet in many applications, making it difficult to uncover failures in real-world autonomous systems.

Another key drawback of traditional falsification techniques is the lack of direct expert engagement in failure discovery, limiting their capability to replicate and generate similar types of failures. This is especially relevant for user reported edge cases that need further investigation and post-failure analysis [17, 18]. Recently, some works have proposed generative models to learn failures from historical data [19, 20, 18], however, these methods are data-intensive by default and face the risk of under-representing infrequent failure cases [21].

In this work, we propose a cost-aware failure discovery methodology that treats both the system and its evaluation process as black boxes, avoiding assumptions about their internal structure or cost functions. Our key contribution is the notion of *contextual failures*—failures that depend on user goals, task constraints, and system intricacies. Unlike traditional failures, these are difficult to define analytically due to multiple plausible causes and limited knowledge of where such failures might arise. Instead, they can be recognized through expert-identified incidents of interest.

Practitioners often cannot formally specify these failures but find them informative and seek to uncover diverse scenarios with similar context—e.g., object detection failures in self-driving under a variety of scene influencing factors such as weather, lighting, or speed of vehicle. Evidently, contextual failures lack clear cost functions, making them difficult to evaluate quantitatively. To address this, we propose a sequential scenario exploration framework that leverages expert feedback to guide the discovery of diverse, context-relevant failures. To reduce reliance on costly human input (leading to high cost of expert), our method also provides the capability to incorporate LLMs as proxy experts for evaluating failures.

Fig. 1 shows an overview of our proposed approach, which consists of three main parts, (a) an active learning strategy that prioritizes the generation of diverse failure scenarios for expert evaluation, (b) expert evaluation to attribute each observed failure to multiple failure modes, and (c) surrogate models trained in parallel to map scenario parameters to failure levels associated with each failure mode. Together, the framework progressively uncovers diverse and meaningful failure scenarios.

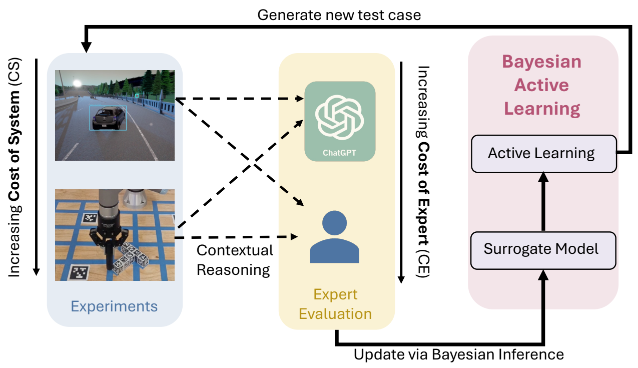


Figure 1: Overview of our framework: We propose a cost-aware methodology for failure discovery of contextual failures using expert evaluation (Section 4.2), Bayesian inference to fit a surrogate model (Section 4.3), and active learning for generating diverse failure scenarios (Section 4.1).

We demonstrate the key strengths of our methodology through two stages of experimental validation. First, we apply it to a single-mode failure discovery task with a known cost function (Section 5.1.1), to motivate the role of coverage in failure discovery, introduced in Section 4.1. We then validate it on three contextual failure tasks in manipulation and self-driving domains, incorporating realistic system constraints and both human and LLM-based evaluations (Sections 5.1.2–5.2.2). Our method emphasizes coverage in parameter and metric spaces, discovering up to 6× more failures than random sampling. It also supports specification of failure severity levels, uncovering failures across the entire severity range—unlike random sampling, which fails to capture rare severe failures (0% in some settings). Finally, in Section 5.2.2, we show the scalability of our active learning strategy on high-dimensional scenarios. We also provide extensive hyperparameter analysis for all our experiments, and ablation studies in the appendix.

In summary, the key contributions of our work include:

1. **Cost-aware evaluation of contextual failures:** We provide a data-efficient strategy for discovering contextual failures that describe nuanced failures of well-engineered systems that we commonly encounter, that cannot be analytically expressed using cost functions.
2. **Expert-driven failure mode attribution:** We leverage expert feedback to attribute each scenario to pre-defined plausible failure modes associated to each type of contextual failure observed. Our framework also supports online addition of failure modes.
3. **Generating diverse failure cases:** We provide a framework for generating diverse failure scenarios, by defining coverage metrics in parameter and metric space.

2 Related works

We provide a brief discussion of a few leading techniques for failure discovery and analysis here and refer the readers to Appendix A for a detailed literature review.

Traditional methods. These approaches frame failure discovery as a cost-guided search over a known dynamic system, either using sampling-based methods (e.g., MCMC, Metropolis-Hastings [22, 12, 10, 23, 24, 22, 25]) or optimization-based methods [13, 14, 15]. These methods cannot be applied in black-box, high-cost settings, or context-relevant falsification. Our method overcomes these limitations using data-efficient, black-box evaluations and expert feedback.

Learning-based methods. These methods use historical data to learn failure or dynamics models [19, 20], often through generative architectures [16, 19]. However, they require large datasets and overlook the imbalance in failure mode frequency. Our approach is data-efficient, models epistemic uncertainty via Bayesian inference [9, 26], and learns separate surrogates per failure mode.

Learning from expert feedback: Prior work incorporates qualitative feedback to fine-tune reward models (e.g., RLHF), typically assuming data-rich settings [27, 28]. We adapt the expert-in-the-loop idea (Fig. 1) for data-efficient failure discovery, where RL techniques cannot be used due to high cost of evaluation. Our work is complementary to [17, 18, 29, 30] that have proposed LLMs for failure diagnosis for runtime monitoring. We assume access to an expert—human or LLM—with the key focus on reproducing and exploring similar failure scenarios that are of interest to the user.

3 Problem Setup

We consider a dynamic system $\dot{x} = f(x, \pi(o, z))$ with state $x \in \mathcal{S}$, with system observation $o \in \mathcal{O}$ and scenario parameters $z \in \mathcal{Z}$, which takes action $a \in \mathcal{A}$ based on a certainy policy $\pi : \mathcal{O} \times \mathcal{Z} \to \mathcal{A}$. We leverage domain knowledge to design $\mathcal{Z}$ to represent scenario parameters that can be modified to generate unique testing conditions. Consider the falsification of an object detection module in a self-driving simulator, with a non-ego vehicle and pedestrian. Here, each scenario z represents braking distances for ego and lead car, and brightness of the scene (Section 5.2.1). These parameters are chosen from prior knowledge as they affect the accuracy of object detection.

Contextual failures with multiple modes. Contextual failures are evaluated by an expert based on a sequence of observations $\boldsymbol{o_z} = (o)_{t=1}^{T} \in \tilde{\mathcal{O}}$ of the system within time horizon T, for a given scenario parameterized by z. The expert can be either a human or LLM. We represent expert assessment as a mapping $g : \tilde{\mathcal{O}} \to \mathbb{R}$. Contextual failures are often attributed to multiple sources, where each source corresponds to a specific *failure mode* that can be independently evaluated by the expert. The cumulative expert evaluation for M failure modes is $\boldsymbol{g}(\boldsymbol{o_z}) = [g_1(\boldsymbol{o_z}), \ldots, g_M(\boldsymbol{o_z})]$, where $g_m(\boldsymbol{o_z}) \in \mathbb{R}$ is the expert evaluation for the m^{th} failure mode.

Severity of failures. We define the severity of a specific failure mode as a measure of the degree of damage caused by it. Qualitative feedback provided by the expert $\boldsymbol{g}$ is often sparse (e.g., binary attribution by an LLM to classify if an object detection failure in a self-driving scene can be attributed to low brightness), but a user may wish to refine this assessment with their own definition of severity. Thus, we include a post-processing step $h : \mathbb{R} \to [0, 1]$, that converts expert evaluation $g_m(\boldsymbol{o_z})$ into a severity score $\gamma_m := h \circ g_m$, such that for a given z, γ_m is monotonic in severity of failure. We discuss this in detail in Section 4.2. Note that we distinguish between h and g purely for the sake of clarity. Our method directly models and uses γ_m.

Discovering failures with varying levels of severity: We use a severity parameter δ_m to define the minimum level of severity of failure we wish to observe. It defines a target set of scenario parameters, and the goal of failure discovery is to sample from this target set. Specifically, falsification of the m^{th} failure mode is equivalent to sampling scenario parameters z from the set $\Omega_m = \{z | \gamma_m(z) > \delta_m, z \in \mathcal{Z}\}$, for $m = 1, \ldots, M$, assuming M failure modes. Therefore, the overall target set Ω can be defined as:

$$\Omega = \{z | \boldsymbol{\gamma}(z) \succ \boldsymbol{\delta}\}, \tag{1}$$

where $\boldsymbol{\gamma}(z) \succ \boldsymbol{\delta} := \gamma_m(z) > \delta_m, m = 1, \ldots, M$.

In this work, we pay special attention to the cost of sampling scenario parameters from (1).Specifically, we consider applications where either the dynamic system f is costly to evaluate, due to resource, time constraints, or safety hazards, and/or, there is a substantial cost associated with expert evaluation g, i.e, at least one of the functions f or g are costly to evaluate. The cost of evaluation imposes an upper limit on the overall experimental trial budget N.

Furthermore, we aim to sample diverse failure candidates from (1). Diversity of a set of sampled candidates $Z = \{z_j\}_{j=1}^N$ in this context is defined using a coverage metric $C : \mathcal{Z}^N \times [0,1]^{M \times N} \to \mathbb{R}$, such that higher value of C denotes better diversity of sampled scenario parameters. In Section 4.1 we provide a more specific definition of the coverage metric C, and an active learning strategy to maximize coverage.

4 Methodology

The requirement of sample efficiency motivates the use of a sequential scenario design strategy [31]. Our approach is an iterative process, summarized in Algorithm 1.

Algorithm 1 Bayesian Active learning for contextual failure discovery

```
1: Input: Scenario space $\mathcal{Z}$, prior $(p(\boldsymbol{w}_m))_{m=1}^M$, initial dataset $\mathcal{D}^0$ (optional)
2: Initialize surrogate models $\boldsymbol{q} = (q_m^*)_{m=1}^M$
3: for $k = 0$ to $N-1$ do
4:     Select scenario $z_{k+1} = \arg\max_{z \in \mathcal{Z}} \alpha(z|\mathcal{D}^k)$ for $\alpha$ in Eq (3) (Section 4.1),
5:     Simulate scenario $z_{k+1}$ and obtain severity adjusted expert evaluation $\boldsymbol{\gamma}(z_{k+1})$ (Section 4.2),
6:     Update dataset: $\mathcal{D}_{k+1} = \mathcal{D}_k \cup \{(z_{k+1}, \boldsymbol{\gamma}(z_{k+1}))\}$
7:     Update surrogate models $\boldsymbol{q} = (q_m^*)_{m=1}^M$, where $q_m^* : \mathcal{Z} \to [0,1]$ models $\gamma_m$, for each failure
       mode, using $\mathcal{D}_{k+1} = (z_j, \boldsymbol{\gamma}(z_j))_{j=1}^{k+1}$ (Section 4.3)
8: end for
```

We adopt a Bayesian framework and define M independent surrogate models $(q_m^*)_{m=1}^M$ required by our active learning strategy, each parameterized by latent variables $\boldsymbol{w}_m$ with prior $p(\boldsymbol{w}_m)$ reflecting expert beliefs. Given data $\mathcal{D} = (\boldsymbol{z}, \boldsymbol{\gamma}(\boldsymbol{z}))$, each surrogate is defined as $q_m^* = p(\boldsymbol{w}_m|\mathcal{D})$. We propose an active learning strategy to guide data collection and adjust expert evaluations based on severity.

Modeling of prior: We can either use expert belief, or, a dataset of historical failures ($\mathcal{D}^0$) to model a prior $p(\boldsymbol{w}_m)$ for each failure mode. In our experiments, we use Gaussian Processes (GP) [32] as surrogate models, and learn a prior model for the GP using 5 uniformly sampled scenarios. We also explore the effect of size of dataset $\mathcal{D}^0$ in the training of prior in Experiment 4B, Appendix C.5.

4.1 Active learning for sampling from target set

The active learning step samples the point z^* that optimizes the expected coverage improvement of desired failure severity over scenarios observed thus far. Specifically, our acquisition strategy must prioritize points that satisfy set membership with Ω described in (1), while covering a broad range of severity levels. Using $(q_m^*)_{m=1}^M$, we can estimate the probability of a point z satisfying the set membership for Ω by defining an indicator variable $I(z) = \mathbb{1}(z \in \Omega)$. Using q_m^*, we can write $\mathbb{1}(z \in \Omega) = \prod_{m=1}^M \mathbb{1}(q_m^*(z) > \delta_m)$, so that $p(I(z) = 1|\mathcal{D})$ is given by:

$$p(I(z) = 1|\mathcal{D}) = \prod_{m=1}^M \mathbb{E}[\mathbb{1}(q_m^*(z) > \delta_m)] \tag{2}$$

To ensure diverse and sample-efficient failure discovery, it is essential to discourage revisiting previously explored areas. We address this using the concept of *coverage neighborhood* adopted from [33]. We define two related coverage metrics: parameter space coverage (C_p) and metric space coverage (C_m). C_p encourages sampling diverse scenarios in $\mathcal{Z}$, measured by Euclidean distance, $\|z - z'\|_2$. Similarly, C_m promotes diversity in failure severity, quantified by Euclidean distance between observed failure responses in metric space, $\|\boldsymbol{\gamma}(z) - \boldsymbol{\gamma}(z')\|_2$.

Coverage neighborhood: For an appropriate distance function $d_p : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}^+$, and pre-specified $r_p \in \mathbb{R}^+$, coverage neighborhood of a scenario parameter $z \in \mathcal{Z}$ in parameter space is defined as

$\mathbb{N}_p(z) = \{z' : z' \in \mathcal{Z}, d_p(z, z') < r_p\}$. Similarly, we use a separate distance function in the metric space, $d_m : \mathbb{R}^{m+} \times \mathbb{R}^{m+} \to \mathbb{R}^l$, and radius of coverage $r_m \in \mathbb{R}^+$ to define a coverage neighborhood in metric space as $\mathbb{N}_m(z) = \{z' : z' \in \mathcal{Z}, d_m(\boldsymbol{\gamma}(z), \boldsymbol{\gamma}(z')) < r_m\}$. In this work, we use Euclidean distances, for which, the expression $\mathbb{N}_p(z)$ and $\mathbb{N}_m(z)$ is equivalent to a ball of radius r_p and r_m in parameter and metric space respectively. This can be extended to define coverage neighborhoods for a set of points $\boldsymbol{z}$ observed so far as: $\mathbb{N}_p(\boldsymbol{z}) = \cup_{z \in \boldsymbol{z}} \mathbb{N}_p(z)$ and $\mathbb{N}_m(\boldsymbol{z}) = \cup_{z \in \boldsymbol{z}} \mathbb{N}_m(z)$. The coverage metrics in parameter and metric space can be formally stated as $C_p(\mathcal{D}) = \text{Vol}(\mathbb{N}_p(\boldsymbol{z}) \cap \Omega)$ and $C_m(\mathcal{D}) = \text{Vol}(\mathbb{N}_m(\boldsymbol{z}) \cap \Omega)$. The relative importance of these two coverage metrics is task-specific. Therefore, we propose a general utility function C, that incorporates both C_m and C_p as $C(\mathcal{D}) = \lambda C_p(\mathcal{D}) + (1-\lambda) C_m(\mathcal{D})$. Here, $\lambda \in [0, 1]$ is used as a hyperparameter to specify trade-off between diversity in parameter and metric space respectively. For the k^{th} iteration of active learning, the objective of data acquisition can be written as: $z_{k+1} = \arg\max_{z \in \mathcal{Z}} \alpha(z|\mathcal{D}^k)$, where

$$\alpha(z|\mathcal{D}^k) = \mathbb{E}_{\boldsymbol{\gamma}(z)}[C\big(\mathcal{D}^k \cup (z, \boldsymbol{\gamma}(z))\big) - C(\mathcal{D}^k)], \tag{3}$$

and is referred to as Expected Coverage Improvement (ECI) [33]. In practice, estimation of coverage in metric and parameter space is not analytically tractable, and is done using Monte Carlo integration. See Appendix B.1 for implementation details of $\alpha(z|\mathcal{D}^k)$.

4.2 Expert evaluation strategy

The selected scenario $z_{k+1} = \alpha(z|\mathcal{D}^k)$ is used to collect observations $\boldsymbol{o}_{z_{k+1}}$ and expert evaluations $\boldsymbol{\gamma}(z_{k+1})$. Here we show an example of an evaluation strategy to construct a function γ_m. In the absence of formal definitions, severity can be estimated by either its impact on system safety or the duration spent in the failure mode. For non-terminal, state-based failures, an expert can provide binary labels $\boldsymbol{b_t} = [b_t^1, \dots, b_t^M] \in \{0, 1\}^M$ at each timestep t, where $b_t^m = 1$ indicates a failure for mode m. The cost is then computed as $\gamma_m(z_k) = \frac{1}{T} \sum t = 1^T b_t^m$. However, some contextual failures cause early termination, making duration-based metrics insufficient. We generalize this evaluation to such conditions as:

$$\gamma_m(z) = \begin{cases} 1 & \text{if early stopping due to observed failure,} \\ \frac{1}{T} \sum_{t=1}^{T} b_t^m & \text{otherwise.} \end{cases} \tag{4}$$

Early stopping is defined as actual rollout $T_A < T_F$, where T_F is the max horizon. We apply this evaluation in Sections 5.1.2 and 5.2.1 using human and LLM feedback, respectively. Appendix C.5 discusses alternate definition of severity used in the AEB case study (Section 5.2.2).

4.3 Bayesian Inference for learning surrogate model

At the k^{th} round, collected data $\mathcal{D}_m^{k+1}$ is used to define a likelihood model over latent parameters $\boldsymbol{w}_m$ as $p(\mathcal{D}_m^{k+1}|\boldsymbol{w}_m)$. The posterior is then given by Bayes' rule: $p(\boldsymbol{w}_m|\mathcal{D}_m^{k+1}) \propto p(\mathcal{D}_m^{k+1}|\boldsymbol{w}_m)p(\boldsymbol{w}_m)$, We approximate the posterior $(p(\boldsymbol{w}_m|\mathcal{D}_m^{k+1})$ using a parametric family of models, specifically GPs in our experiments. Assuming exchangability of data, the likelihood factorizes as $p(\mathcal{D}_m^{k+1}|\boldsymbol{w}_m) = \prod_{i=1}^{k+1} p(\mathcal{D}_m^i|\boldsymbol{w}_m)$. Each round constructs M independent likelihood models $p(\mathcal{D}_m^{k+1}|\boldsymbol{w}_m)$. With limited evaluations, training M models is computationally efficient and parallelizable.

5 Simulations & Experiments

Our experimental analysis considers the following performance criteria: **(C1)** Failure discovery across varying severity ranges , **(C2)** Performance of our method across various hyperparameter settings and **(C3)** Discovering failure modes with varying frequencies of occurrence. We first demonstrate our methodology on a low-fidelity simulation of a manipulation task (Push-T) involving a single failure mode and a known cost function (Section 5.1.1). We then validate it across three case studies: (1) Push-T on a UR3E robot with human evaluation (Section 5.1.2), (2) perception failures in a self-driving task in CARLA (Section 5.2.1), and (3) a Simulink-based Autonomous Emergency Braking system (Section 5.2.2), both with LLM evaluation. We conduct ablation studies on: (1)

hyperparameter sensitivity ((r_p, r_m, δ_m)), (2) surrogate model prediction accuracy (Expt. 1B/1C, Appendix C.2), (3) adaptive inclusion of new failure modes (Expt. 2B, Appendix C.3), (4) customizable evaluation strategies (Expt. 3B, Appendix C.4), (5) role of prior knowledge (Expt. 4B, Appendix C.5), and (6) performance vs. evaluation budget (Appendix C.6).

Baselines and metrics: We compare our active learning strategy against a Random walk baseline across all experiments since it is popularly used in autonomous system testing [8], and additionally against Upper Confidence Bound (UCB) [34, 35] for the Push-T (Sim) task, where a single failure mode allows for standard UCB formulation: $\alpha_{\text{UCB}}(z) = \mathbb{E}(q_m^*(z)) + \beta_m \sqrt{\text{Var}(q_m^*(z))}$. We evaluate UCB for $\beta_1 = 0.1, 0.5, 1.0$ (denoted UCB-1/2/3). All scenario parameters are normalized to $[0, 1]$ for consistency. We assess performance using three metrics: **Positive Samples** (P.S.), measures constraint satisfaction [33]; **Coverage-I** (Cov-I) and **Coverage-II** (Cov-II), for diversity in parameter and metric space, estimated using $C_p(\mathcal{D})$ and $C_m(\mathcal{D})$. See Appendix C.1 for details.

5.1 Failure discovery for Diffusion policy on a Manipulation task (Push-T)

Problem Setup: We consider a T-block pushing task using a circular end-effector controlled by a Diffusion policy, trained via a pymunk simulator [36, 37]. The system state is $x_t = [x_E^t, y_E^t, x_T^t, y_T^t, \theta_T^t]$, where (x_E, y_E) and (x_T, y_T, θ_T) denote the end-effector and T-block pose. Scenarios are parameterized by the T-block's initial position $z = (x_T^0, y_T^0)$, with $\theta_T^0 = 45°$.

5.1.1 Failure discovery in simulation (Push-T simulation)

We perform failure discovery in simulation for a known (but non differentiable) cost function $\gamma_1(z) = 1 - \max_{t=[1,\ldots,K]} r_t(s_t)$, where $r_t \in [0, 1]$ refers to policy reward, for $T = 700$.

Experiment 1A (*Failure coverage analysis*): We perform failure coverage analysis for our method with $\lambda = 0, 0.5, 1, r_p, r_m = 0.01, 0.05, 0.1$ (ECI-1/2/3) and $\delta_1 = 0.9, 0.3$ and compare against UCB and Random walk for $N = 50$ evaluations. Key results are reported in Table 1 as mean values for 10 seeds, (see Table 4 for more details). Fig. 2 shows scenarios sampled by different methods against the true cost contour.

Table 1: Performance metrics by method under different values of δ_1 for $M = 1$ (Push-T simulation). λ reported in brackets for ECI

Method	$\delta_1 = 0.9$ (high severity failure discovery)			$\delta_1 = 0.3$ (low severity failure discovery)		
	Positive Samples	**Cov-I**	**Cov-II**	**Positive Samples**	**Cov-I**	**Cov-II**
ECI-3 (1)	0.38	**0.15**	**0.80**	0.64	**0.21**	**0.83**
ECI-3 (0)	0.55	0.14	0.70	0.54	0.17	0.76
UCB-1	**0.86**	0.08	0.41	**0.91**	0.08	0.41
Random	0.28	0.09	0.77	0.44	0.15	0.77

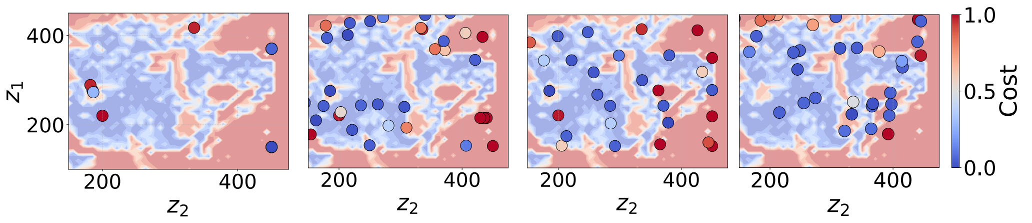


Figure 2: 2D scatter plots of z parameters from $N = 50$ evaluations using UCB-1, ECI-3 (1), ECI-3 (0), and Random sampling (Left to Right) for Push-T failures (Section 5.1.1). UCB yields poor coverage in parameter space. ECI variants prioritize coverage, providing efficient scenario diversity by trading off constraint satisfaction. Colorbar shows cost value, higher being more severe failure.

5.1.2 Failure discovery in experimental setup (Push-T hardware)

Table 2: Performance metrics for Push-T hardware experiments.

Method	P.S	M1	M2	C-I	C-II	Avg 1	Avg 2
ECI (1)	**0.30**	**0.45**	0.6	**0.075**	**0.073**	**0.43**	**0.55**
ECI (0.5)	0.15	0.28	0.25	0.050	0.059	0.28	0.52
ECI (0)	0.10	0.25	**0.65**	0.060	0.057	0.25	0.52
Random	0.025	0.18	0.3	0.015	0.015	0.18	0.20

We deploy the simulation-trained Diffusion Policy [37] on a UR3E collaborative robot end-effector. Due to the sim-to-real gap, the policy encounters previously unseen failures in hardware. We consider two failure modes ($M = 2$): **Mode 1** arises from joint limits or self-collisions, preventing the robot from reaching goal states. **Mode 2** results from sparse training data and sensor noise, leading to inefficient trajectories and task failure. Fig. 3 shows trajectories corresponding to failure modes from scenarios generated by our method.

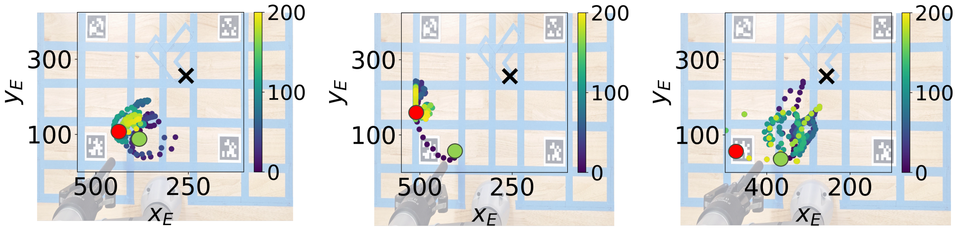


Figure 3: Left to Right: Trajectories corresponding to failure scenarios due to **Mode 1**, **Mode 2** and **Mode 1 and 2** in Push-T hardware task. Colorbar shows time horizon. More details in Appendix C.3

Experiment 2A (*Hardware experiment with human expert*): We conduct $N = 20$ evaluations across two seeds for Random Walk and ECI ($\lambda = 0, 0.5, 1.0$). Table 2 reports mode-wise failure discovery rates (M1, M2) and average costs (Avg 1, Avg 2) amongst other metrics, using $\delta_1 = 0.2$, $\delta_2 = 0.3$, radii $r_p = r_m = 0.05$. An additional failure mode—T-block motion off the table due to workspace limits was observed (**Mode 3**). **Experiment 2B** (Appendix C.3) shows how our strategy adaptively incorporates such newly observed modes online.

5.2 Failure discovery in Self-driving

We consider contextual failure discovery for perception failures and AEB system with LLM-based evaluation for each. Details of scenario design and LLM-based evaluation are provided in Appendix C.4 and Appendix C.7 respectively.

5.2.1 Contextual failures of perception module in Self-driving (YOLO+CARLA)

This case study targets contextual failures in YOLO-based object detection [38] in the CARLA simulator [39]. We consider two failure modes ($M = 2$): **Mode 1**—mispredictions due to distant non-ego agents, and **Mode 2**- errors under poor lighting. Fig. 4 shows examples of failures from scenarios generated using our method. We use $r_p, r_m = 0.05$ for $N = 30$ with 4 seeds.

Experiment 3A (*Coverage across severity levels*): Table 3 summarizes the performance across all metrics for ECI and Random walk sampling. We also conduct experiment **Experiment 3B** (Appendix C.4) with LLM evaluation augmented with user-defined conditions for **Mode 2** failures.

5.2.2 Contextual failures in Autonomous Emergency Braking systems (Simulink AEB)

This case study evaluates braking time of an AEB system with sensor fusion on a nonlinear vehicle model in Simulink [40]. The system prioritizes braking for non-ego agents but occasionally exhibits early stops due to sudden detections. We evaluate two failure modes ($M = 2$): **Mode 1**—early braking due to partial occlusion of the cyclist by a parked vehicle; **Mode 2**—early activation from delayed detection of a crossing pedestrian, measured by large ego-pedestrian distance at braking.

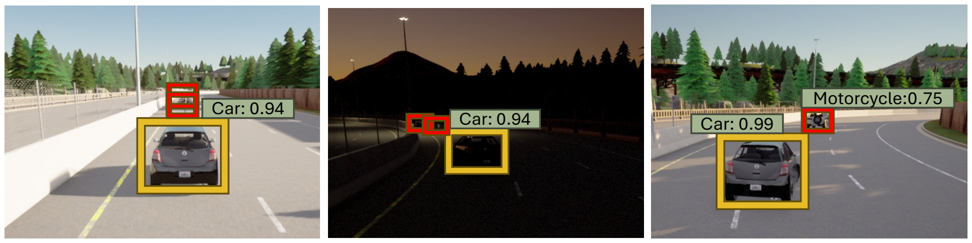


Figure 4: Left to Right: misdetection due to **Mode 1** (distance), **Mode 2** (poor light) and **Mode 1 and 2** (distance and poor light), respectively. Bounding boxes for detected objects (misdetections) shown in yellow (red) with detection confidence numbers. Each scene has two cars and a pedestrian.

Table 3: Performance metrics by method under different values of δ (YOLO-CARLA).

Method	$\delta_1, \delta_2 = 0.1$			$\delta_1, \delta_2 = 0.5$			$\delta_1, \delta_2 = 0.8$		
	P.S.	**C-I**	**C-II**	**P.S.**	**C-I**	**C-II**	**P.S.**	**C-I**	**C-II**
ECI (1)	0.59	0.09	0.36	**0.35**	0.061	**0.37**	**0.43**	0.064	0.28
ECI (0.5)	**0.625**	0.089	**0.375**	0.3	**0.091**	0.21	0.32	**0.085**	0.29
ECI (0)	0.58	0.085	0.36	0.3	0.046	0.22	0.32	0.058	**0.32**
Random	0.53	**0.1**	0.24	0.0	0.0	0.0	0.0	0.0	0.0

Experiment 4A (*Demonstration on high dimensional scenario*): We demonstrate our method for $N = 40$ evaluations, for $\delta_1 = 0.5, \delta_2 = 0.1$ and $r_p, r_m = 0.2$ for 2 seeds. All results for this experiment are provided in Appendix C.5, with additional implementation details. Table 6 (Appendix C.5) reports the results of our experimental analysis.

5.3 Results

C1: Across severity levels, all ECI variants outperform or match Random sampling (Tables 1,3). Random's performance drops at higher severities, failing to find critical failures. UCB excels in constraint satisfaction (Table 1) but suffers from poor coverage, as it stops exploring early (Fig. 2).

C2: All ECI variants outperform the baseline, but $\lambda = 0$ performs consistently well across metrics—especially Coverage-II—even without directly optimizing it. This suggests interplay between coverage metrics depends on the underlying cost. See Appendix C.8 for details.

C3: Tables 2, 6 show that our method discovers both failure for both applications modes reliably, while Random walk is sensitive to frequency of occurrence of the modes, leading to low Positive Samples. Our approach uses coverage centric exploration to be agnostic to frequency of occurrence.

6 Conclusion

In this paper, we proposed a framework for discovering contextual failures, under the constraints that the cost of evaluation is high, and the failures are not well modeled using analytical functions. In Section 4, we proposed a Bayesian active learning strategy for sequential discovery of failure scenarios, that incorporates expert evaluation to learn multiple surrogate models for failures. In Section 5, we showed that our method can discover diverse failure modes across a range of contextual failure tasks. Our method can be used towards performance improvement of autonomous systems, and interpreting user preferences in autonomous decision making (Appendix D). Our current methodology does not explicitly accommodate stochasticity from experimentation (aleatoric uncertainties), and the active learning strategy focuses mainly on uncertainty from lack of evaluations (epistemic uncertainties). In our future work, we are interested in expanding the contextual failure discovery to a stochastic setting.

7 Limitations

7.1 Explicitly accounting for stochasticity in evaluation

As mentioned, our current setup does not explicitly account for randomness in evaluation, or noise in system observables, based on which the expert evaluation is provided. This can be achieved by modifying the likelihood function in our Bayesian inference update procedure. Additionally, our active learning strategy makes an assumption that a scenario that has been evaluated must not be revisited. In the presence of uncertainty in evaluation, this constraint must be lifted, and repeat evaluations must be permitted to also account for confidence of evaluation. Hence, this would lead to a different active learning strategy, one that is beyond the scope of the present work. Across all the experiments, any modules with known stochasticity were seeded for reproducible behavior to demonstrate the proposed technique, however, in a more generalized setting, stochasticity associated with different modules must be explicitly incorporated, as this is relevant to failure analysis of autonomous systems.

7.2 Accuracy of expert evaluation

Our method assumes an expert, and does not distinguish between a human and LLM to the extent of accuracy of evaluation provided by each. There is no way to provide ground truth assessment of evaluations provided by each, and both have challenges with accurate evaluations. For humans, the decision may vary with time, and may lead to self-inconsistent evaluations for highly subjective failures. In such a case, multiple iterations of evaluations must be performed. With LLMs, inaccuracy in logical analysis and hallucination can occur, which cannot be mitigated directly with our approach. In future work, we aim to explore ways of grounding the LLM evaluation in a user preference in a structured way, leveraging recent works in constrained auto-regressive generation [41].

7.3 Requirement of increasing sample size with dimension and scenario complexity

To efficiently discover relevant failure modes, the number of samples required for evaluation may increase with the increase in the dimensionality of the search space. If a system permits low cost evaluation for failure, traditional techniques discussed in Section 2 are more suitable. Our approach works best when it is not possible to perform low cost evaluations. For systems with such constraints that ideally require more samples for extensive evaluation, but the requirement cannot be practically met, the prior and choice of surrogate model plays a critical role in capturing user's belief of failure regions in scenario parameter space. This imposes a dependency on careful prior design for complex system evaluation with sampling constraints. This issue can be avoided to a great extent with careful design of scenario to incorporate key variables of interest, and design efficient scenario parameters.

Presently, user feedback in scenario construction itself (i.e, the size and choice of scenario settings) is not explicitly accounted for, and assumed to be known a-priori for the methodology. For example, in discovering failures in object detection due to bad light and distance, wind speed is not heavily influential, and was therefore excluded from the set of weather parameters considered in the scenario definition. In future, automating the scenario construction can be a possible interesting direction of work. Regardless, it is possible that the scenario becomes complex and grows large in size. To this end, our recommendation is to use surrogate models that are compatible with high sample sizes, and dimensionality, such as Bayesian Neural Networks [42].

Acknowledgments

This work was partly supported by the Office of Naval Research (ONR) grant N000142512080, Air Force Office of Scientific Research (AFOSR) grant FA9550-23-1-0099, and the Defense Science and Technology Agency in Singapore. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods: 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings 7*, pages 127–142. Springer, 2015.

[2] J. M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Algorithmic foundations of robotics vi*, pages 107–121. Springer, 2005.

[3] A. Corso, R. Lee, and M. J. Kochenderfer. Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[4] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019.

[5] A. Sinha, M. O'Kelly, R. Tedrake, and J. C. Duchi. Neural bridge sampling for evaluating safety-critical autonomous systems. *Advances in Neural Information Processing Systems*, 33: 6402–6416, 2020.

[6] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research*, 72:377–428, 2021.

[7] P. Koopman and M. Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, 2017.

[8] M. Li, B. Meng, H. Yu, K. Siu, M. Durling, D. Russell, C. McMillan, M. Smith, M. Stephens, and S. Thomson. Requirements-based automated test generation for safety critical software. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2019.

[9] C. Dawson and C. Fan. A Bayesian approach to breaking things: efficiently predicting and repairing failure modes via sampling. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=fNLBmtyBiC.

[10] A. Parashar, J. Yin, C. Dawson, P. Tsiotras, and C. Fan. Learning-based bayesian inference for testing of autonomous systems. *IEEE Robotics and Automation Letters*, 2024.

[11] H. Delecki, A. Corso, and M. Kochenderfer. Model-based validation as probabilistic inference. In *Learning for Dynamics and Control Conference*, pages 825–837. PMLR, 2023.

[12] M. O' Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi. Scalable End-to-End Autonomous Vehicle Testing via Rare-event Simulation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[13] U. Ghai, D. Snyder, A. Majumdar, and E. Hazan. Generating Adversarial Disturbances for Controller Verification. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, pages 1192–1204. PMLR, May 2021.

[14] N. Hanselmann, K. Renz, K. Chitta, A. Bhattacharyya, and A. Geiger. KING: Generating Safety-Critical Driving Scenarios for Robust Imitation via Kinematics Gradients. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*, pages 335–352, Berlin, Heidelberg, Oct. 2022. Springer-Verlag. ISBN 978-3-031-19838-0. doi:10.1007/978-3-031-19839-7_20.

[15] E. Wong and Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning*, pages 5286–5295. PMLR, 2018.

[16] A. Parashar, K. Garg, J. Zhang, and C. Fan. Failure prediction from few expert demonstrations. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*.

[17] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. A. Nesnas, and M. Pavone. Semantic anomaly detection with large language models. *Autonomous Robots*, 47(8):1035–1055, 2023.

[18] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone. Real-time anomaly detection and reactive planning with large language models. *arXiv preprint arXiv:2407.08735*, 2024.

[19] C. Xu, D. Zhao, A. Sangiovanni-Vincentelli, and B. Li. Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In *The Second Workshop on New Frontiers in Adversarial Machine Learning, ICML*, 2023.

[20] P. Huang, X. Zhang, Z. Cao, S. Liu, M. Xu, W. Ding, J. Francis, B. Chen, and D. Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In *Conference on Robot Learning*, pages 734–760. PMLR, 2023.

[21] C. Dawson, V. Tran, M. Z. Li, and C. Fan. Rare event modeling with self-regularized normalizing flows: what can we learn from a single failure? *arXiv preprint arXiv:2502.21110*, 2025.

[22] H. Delecki, M. Itkina, B. Lange, R. Senanayake, and M. 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.

[23] C. Dawson, A. Parashar, and C. Fan. Beyond adversarial examples: sampling and repairing diverse failures with radium.

[24] L. A. Kruse, A. Tzikas, H. Delecki, M. Arief, and M. J. Kochenderfer. Enhanced importance sampling through latent space exploration in normalizing flows. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17983–17989, 2025.

[25] R. Lipkis and A. Agogino. Discovery and analysis of rare high-impact failure modes using adversarial rl-informed sampling. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 123–140. Springer, 2023.

[26] Y. Zhou, S. Booth, N. Figueroa, and J. Shah. Rocus: Robot controller understanding via sampling. In *Conference on Robot Learning*, pages 850–860. PMLR, 2022.

[27] Y. Cao, B. Ivanovic, C. Xiao, and M. Pavone. Reinforcement learning with human feedback for realistic traffic simulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14428–14434. IEEE, 2024.

[28] J. Hejna, R. Rafailov, H. Sikchi, C. Finn, S. Niekum, W. B. Knox, and D. Sadigh. Contrastive preference learning: Learning from human feedback without reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[29] C. Agia, R. Sinha, J. Yang, Z.-a. Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *arXiv preprint arXiv:2410.04640*, 2024.

[30] K. Chakraborty, Z. Feng, S. Veer, A. Sharma, B. Ivanovic, M. Pavone, and S. Bansal. System-level safety monitoring and recovery for perception failures in autonomous vehicles. *arXiv preprint arXiv:2409.17630*, 2024.

[31] T. Tran, T.-T. Do, I. Reid, and G. Carneiro. Bayesian generative active deep learning. In *International conference on machine learning*, pages 6295–6304. PMLR, 2019.

[32] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[33] G. Malkomes, B. Cheng, E. H. Lee, and M. Mccourt. Beyond the pareto efficient frontier: Constraint active search for multiobjective experimental design. In *International Conference on Machine Learning*, pages 7423–7434. PMLR, 2021.

[34] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[35] P. Auer. Using upper confidence bounds for online learning. In *Proceedings 41st annual symposium on foundations of computer science*, pages 270–279. IEEE, 2000.

[36] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[37] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[38] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma. A review of yolo algorithm developments. *Procedia computer science*, 199:1066–1073, 2022.

[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[40] W. Hulshof, I. Knight, A. Edwards, M. Avery, and C. Grover. Autonomous emergency braking test results. In *Proceedings of the 23rd International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, pages 1–13. National Highway Traffic Safety Administration Washington, DC, USA, 2013.

[41] A. K. Lew, T. Zhi-Xuan, G. Grand, and V. K. Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

[42] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17 (2):29–48, 2022.

[43] C. P. Robert and G. Casella. *The Metropolis—Hastings Algorithm*, pages 267–320. Springer New York, New York, NY, 2004. ISBN 978-1-4757-4145-2.

[44] D. Tran, R. Ranganath, and D. M. Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

[45] S. Chitta, I. Sucan, and S. Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.

[46] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*, pages 63–78, 2019.

[47] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang. Git: A generative image-to-text transformer for vision and language. *Transactions on Machine Learning Research*, 2022.

Contents

A Literature Review

Model based methods: These construct failure discovery as a cost-guided search for a known dynamic system, and can be further sub-divided into two categories, based on the method used for executing the search, namely, search using sampling-based methods [22, 12, 5, 9] and optimization techniques [15, 14]. The sampling-based techniques construct the problem of failure discovery and repair as a probabilistic search, which can then be solved using sampling techniques like MCMC, Metropolis Hastings [43], etc. The optimization methods on the other hand, model the problem of failure discovery as an optimization problem to be solved using gradient based tools [14]. As discussed previously, these methods make strict assumptions regarding analytical cost models of failures, knowledge of underlying dynamic system, and samples available for evaluation, and cannot be used towards contextual failure discovery in high cost settings. Our method relaxes each of these assumptions and works well for black-box systems, incorporating expert feedback from observed rollouts, in a data-efficient manner.

Learning based methods: These methods relaxes some of the requirements pertaining to access to a model, by taking a data-driven approach and leveraging learning based models for failure discovery [19, 20, 18], using the data to either learn a model for dynamic system, or failure directly. Our method falls within the scope of learning based methods by using observed data to construct a surrogate model for failure and is generative by design. A key challenge with existing techniques is the data intensive requirements for training generative architectures such as Diffusion models, which may not always be available for failures [7]. Additionally, different failure modes have different frequency of occurrence, leading to imbalance in dataset, which is not inherently accounted for. These methods also require a well-defined cost function to distinguish failures from nominal events. Our method is well-suited for data efficient applications due to the sequential design strategy. Additionally, using Bayesian inference for learning surrogate models allows to explicitly accommodate epistemic uncertainty associated with lack of sufficient evaluations. Learning separate surrogate models for each failure mode helps to mitigate the imbalance concern, and we prioritize even exploration of all failure modes, agnostic to their frequency of occurence, using our coverage centric active learning strategy.

Learning from expert feedback: Several works in literature have considered incorporating qualitative feedback in the process of system evaluation and design. Some of these works leverage human feedback for fine-tuning reward models in RLHF, and largely assume a data-extensive setting for system experimentation [27, 28]. Our work adopts some of these ideas of incorporating expert-in-the-loop for evaluation, and focuses on a data efficient setting, where sample extensive techniques such as RLHF may not be suitable. Other works leverage LLMs for qualitative failure diagnosis, and are optimized for runtime monitoring [17, 18, 29, 30]. Our work closely aligns with some of these methods in adopting expert evaluation, however, the key focus is on reproducing and exploring similar failure scenarios that are of interest to the user, by actively incorporating user feedback in the exploration of search space. Additionally, our method provides granularity of information by seeking multi-output feedback from the expert.

A.1 Quantifying diversity of failures

Some works have considered the concept of quality of failure discovery from the lens of diversity of samples collected. Collecting diverse set of failure samples is helpful from two perspectives– in providing the experts dissimilar scenarios for failures with similar root cause, and providing diverse training dataset for training surrogate models. In [23, 11], the methodology design was attributed to providing diverse failure scenarios by encouraging exploration via MCMC sampling, however, a formal method for quantifying the diversity of failures was not used. In [10], a coverage metric for estimation of failure diversity was utilized for evaluation, but the methodology did not explicitly utilize the proposed metric. Coverage of failures in sample heavy settings has previously been explored in [2], however, the sample extensive nature of the search is utilized in meeting the proposed coverage criteria.

In this work, we introduce a two fold coverage criteria, with coverage in parameter and metric space, that applies well for sample efficient settings. In most experimental evaluations, it is observed that optimizing parameter space coverage works better in practice than focusing only on metric space coverage. We analyze this with possible explanations in detail in Section C.8.

A.2 Model choice

Conventional surrogate model choices such as Neural networks allow for great modeling flexibility, but are prone to overfitting in the lack of sufficient training data. Instead, architectures that leverage Bayesian principles for model fitting inherently account for such epistemic uncertainties, and mitigate the issue of overfitting. In all our experiments, we used Gaussian Process models GP [34, 32] initialized with zero mean and RBF kernel as surrogate models, and initially trained using 5 randomly sampled data-points, and the likelihood is governed by Gaussian Process regression (GPR), which has been popularly used in limited evaluations contexts in literature. Below we discuss alternatives and recommendations that can be used with the Bayesian inference principles used in this work with the proposed active learning strategy.

1. For high dimensional exploration, if the system permits large number of evaluations, larger experimental budget should be used. In this case, the GPs experience a computational bottleneck of $\mathcal{O}(N^3)$, with N being the experimental budget. Here, VGP [44] can be used as a workaround for learning surrogate models from higher number of evaluations, due to the presence of inducing points relaxing the computational burden.

2. An alternative to GPs is Bayesian Neural Networks (BNNs) [42], which allow the flexibility of Neural network architectures in modeling, and meet probabilistic requirements of expectation and covariance computation required to implement the active learning strategy discussed in Section B. However, BNNs are not necessarily as data efficient as GPs and also work well with larger datasets.

3. Kernel modification- An alternate workaround to using GPs with RBF kernels, which can lead to limited expressivity in certain settings, is the use of deep Kernels, which involve using complex kernels to work with high dimensional inputs such as images.

4. For high dimensional scenarios that have redundancy in their definition such as rotational symmetry, etc., Variational Auto Encoders (VAEs) and Principal Component Analysis (PCA) can be used for dimensionality reduction.

B Coverage-based active learning strategy

The implementation for the active learning strategy for parameter coverage was adopted from [33] and metric coverage was added separately in our implementation, details of which are provided below.

B.1 Implementation of ECI

As noted in Eq. 3, the active learning strategy performs optimization of function α given by:

$$\alpha(z|\mathcal{D}^k) = \mathbb{E}_{\boldsymbol{\gamma}(z)}[C(\mathcal{D}^k \cup (z, \boldsymbol{\gamma}(z))) - C(\mathcal{D}^k)] \tag{5}$$

With $\boldsymbol{z}_k = (z_j)_{j=1}^k$ and $\boldsymbol{z'} = [z, \boldsymbol{z}_k]$, this can be expanded as:

$$\alpha(z|\mathcal{D}^k) = \lambda \mathbb{E}_z[\text{Vol}(\{\mathbb{N}_p(\boldsymbol{z'}) \cap \Omega\} \setminus \mathbb{N}_p(\boldsymbol{z}_k))] + (1-\lambda)\mathbb{E}_z[\text{Vol}(\{\mathbb{N}_m(\boldsymbol{z'}) \cap \Omega\} \setminus \mathbb{N}_m(\boldsymbol{z}_k))]. \tag{6}$$

The optimization of α is carried out using a Botorch acquisition function class. For the exact implementation, we need probability of acceptance in parameter and metric space for a proposed candidate parameter, that depends on C_p, and C_m. Our active learning strategy initializes by choosing N_{sample} proposal candidates $\boldsymbol{z}_s$. Assuming $k-1$ evaluations have been performed, $\boldsymbol{z}_{k-1}$ denote the set of candidates that have already been selected.

The selected candidates $z \in \boldsymbol{z}_s$ for which condition of expected constraint satisfaction is not met, i.e., $\mathbb{E}[q_m^*(z)] \leq \delta_m$ for at least one $m \in [1, \ldots, M]$, are rejected using a smooth sigmoid masking to assign a very low probability of acceptance $p_{\text{satisfy}}(z)$ to these samples. The remaining samples are assigned a uniform, high probability of acceptance.

Coverage in parameter space: The covariance k, of the learnt surrogate model for two points z, z' given by $k(z, z')$ is used as a measure of the Euclidean distance between the two points. To estimate expected value of neighbhourhood $\mathbb{N}_p$ in parameter space for a collection of points $\boldsymbol{z}_{\text{proposed}} = [z_{k-1}, z_s]$ for $z_s \in \boldsymbol{z}_s$, we assign probability p_1 to samples for which $k(\boldsymbol{z}_{k-1}, z_s) > r_p$, and a low probability to the remaining samples, again using smooth sigmoid masking. This is used to define a probability of acceptance in parameter space over all proposed candidates using the cumulative $p_1 p_{\text{satisfy}}$.

Coverage in metric space: Similarly, the expected value of q_m^* is used to estimate the distance in metric space. Specifically, for each pair of points (z, z') in $\boldsymbol{z}_{\text{proposed}}$, we estimate $||\mathbb{E}[\boldsymbol{q}(z)] - \mathbb{E}[\boldsymbol{q}(z')]||_2^2$, and this is used to assign a probability p_2 to candidates for which $||\mathbb{E}[\boldsymbol{q}(z)] - \mathbb{E}[\boldsymbol{q}(z')]||_2^2 > r_m$, and estimate probability of acceptance in metric space over all proposed candidates as $p_2 \cdot p_{\text{satisfy}}$. Overall probability of acceptance is then estimated as $p_{\text{acceptance}} = \lambda p_1 \cdot p_{\text{satisfy}} + (1-\lambda) p_2 \cdot p_{\text{satisfy}}$. This is used to estimate the pdf $p_{\text{acceptance}}(\boldsymbol{z}_s)$, finally used towards the optimization of α.

C Experimental details

C.1 Baselines and metrics

We compare the proposed active learning strategy against Random walk baseline for all experiments. We also compare against Upper Confidence Bound (UCB) [35] for Push-T (sim) task, as UCB is used extensively in a single objective optimization setting to formalize the exploration-exploitation trade-off in a parameter space using the following equation:

$$\alpha_{\text{UCB}}(z) = \mathbb{E}(q_m^*(z)) + \beta_m \sqrt{\text{Var}(q_m^*(z))}. \tag{7}$$

Here, Var refers to variance. Since (7) is only valid for a single objective, we compare (7) in Push-T (Sim) failure discovery with only one failure mode ($M = 1$), and our proposed strategy for three values of $\beta_1 = 0.1, 0.5, 1.0$ (reported as UCB-1, UCB-2 and UCB-3 respectively in the experimental analysis). For the sake of consistency, all scenarios are designed such that the scenario parameter is normalized between 0 and 1 before inputting to the surrogate model.

We evaluate our active learning strategy with various hyperparameter settings on three metrics– **Positive samples** (short P.S.), **Coverage-I** (short Cov-I), and **Coverage-II** (short Cov-II). **Positive samples** has been adopted from [33] and corresponds to the number of samples proposed by each strategy within the set Ω, and measure the degree of constraint satisfaction across all failure modes. **Coverage-I** and **Coverage-II** measure the actual coverage of parameter space and metric space

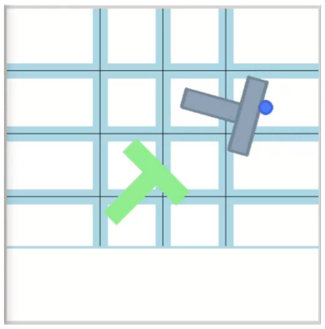


Figure 5: Push-T-Simulation task: The T-region in green corresponds to the target area, and grey denotes location of T-block at a given time, and circle shows the location of end-effector. The white region at the bottom denotes extra space that is present in simulation platform but not accounted for in hardware setup, which leads to **Mode 3** failures discussed later in Section C.3

by the generated samples, and correspond to the actual coverage metrics in parameter and metric space respectively, estimated using $C_p(\mathcal{D})$ and $C_m(\mathcal{D})$. We show experiments across a range of severity values δ_m (**C1**) and also report mode-wise satisfaction rate (**C3**). To explore the role of each coverage metric of failure diversity, we report results for three values of λ for (3), $\lambda = 0, 0.5, 1.0$, where $\lambda = 0, 1$ correspond to the extreme cases of $C = C_p$ and $C = C_m$ respectively (**C2**).

C.2 Push-T Simulation

Problem setup: The pymunk simulator simulates the contact dynamics between T-block and end-effector, which is also used to collect human demonstrations to train the Diffusion policy for deployment. The cost function $\gamma_1(z)$ measures the extent of overlap of the T-block with the target area, which is 0% if $\gamma_1(z) = 1.0$ and 100% if $\gamma_1(z) = 0.0$ for a given z. Fig 5 shows the setup of pymunk simulator considered. The actual cost landscape of γ_1 is highly discontinuous, as visible in Fig. 2.

Experiment 1A Table 4 provides detailed evaluation results with additional results unreported in Table 1 due to space restrictions. Table 1 reports the results for best from each method.

Table 4: Performance metrics by method under different values of δ (Push-T simulation)

Method	$\delta = 0.9$(high severity failure discovery)			$\delta = 0.3$(low severity failure discovery)		
	Positive Samples	**Cov-I**	**Cov-II**	**Positive Samples**	**Cov-I**	**Cov-II**
ECI-1 (1)	0.48	0.06	0.72	0.58	0.07	0.72
ECI-2 (1)	0.44	**0.17**	0.75	0.63	**0.25**	0.76
ECI-3 (1)	0.38	0.15	**0.80**	0.64	0.21	**0.83**
ECI-1 (0)	0.49	0.04	0.71	0.62	0.07	0.75
ECI-2 (0)	0.44	0.07	0.56	0.61	0.16	0.77
ECI-3 (0)	0.55	0.14	0.70	0.54	0.17	0.76
UCB-1	**0.86**	0.08	0.41	**0.91**	0.08	0.41
UCB-2	0.84	0.11	0.49	0.90	0.12	0.49
UCB-3	0.84	0.14	0.45	0.87	0.15	0.45
Random	0.28	0.09	0.77	0.44	0.15	0.77

Experiment 1B (*Hyperparameter analysis*): From Table 4, it is evident that there are two main hyper-parameters affecting the performance of our proposed algorithm, radius of coverage, and level of severity. For high-dimensional problems, with multiple contextual failure modes and a hybrid setting, with $\lambda \in (0, 1)$, the influence of these hyperparameters can be challenging to interpret. We exploit the simplicity of this problem to analyze the interplay of r_p, r_m and δ. Fig. 6 and Fig. 7

show heatmaps of the three metrics for a fine-grained values of δ and r, for $\lambda = 1, 0$ respectively, for a single seed evaluation. Evidently, r_p influences coverage in parameter space directly, (Fig. 6, middle), across all values of δ. This is consistent with the findings in Table 4, with low performance metrics for smaller values of r_p due to insufficient exploration. The relationship of r_p and coverage in metric space is not as direct, and sensitive to δ. This is also consistent with findings across experiments, where the method that recorded highest C_m value varied from experiment to experiment. Fig. 7 (middle) shows that the relationship between r_m and coverage in metric and parameter space is more direct, but also highly dependent on δ. Fig. 7 suggests that C_m is more strongly dependent on δ, and less dependent on r_m itself, and therefore, for optimizing diversity, in the absence of a preferred severity level, the recommendation is to choose low severity threshold δ, to experience better overall performance across both coverage metrics, for both $\lambda = 1.0, 0.0$.

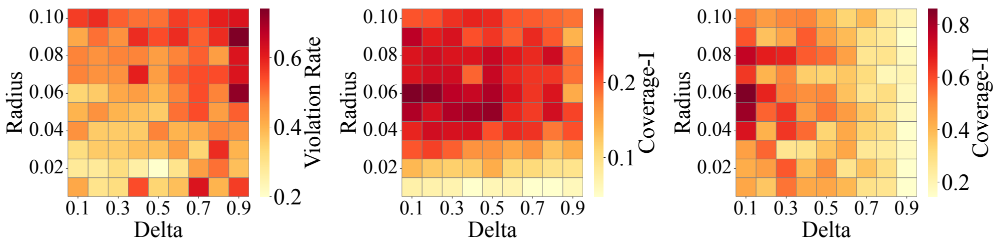


Figure 6: Left to right: Constraint violation rate, Coverage-I and Coverage-II reported for various values of parameter radius r_p and delta δ for (3) with $\lambda = 1$.

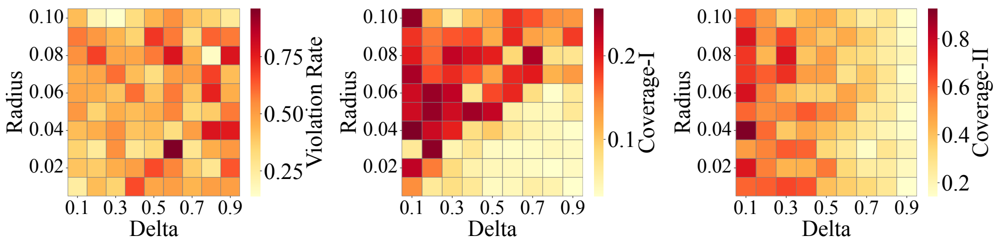


Figure 7: Left to right : Constraint violation rate, Coverage-I and Coverage-II reported for various values of metric radius r_m and delta δ for (3) with $\lambda = 0$.

Experiment 1C *Prediction accuracy*: We use the ground truth data from constructed cost function to evaluate the prediction accuracy of the learnt surrogate models. We construct a test dataset of 25 points, $\mathcal{D}_{\text{test}} = (z_i, \gamma(z_i))_{i=1}^{25}$ and use absolute error $m(z_i) = |\gamma(z_i) - q_1^*(z_i)|$ as a metric of performance. For each seed and method reported in Table 4, we tally the number of scenarios for which $m(z_i) < 0.1$, which corresponds to high prediction accuracy. The mean and standard error of prediction estimated across all seeds is reported in Fig. 8 for $\delta_1 = 0.9, 0.3$. Clearly, ECI with a very low radius r performs poorly in prediction, due to extremely constricted sampling region. In terms of global performance, ECI methods do well, but the performance depends on the value of radius r. It must also be noted that UCB tends to perform consistently across all β values and is more immune to hyperparameter sensitivity than our method. These experimental results show that very low values of r are not suitable for sufficient exploration for learning a good quality surrogate model, but the discussion on what value of r gives best performance is open-ended. In future works, we aim to address optimization of the value of radius to achieve optimal prediction accuracy.

C.3 Push-T Hardware

Problem Setup: We implement the policy learnt and fine-tuned in simulation using human demonstrations, to perform the task of pushing the T-block using UR3E collaborative robot arm, by engaging a low-level MoveIt planner [45] to reach a set of goal locations supplied by the Diffusion

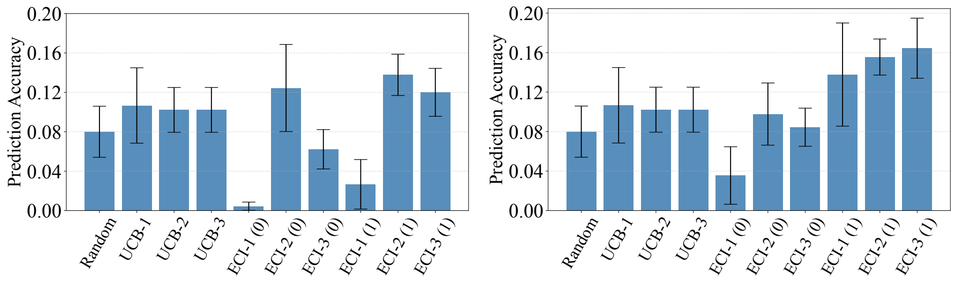


Figure 8: Prediction accuracy for Push-T sim task, histograms and error bars showing mean and standard error values for a test dataset of 25 uniformly generated points for $\delta = 0.9$ (left) and $\delta = 0.3$ (right) respectively.

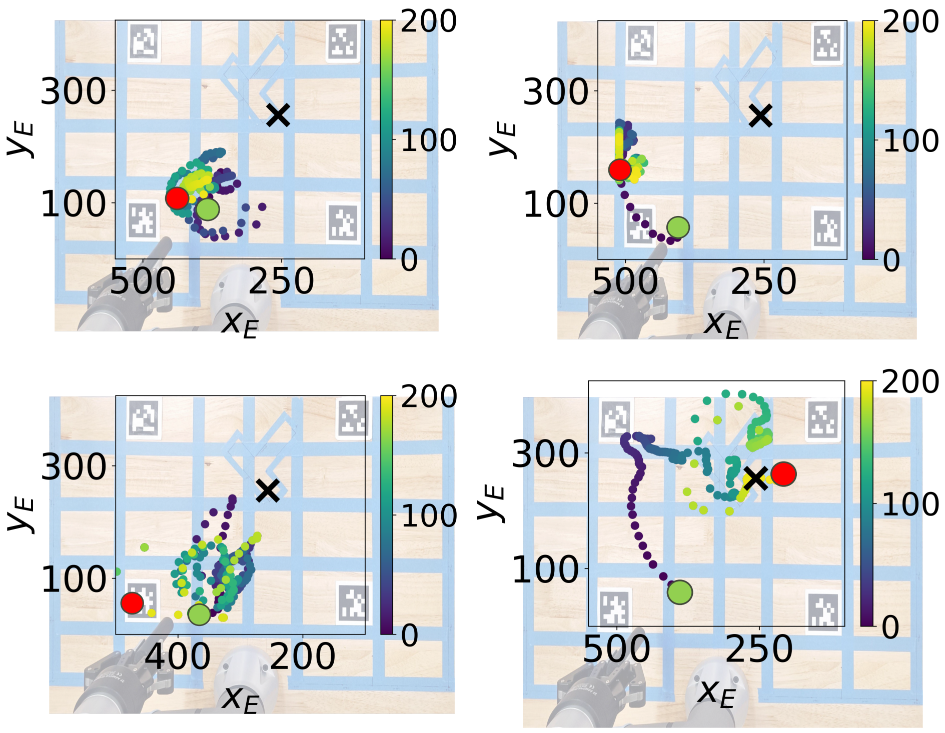


Figure 9: Left to Right: Trajectories from experiments corresponding to failure scenarios due to **Mode 1** (top left), **Mode 2** (top right), **Mode 1 and 2** (bottom left), and **Mode 3** (bottom right) in Push-T hardware task. Colorbar shows time horizon. More details in Appendix C.3

Policy [37]. Due to the evident sim-to-real gap in the dynamic systems on which the policy is trained and deployed on, we observe failures that were previously unseen in simulation.

Failure modes: We considered two modes of failures here ($M = 2$), **Mode 1** failure correspond to failure due to joint angle limits and self-collision, which was observed due to parts of the trajectory being very close to the manipulator arm. This led to some trajectories failing at accomplishing the task as the actions required arm to be pushed beyond its limit. There was also a separate category of failures, which we refer to as **Mode 2** failures, which occur due to lack of sufficient training data in a specific region, leading to inefficient movements that eventually led to task failure. **Mode 1** failures were considered terminal, and led to early-stopping, while **Mode 2** failures can be corrected by supplying more training data for the policy, and were not observed to be a safety hazard in general. Fig. 3 shows examples of trajectories with **Mode 1** and **Mode 2** failures. A human expert was used to provide evaluations γ_1, γ_2 as in (4).

Table 5: Performance metrics for Push-T hardware experiments (3 modes)

Method	Avg 1	Avg 2	Avg 3	M1	M2	M3	C-I	C-II
ECI (1)	**0.15**	0.30	0.52	**0.15**	0.3	**0.65**	0.35	0.11
ECI (0.5)	0.0	0.20	**0.55**	0.0	0.2	0.8	0.22	0.09
ECI (0)	0.0	**0.35**	0.2	0.0	0.35	**0.35**	0.22	0.09
Random	0.07	0.27	0.3	0.1	0.3	0.45	**0.42**	**0.15**

We performed $N = 20$ evaluations for two seeds, for Random Walk and ECI with $\lambda = 0, 0.5, 1.0$. Table 2 shows results of experimental evaluations. We also present average costs for **Mode 1** and **Mode 2** failures reported as **Avg 1** and **Avg 2** respectively, in addition to mode specific failure discovery rate (second and third column in Table 2). For all ECI baselines, severity levels were chosen as $\delta_1 = 0.2, \delta = 0.3$, and radii were chosen as $r_p, r_m = 0.05$.

Experiment 2B (*Adaptive addition of failure mode*): In one of the runs for **Experiment 2A**, a new type of failure was observed, which was not accounted for a-priori, which we refer to as **Mode 3**, and pertains to failure due to manipulator reaching workspace limits which was not accounted for in the simulation design. Fig. 9 (bottom right) shows trajectory corresponding to **Mode 3** failure. The sequential nature of our strategy makes it easy to adaptively include additional failure modes observed by experts during the evaluation procedure. We use last 5 evaluations for $\lambda = 1$, for one of the seeds, as initial dataset for training a preliminary surrogate model, and revise active learning strategy to include $\gamma_3 \geq \delta_3$, with $\delta_3 = 0.3$, in addition to $\gamma_1 \geq \delta_1, \gamma_2 \geq \delta_2$ from **Experiment 2A**. Runs where no failure corresponding to **Mode 3** was observed were also used for training, by simply recording $\gamma_3 = 0$ for those iterations. Mode 3 was always observed to lead to early stopping, due to which we assign cost $\gamma_3(z) = 1$ for all runs with **Mode 3** failures.

Table 5 shows the results pertaining to $N = 20$ evaluations for different ECI variants, with one seed for all three modes. Since both **Mode 1** and **Mode 3** lead to early stopping, it was observed that they do not occur together, leading to no scenario z where all three failures occur together. Hence,we report mode-wise statistics for this study. The coverage metrics are calculated with an 'or' criteria instead of 'and' criteria, and include any point for which at least one of $\gamma_i \geq \delta_i$ is observed, for $i = 1, 2, 3$. Note that the active learning strategy still works because none of the cost constraints are hard constraints– they are soft constraints used to encourage exploration in specific regions. Despite the drastic difference in failure rates and values across failure modes, our active learning strategy fairs well in discovering high average value failures with comparable coverage to Random sampling, which performs well on coverage, given the modified coverage criteria is more relaxed and accomodates larger number of samples.

C.4 CARLA

To perform the maneuvering of the vehicle in the environment, we use the 'autopilot' feature. The environment of the car is completely described using static settings and environmental parameters supplied by the user-defined scenario. The static settings and scenario are composed to render scenes in an environment using the probabilistic programming language Scenic [46]. We choose a static setting corresponding to a pedestrian crossing a street with two non-ego cars, and the ego car and one of the non-ego cars ('lead car') are required to abruptly brake before the pedestrian. A scenario in this case is defined using $z = (b_e, b_l, s)$. Here b_e, b_l refer to the braking threshold for the ego and the lead car respectively, and s denotes the sun altitude angle, which controls the brightness of the scene. The simulations were seeded to generate reproducible results. Each evaluation corresponds to $T = 60$ steps of simulations. Images recorded from the camera view are used for object detection and classification at every 10 steps using YOLO-v3 [38], and the classified image are used as inputs to GiT [47] to predict a likely failure type based on fine-tuning data. Results obtained from YOLO along-with the reports from GiT are used as inputs to GPT3 model for failure evaluation, which is queried 6 times per evaluation, and assigns binary scores pertaining to each failure mode for

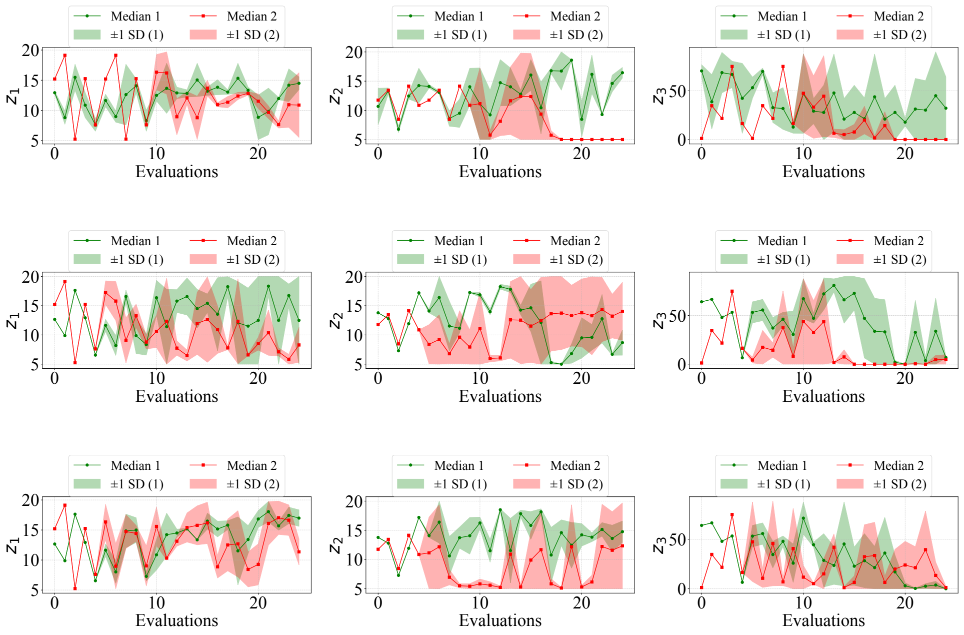


Figure 10: Sampled z values for ECI with $\lambda = 1.0, 0.5, 0.0$ (top to bottom), with $b_e, b_l, s(z_1, z_2, z_3)$ (left to right) for nominal LLM evaluation (red) and hybrid LLM evaluation. Graphs show mean (solid lines) and standard deviation (shaded). The hybrid evaluation results in higher mean values of b_e, b_l compared to nominal evaluation criteria. Table 7 shows the difference in shrinkage in explored space more prominently with decreased positive samples

each scene (camera image). The average value reported across 6 scenes is used to construct γ_1, γ_2 as in Eq (4). We discuss customization of information retrieval with severity specifications and evaluation criteria in **Experiment 3B**. All results are reported as mean values across 4 seeds. Fig. 4 shows scenes with object detection failures corresponding to **Mode 1** and **Mode 2** from scenarios generated using our methodology.

Experiment 3B (*Customizing evaluation strategy*): One of the advantages of the binary evaluation strategy used in Eq 4 is that for each timestep t, we can include additional expert knowledge to influence the evaluation. This is especially relevant to add robustness to the evaluation conducted using LLMs, where a human expert may have prior belief regarding conditions pertaining to certain failure modes. The final binary evaluation for each scene can then be represented as a composition of the evaluation given by the LLM and a pre-specified user condition.

We demonstrate this capability by augmenting our evaluation strategy with a condition $(b_l > 10 \wedge b_e > 10) \vee (b_l > 15) \vee (b_e > 15)$. Fig. 10 shows the trends of parameters z_1, z_2, z_3 for each variant of ECI for with and without (nominal) augmented evaluation for 4 seeds, for $\delta_1 = 0.8, \delta_2 = 0.8$ and $r_p, r_m = 0.05$. Table 7 shows the difference between metrics for nominal evaluation strategy with LLM for this case with augmented conditional evaluation. As we can see, number of Positive samples are reduced, which is due to the additional criteria causing feasible parameter space to shrink.

We also test for an additional conditional evaluation with $b_l > 10 \wedge b_e > 10$, with $N = 50$ evaluations using ECI with $\lambda = 0.5$. This further restricts the feasible region for parameters, as seen in Fig. 11, the values of b_l, b_e converge to the region corresponding conditional evaluation criteria, whereas nominal method converges to a lower value of b_l. This shows how user imposed conditions can be used to control the specifics of discovered failure scenarios.

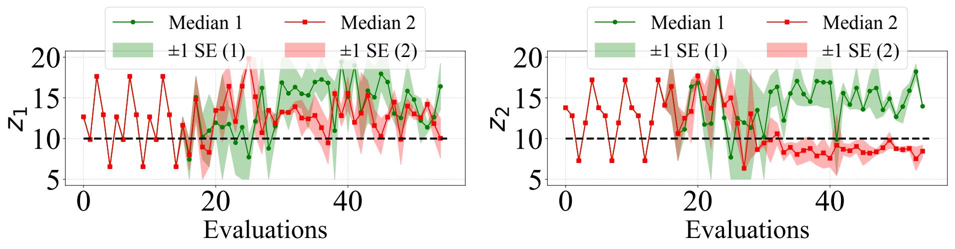


Figure 11: Mean and std deviation for ego and lead braking distance ($b_e - z_1$, $b_l - z_2$), using ECI with $\lambda = 0.5$ shown for nominal (red) and hybrid evaluation strategies (green) with condition-2 ($b_e > 10, b_l > 10$) for $N = 50$ evaluations. Dashed lines showing the minimum value of b_e, b_l based on the condition. As expected, with the conditional evaluation strategy we observe convergence to higher average values, especially for b_l.

C.5 AEB

Scenario description: This problem consists of testing late and early braking by AEB in a self-driving context, with a pedestrian and cycle crossing the road, with a parked car on the side. The scenario is parameterized by $z = [r, \alpha, t_{1p}, t_{2p}, t_{1c}, t_{2c}, t_{\text{parked}}, v_p, v_c]$. Here, r, α refer to the amplitude and time period of a sinusoidal curve that is used to design the trajectory of the ego vehicle, and the road, so the trajectory of ego vehicle is given by $x = 40\alpha t, y = r\sin(\alpha t)$, for $t \in [0, 2\pi]$. Both r, α cumulatively control the curvature of the road. The road width is fixed $w = 7$, t_{1p}, t_{2p} are used to specify the initial and final locations for the movement of pedestrian crossing from left to right, and similarly t_{1c}, t_{2c} for the cycle moving from right to left, given by:

$$\begin{aligned} x &= 40\alpha t_{ik} \\ y &= r\sin(\alpha t_{ik}) \pm 0.5w, \end{aligned} \tag{8}$$

for $i = 1, 2$ to denote initial and final location, and $k = p, c$ to denote pedestrian and cycle. Similarly, t_{parked} is used to control the location of parked car, and v_p, v_c denote the constant speeds for pedestrian and cycle respectively.

Definition of severity: We choose $h \circ g_m = \frac{1}{T}\sum(\text{AEB}_t)b_t^m$, where $\text{AEB}_t = \{0, 1, 2, 3\}$ denotes amount of braking, here we combine time and amount of braking to denote severity.

Failure modes: We observe two failure modes, **Mode 1** is delayed braking due to occlusion of cycle by the car, leading to late stopping, and **Mode 2** is early stopping due to pedestrian becoming visible later in the simulation, due to its initial position relative to the scene, and approaching the ego vehicle. For testing scenarios pertaining to **Mode 1**, we deploy a naíve condition of the cycle being parked behind the car, i.e, $t_{1c} > t_{\text{parked}}$, and for **Mode 2**, we use LLM to provide a binary label for whether a scenario qualifies as failure due to **Mode 2**, based on direction of approach of pedestrian to ego vehicle $t_{1p} > t_{2p}$, final distance between the ego vehicle and pedestrian exceeding a threshold (for early braking), relative speed of pedestrian and cycle ($v_p < v_c$) to confirm that pedestrian is discovered last in scene. Note that these failures are not related to system safety in the strictest sense, and rather asses performance asepcts of the AEB. This example demonstrates the capability of our pipeline to observe a set of scenarios with desirable properties with our method. Fig. 12 shows examples of initial conditions for scenarios for different failures considered. The supplementary material also includes a video showing the demonstration of AEB on these scenarios.

Choosing δ_1, δ_2: From the design of scenario, it can be observed that **Mode 1** occurs in a specific range of scenario parameters, hence we choose $\delta_1 = 0.5$ to reduce the search volume. We set $\delta_2 = 0.1$ as we observe a range of values for **Mode 2** failures. With the high dimension of the space, to encourage better exploration, radius was set to $r_p, r_m = 0.2$. Table 6 shows the results of all ECI variants against Random sampling. As we can see, $\lambda = 1.0$ has overall good performance, with all ECI methods performing better than Random sampling in all metrics. We also provide a

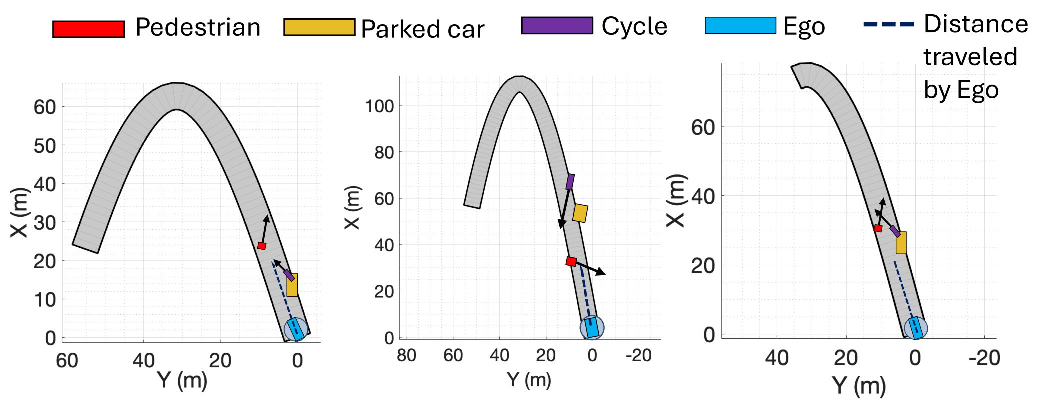


Figure 12: Left to Right: Scenario arrangements corresponding to **Mode 1** and **Mode 2**, and both **Mode 1 and 2**. Arrows indicate the initial direction of velocity vector for each dynamic non-ego agent.

Table 6: Performance metrics for AEB simulations.

Method	**P.S.**	**C-I** $\times 10^{-5}$	**C-II**	**Mode 1**	**Mode 2**
ECI (1)	**0.22**	**4.59**	**0.43**	**0.27**	0.30
ECI (0.5)	0.19	1.37	0.34	0.22	0.28
ECI (0)	0.19	2.85	0.30	0.22	0.28
Random	0.11	0.85	0.35	0.14	**0.31**

failure discovery rate for **Mode 1** and **Mode 2**, where Random sampling's performance highlights the differing frequency of occurrence for each. Also note that **Coverage-I** has a very low value, which is expected due to the scenario being high dimensional. This motivates the discussion on the number of samples needed to fully explore a scenario, which we discuss in Section C.6.

Experiment 4B (*Evaluating the role of prior belief*): To analyze the role of prior, we conducted an ablation study for the AEB failure discovery task with $\delta_1 = 0.5, \delta_2 = 0.1$, for a single seed, by fixing the number of evaluations with active learning to $N = 60$, and changing the number of initial data-points for random sampling to 20 (model-1) and 40 (model-2), that is used to generate an initial surrogate model. This corresponds to a prior belief in this context, as having more initial evaluations will lead to stronger prior belief. Fig. 13 shows the trends of the three performance metrics for $\lambda = 0, 0.5, 1.0$ for this single seed evaluation as a function of number of active learning evaluations. Clearly, not only do we achieve higher overall performance metrics with model-2, the performance metrics also converge within lesser evaluations.

Table 7: Performance metrics for nominal vs condition-1 augmented evaluation for 4 seeds, CARLA simulations ($\delta_1 = 0.8, \delta_2 = 0.8$).

Method	**Nominal**			**Condition-1**		
	P.S.	**C-I**	**C-II**	**P.S.**	**C-I**	**C-II**
ECI (1)	**0.43**	0.064	0.28	**0.34**	**0.07**	**0.31**
ECI (0.5)	0.32	**0.085**	0.29	0.11	0.024	0.27
ECI (0)	0.32	0.058	**0.32**	0.25	0.05	0.22
Random	0.0	0.0	0.0	0.008	0.001	0.10

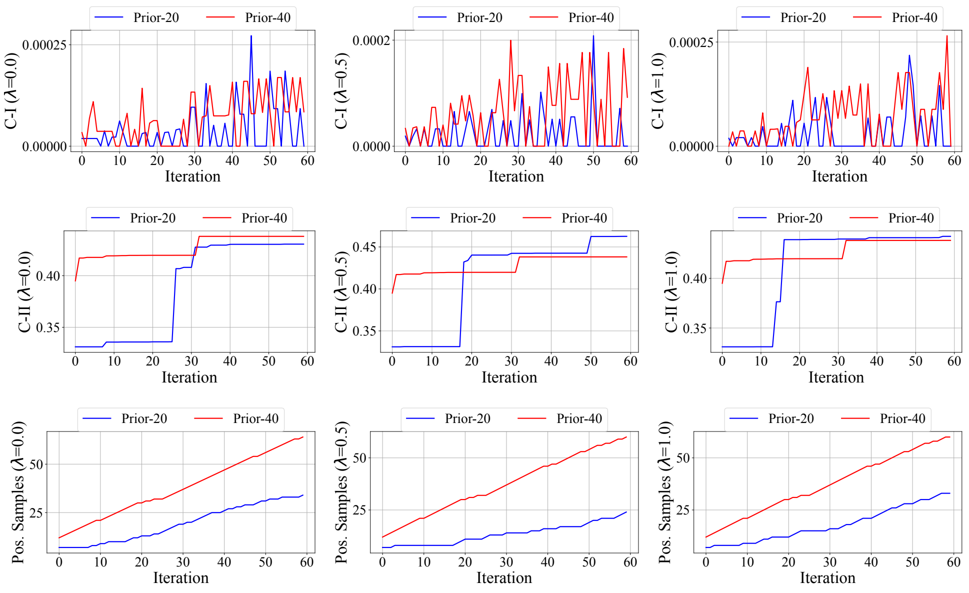


Figure 13: Performance metrics (C-I,C-II, Positive Samples- top to bottom) for ECI with $\lambda = 0, 0.5, 1.0$ (left to right), for an initial model trained with 20 ad 40 initial evaluations, proceeded by 60 evaluations using our method.

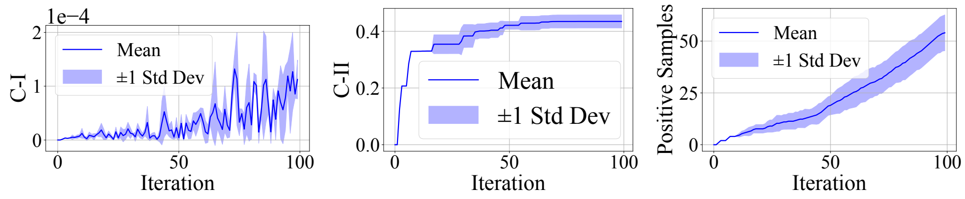


Figure 14: Metrics with experimental budget performance (AEB), $N = 100$, 1 seed. C-I reported as C-I$\times 10^{-4}$.

C.6 Experimental budget vs performance metrics

To analyze the influence of experimental budget on performance metrics, we ran AEB task with two failure modes for three seeds with $\lambda = 0.5$ for a longer experiment with $N = 100$ evaluations, with $\delta_1 = 0.5, \delta_2 = 0.1$. Fig. 14 shows the coverage metrics and positive samples detected with number of evaluations. Note that coverage in parameter space grows more slowly compared to coverage in metric space, which is due to two reasons– high dimension of parameter, leading to marginal improvement per additional evaluation, and the low dimension of metric space; since we only have two failure modes, the possible ways of covering the metric space saturates after a while. This also depends on the frequency of occurrence of failure modes and the desired range of severity for each.

C.7 LLM-based evaluation

We use GPT-3.5 model for implementing LLM based evaluation in CARLA and AEB experiments. Shown below are the prompts for each of the settings.

C.7.1 LLM-based evaluation for CARLA

We use a pretrained Generative Image to Text model (GiT) [47], fine-tuned on a dataset consisting of scenes with previously seen failures of the two modes, and nominal scenes. The GiT model takes a scene (image) as an input, and provides a caption, that is used as an input to the LLM as a part of the prompt.

The generated caption alongwith objects detected by YOLO are used by the LLM to first conclude if a detection failure has happened, which corresponds to one of the objects being undetected, and subsequently provide a discrete evaluation on whether the reason for failure corresponds to one or both of the modes. This process is conducted for several scenes observed during a simulation and processed by YOLO within each roll-out. The output generated by LLM is transformed and used to generate $\gamma_1(z), \gamma_2(z)$ for a given z as in (4). This case has no early stopping. Shown below is the prompt used for LLM based evaluation. Text in red and blue shows the input received from GiT and CARLA respectively.

Prompt Used for CARLA evaluation

You will be provides the analysis of YOLO Object detection on an image that was taken from the camera feed of CARLA simulator. The simulator is simulating a pedestrian crossing the road before a car infront of the ego car. There are two cars and one pedestrian in each image. The information provided:

1. *Objects detected*: List of objects detected by YOLO in the image. This list should have at least one object from the *Objects to detect* list
2. *Objects to detect* list:
   - One object with one of the following labels: 'car','truck',
   - One object with one of the following labels: 'car', 'truck','bus','motorcycle','bicycle'
   - One object with one of the following labels: 'person'
3. Reason: The reason is a brief explanation of the failure to detect all objects, if that happens, and is generated by a pre-trained GiT model in the form of captions for the image.

We are looking to discover images where YOLO fails to detect an object due to **bad light** and/or l**arge distance.** If the list of *objects detected* has an object missing from the *objects to detect* list, look at the reason. Follow the response instructions while responding.
Response Instructions: Respond should be an integer 0, 1, 2, 3 or 4:

- 0 indicating that at least one object was missing from the 'objects to detect' list, but the reason provided does not correspond to bad light or large distance.
- 1 indicating that an object was not detected and the reason provided corresponds to bad light only.
- 2 indicating that an object was not detected, and the reason corresponds to large distance only.
- 3 indicating that an object was not detected, and the reason corresponds to both large distance and bad light.
- 4 indicating all objects are detected. Do not provide explanation.

Response format: Response: [integer], where integer = 0,1,2,3,4.
The list of objects detected and reason for incomplete detection for the image are as follows:

- Objects detected: {objects}
- Reason: {reason}

C.7.2 LLM based evaluation for AEB

Shown below is the LLM prompt used for **Mode 2** failure evaluation. **Mode 1** failure evaluation is done using the condition $t_{1c} > t_p \wedge (\sum_{t=1}^{T}$ AEB$_t$ $>$)0, for $T \leq 10$, Here AEB$_t$ denotes the value of AEB status at each timestep, and can take values 0,1,2 or 3, corresponding to inactive, partial brakings and full brake respectively. If AEB leads to full brake, early stopping may happen. Text in blue shows observables from simulations and scenario parameters z.

Prompt Used for AEB evaluation

You will be provides the analysis of an Autonomous Emergency Braking (AEB) system, with a scene decription. Each scene consists of a cycle and a pedestrian crossing a road, a vehicle parked on the road, and an ego agent navigating in this scenario.
You will receive the following information:

1. *Cycle to Ego*: Relative distance of cycle and ego agent final distance value
2. *Pedestrian to Ego*: Pedestrian and ego agent final distance value
3. *Sim list*: Simulation time between 0 and 10.
4. *AEB Status*: reflects AEB status for each simulation timestep. AEB status at any timestep is either 0 or 1, where 0 corresponds to unactivated AEB (or nominal manuevring) and 1 corresponds to AEB activated.
5. *Map1*: relative initial arragement of cycle and parked vehicle. Map1=1 implies parked vehicle occludes cycle
6. *Map2*: relative direction of pedestrian movement. Map2=1 implies that the pedestrian is approaching the direction of ego vehicle.
7. *Road curvature*: a scalar value between 0 and 1 reflecting the curvature of road.
8. *Pedestrian Speed*: Constant speed of pedestrian crossing the road
9. *Cycle Speed*: Constant speed of cycle crossing the road

Your job is to assess if the given scenario and the response of AEB system corresponds to a contextual failure: Failure definition: Late discovery of pedestrian- this type of failure occurs if all the below conditions are met:

- AEB Status is 1 at at least one of the time-steps
- Pedestrian is approaching vehicle's direction, which corresponds to 'Map2'=1.
- If 'Map2'=0 condiiton for Failure 2 is violated.
- 'Pedestrian to Ego' distance is lower than 'Cycle to Ego', and 'Pedestrian to Ego' greater than 40
- 'Pedestrian Speed' is less than 'Cycle speed'

Evaluate the information provided for this scenario and simulation, and evaluate if conditions corresponding to Failure are met. Follow the response instructions while responding.
Response Instructions: Respond should be an integer 0, 1:

- 0 indicating that Failure has NOT happened.
- 1 indicating that Failure has happened

Response format: Response: [integer], where integer = 0,1. Do not provide explanation.
The list of objects detected and reason for incomplete detection for the image are as follows:

- 'Cycle to Ego': {ego cycle}
- 'Pedestrian to Ego': {ego ped}
- 'Sim list': {AEB time}
- 'AEB Status': {AEB data}
- 'Map1': {Map 1}
- 'Map2': {Map 2}
- 'Road curvature': {Road curvature}

C.8 Discussion on ECI variants

Across experiments, it has been observed that ECI with $\lambda = 1, 0.5$ performs better than $\lambda = 0.0$. This indicates that for coverage, parameter space coverage plays a more important role than just metric space coverage. This can also be understood from Fig. 14, where C_m quickly converges after a few iterations due to its low dimensionality and also sparse nature, however, C_p continues to increase with evaluations. Another potential issue that inhibits the performance of metric space coverage is that cost domain γ, is often very sparse. For instance, for CARLA task, it was observed that most cost outputs γ_1, γ_2 clustered around a few output values. We observe better performance of metric space coverage with examples where cost range is more continuous, such as Push-T Simulation, and AEB. This suggests that the most efficient strategy would be to optimize the value of λ based on the expected sparsity of the various failure modes. Since this is unknown a-priori, and also depends heavily on the design of cost functions and application, we defer this discussion for future research.

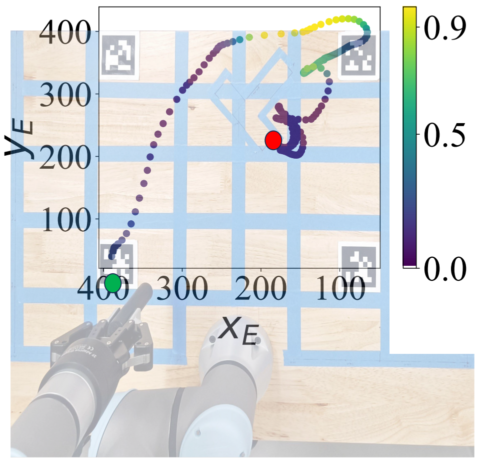
D Using failure modes

The collected failures can be used towards policy repair and additional failure diagnosis. The learnt surrogate models for various failures can be used to sample scenarios with high predicted cost γ_m, and these scenarios can be used for the next steps in failure diagnosis and system monitoring. Not all failures can be repaired, for example, failures that originate as a consequence of system limits. Nonetheless, knowledge of such irreparable failures is essential. With the contextual failures studied in this work, there is no proposed way to distinguish between failures that can or cannot be repaired.

As an example of how we can use the surrogate models for repair, we demonstrate repair of the Diffusion policy used in Push-T task on the UR3E robot. Push-T failures were generally observed to be reparable, since better human demonstrations can be supplied such that the resultant trajectories do not lead to singularities experienced in practice. As an example, we repaired some of the Mode 3 failures by sampling high cost scenarios z and requesting additional human demonstrations for these scenarios such that the generated trajectory would avoid the failure modes observed.

Fig. 15 shows an example of a scenario with predicted trajectory using the simulation. For the chosen scenario, the trajectory fails due to Mode 3 failure, as shown in left figure. After repair by retraining with additional demonstrations, we observe that the trajectory succesfully mitigates this issue. This example illustrates the benefit of learning safety critical failure scenarios from limited number of evaluations such that system damage is minimized, that can be used for further repair. For a system that permits slightly larger number of evaluations $N \sim 100-500$, the failure discovery and repair can be done in a sequential manner, as suggested in [9]. We also provide video demonstration of an example of policy repair on the UR3E manipulation arm for a failure mode observed using our method in the supplementary attachment.

Supplementary material also consists of a video showing the experimental demonstration of sampled failure scenarios for Push-T on UR3E platform, and AEB in Simulink, alongwith policy repair demo for Push-T task.



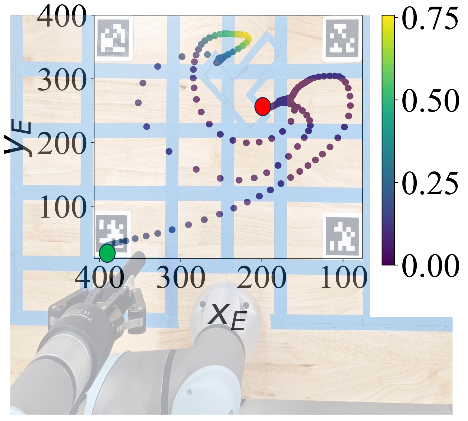


Figure 15: Push-T Hardware: **Mode 3** before repair (left) and after repair (right) respectively. Trajectories of the end effector generated using pygame environment shown. As seen from figure on left, trajectory of end effector extends beyond the workspace, leading to failure in real-time experimentation attributed to **Mode 3**. Our method helps to discover more such failures that can be subsequently used to fine-tune the diffusion policy by providing scenarios for providing better quality human demonstrations on. Figure on right shows rollout of policy trained with additinal data collected from scenarios supplied by our method. Colorbar shows range of expected value of surrogate model, $\mathbb{E}[q_3^*]$ to show states corresponding to high-cost failure (this happens because scenario and state have shared parameterization). As we can see, the updated policy successfully avoids the failure.