

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 occurrence, 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.

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

510 A.2 Model choice

511 Conventional surrogate model choices such as Neural networks allow for great modeling flexibility,
 512 but are prone to overfitting in the lack of sufficient training data. Instead, architectures that leverage
 513 Bayesian principles for model fitting inherently account for such epistemic uncertainties, and miti-
 514 gate the issue of overfitting. In all our experiments, we used Gaussian Process models GP [34, 33]
 515 initialized with zero mean and RBF kernel as surrogate models, and initially trained using 5 ran-
 516 domly sampled data-points, and the likelihood is governed by Gaussian Process regression (GPR),
 517 which has been popularly used in limited evaluations contexts in literature. Below we discuss alter-
 518 natives and recommendations that can be used with the Bayesian inference principles used in this
 519 work with the proposed active learning strategy.

520 1. For high dimensional exploration, if the system permits large number of evaluations, larger
 521 experimental budget should be used. In this case, the GPs experience a computational bottleneck
 522 of $\mathcal{O}(N^3)$, with N being the experimental budget. Here, VGP [45] can be used as a workaround
 523 for learning surrogate models from higher number of evaluations, due to the presence of inducing
 524 points relaxing the computational burden.

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

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

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

535 B Coverage-based active learning strategy

536 The implementation for the active learning strategy for parameter coverage was adopted from [32]
 537 and metric coverage was added separately in our implementation, details of which are provided
 538 below.

539 B.1 Implementation of ECI

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

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

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

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

550 **Coverage in parameter space:** The covariance k , of the learnt surrogate model for two points z, z'
551 given by $k(z, z')$ is used as a measure of the Euclidean distance between the two points. To estimate
552 expected value of neighbourhood \mathbb{N}_p in parameter space for a collection of points $\mathbf{z}_{\text{proposed}} =$
553 $[z_{k-1}, z_s]$ for $z_s \in \mathbf{z}_s$, we assign probability p_1 to samples for which $k(z_{k-1}, z_s) > r_p$, and a low
554 probability to the remaining samples, again using smooth sigmoid masking. This is used to define
555 a probability of acceptance in parameter space over all proposed candidates using the cumulative
556 $p_1 p_{\text{satisfy}}$.

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

563 C Experimental details

564 C.1 Baselines and metrics

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

$$\alpha_{\text{UCB}}(z) = \mathbb{E}(q_m^*(z)) + \beta_m \sqrt{\text{Var}(q_m^*(z))}. \quad (6)$$

569 Here, Var refers to variance. Since (6) is only valid for a single objective, we compare (6) in Push-T
570 (Sim) failure discovery with only one failure mode ($M = 1$), and our proposed strategy for three
571 values of $\beta_1 = 0.1, 0.5, 1.0$ (reported as UCB-1, UCB-2 and UCB-3 respectively in the experimental
572 analysis). For the sake of consistency, all scenarios are designed such that the scenario parameter is
573 normalized between 0 and 1 before inputting to the surrogate model.

574 We evaluate our active learning strategy with various hyperparameter settings on three metrics—
575 **Positive samples** (short P.S.), **Coverage-I** (short Cov-I), and **Coverage-II** (short Cov-II). **Positive**
576 **samples** has been adopted from [32] and corresponds to the number of samples proposed by each
577 strategy within the set Ω , and measure the degree of constraint satisfaction across all failure modes.
578 **Coverage-I** and **Coverage-II** measure the actual coverage of parameter space and metric space
579 by the generated samples, and correspond to the actual coverage metrics in parameter and metric
580 space respectively, estimated using $C_p(\mathcal{D})$ and $C_m(\mathcal{D})$. We show experiments across a range of
581 severity values δ_m (**C1**) and also report mode-wise satisfaction rate (**C3**). To explore the role of each
582 coverage metric of failure diversity, we report results for three values of λ for (3), $\lambda = 0, 0.5, 1.0$,
583 where $\lambda = 0, 1$ correspond to the extreme cases of $C = C_p$ and $C = C_m$ respectively (**C2**).

584 C.2 Push-T Simulation

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

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

592 **Experiment 1B (Hyperparameter analysis):** From Table 4, it is evident that there are two main
593 hyper-parameters affecting the performance of our proposed algorithm, radius of coverage, and

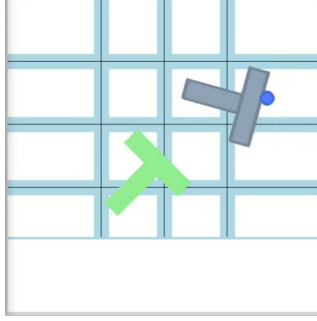


Figure 5: Push-T-Simulation task: The T-region in green corresponds to the target area, and grey denotes location of T-block at a given time, and circle shows the location of end-effector. The white region at the bottom denotes extra space that is present in simulation platform but not accounted for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

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

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 $\lambda = 1.0, 0.0$.

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 .

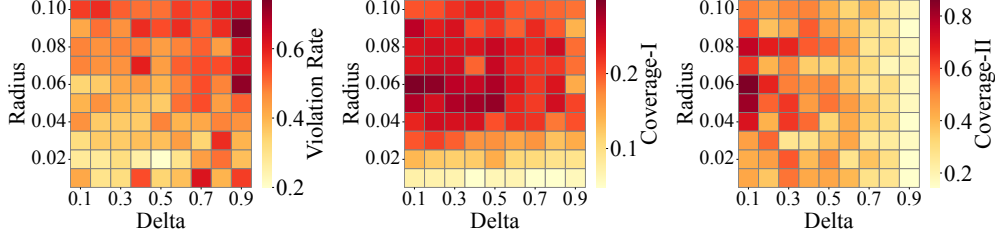


Figure 6: Left to right: Constraint violation rate, Coverage-I and Coverage-II reported for various values of parameter radius r_p and delta δ for (3) with $\lambda = 1$.

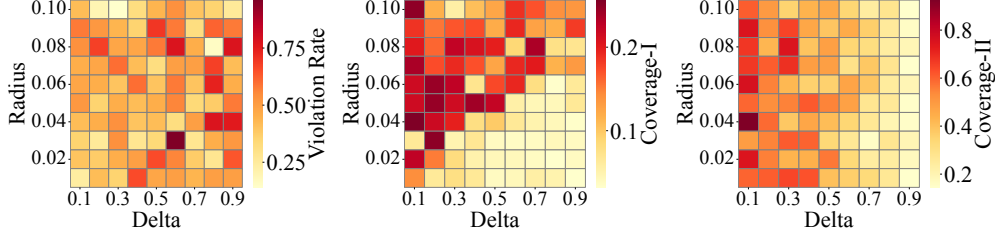


Figure 7: Left to right : Constraint violation rate, Coverage-I and Coverage-II reported for various values of metric radius r_m and delta δ for (3) with $\lambda = 0$.

616 It must also be noted that UCB tends to perform consistently across all β values and is more immune
 617 to hyperparameter sensitivity than our method. These experimental results show that very low values
 618 of r are not suitable for sufficient exploration for learning a good quality surrogate model, but the
 619 discussion on what value of r gives best performance is open-ended. A deeper theoretical analysis
 620 of the impact of hyperparameter on system performance is beyond the scope of this work. However,
 621 in future works, we aim to address optimization of the value of radius to achieve optimal prediction
 622 accuracy.

623 C.3 Push-T Hardware

624 **Problem Setup:** We implement the policy learnt and fine-tuned in simulation using human demon-
 625 strations, to perform the task of pushing the T-block using UR3E collaborative robot arm, by en-
 626 gaging a low-level MoveIt planner [44] to reach a set of goal locations supplied by the Diffusion
 627 Policy [37]. Due to the evident sim-to-real gap in the dynamic systems on which the policy is
 628 trained and deployed on, we observe failures that were previously unseen in simulation.

629 **Failure modes:** We considered two modes of failures here ($M = 2$), **Mode 1** failure correspond to
 630 failure due to joint angle limits and self-collision, which was observed due to parts of the trajectory

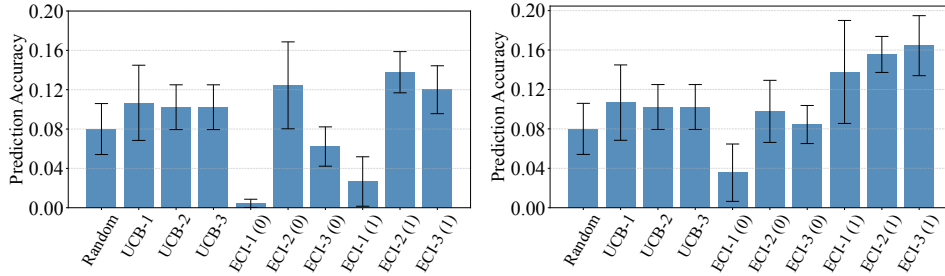


Figure 8: Prediction accuracy for Push-T sim task, histograms and error bars showing mean and standard error values for a test dataset of 25 uniformly generated points for $\delta = 0.9$ (left) and $\delta = 0.3$ (right) respectively.

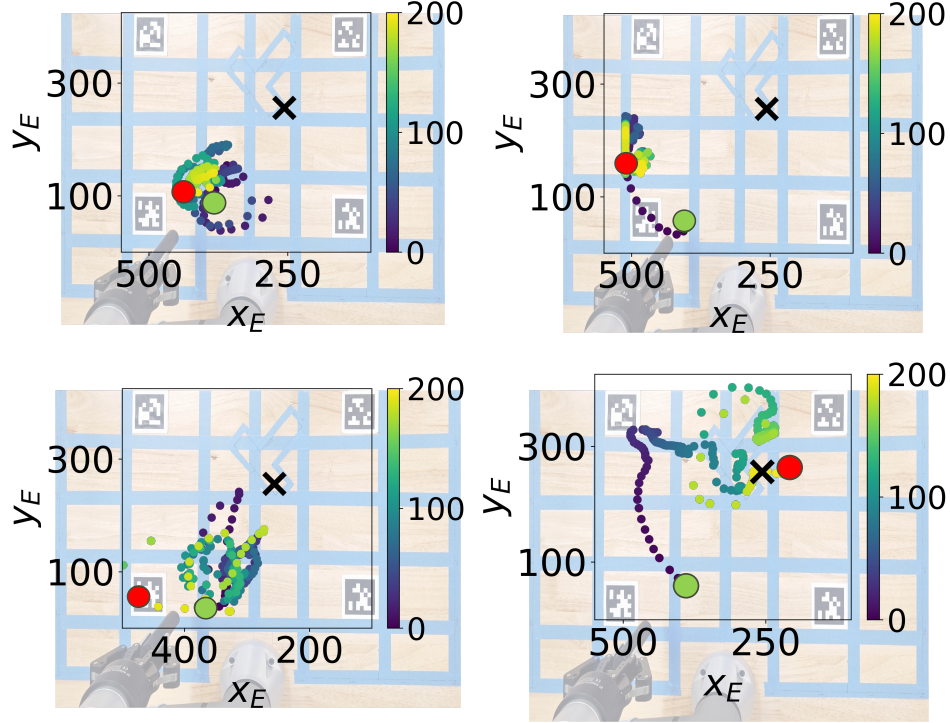


Figure 9: Left to Right: Trajectories from experiments corresponding to failure scenarios due to **Mode 1** (top left), **Mode 2** (top right), **Mode 1 and 2** (bottom left), and **Mode 3** (bottom right) in Push-T hardware task. Colorbar shows time horizon. More details in Appendix C.3

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).

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: 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

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 atleast 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 [47]. 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 $\phi = (b_e, b_l, s)$. 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 [46] 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 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

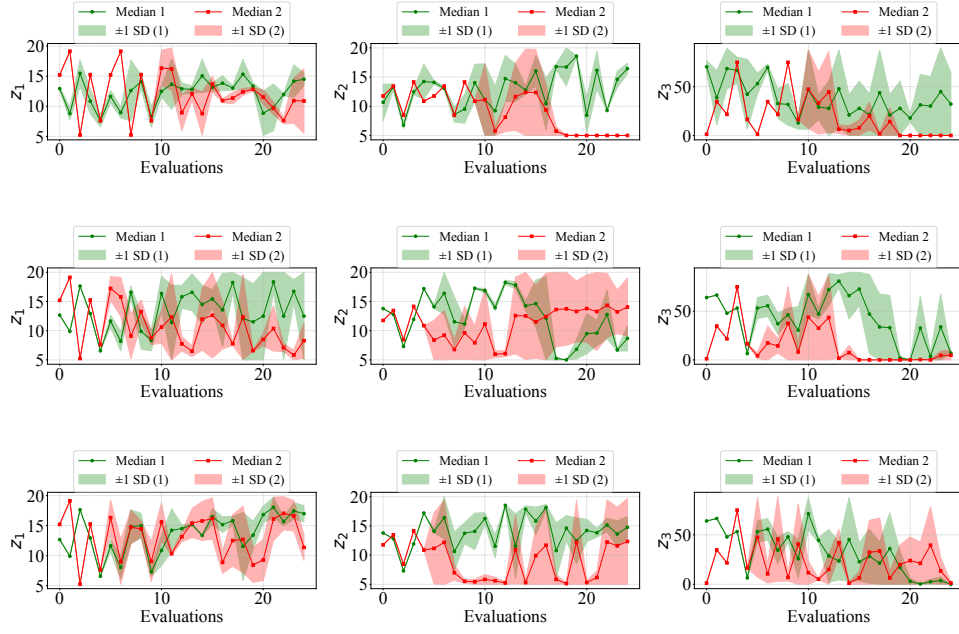


Figure 10: Sampled z values for ECI with $\lambda = 1.0, 0.5, 0.0$ (top to bottom), with $b_e, b_l, s(z_1, z_2, z_3)$ (left to right) for nominal LLM evaluation (red) and hybrid LLM evaluation. Graphs show mean (solid lines) and standard deviation (shaded). The hybrid evaluation results in higher mean 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

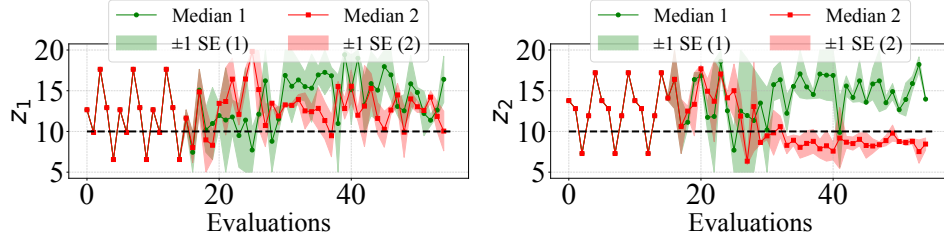


Figure 11: Mean and std deviation for ego and lead braking distance ($b_e - z_1, b_l - z_2$), using ECI with $\lambda = 0.5$ shown for nominal (red) and hybrid evaluation strategies (green) with condition-2 ($b_e > 10, b_l > 10$) for $N = 50$ evaluations. Dashed lines showing the minimum value of b_e,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

697 samples are reduced, which is due to the additional criteria causing feasible parameter space to
698 shrink.

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

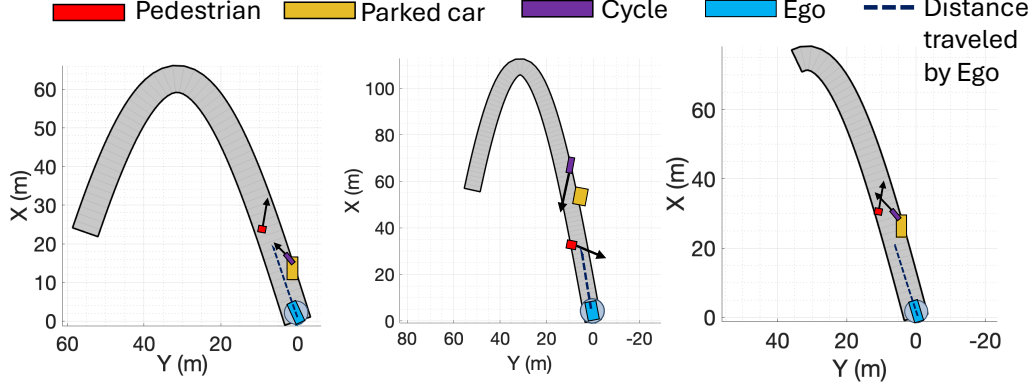


Figure 12: Left to Right: Scenario arrangements corresponding to **Mode 1** and **Mode 2**, and both **Mode 1 and 2**. Arrows indicate the initial direction of velocity vector for each dynamic non-ego agent.

704 C.5 AEB

705 **Scenario description:** This problem consists of testing late and early braking by AEB in a self-
 706 driving context, with a pedestrian and cycle crossing the road, with a parked car on the side. The
 707 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 ampli-
 708 tude and time period of a sinusoidal curve that is used to design the trajectory of the ego vehicle, and
 709 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
 710 r, α cumulatively control the curvature of the road. The road width is fixed $w = 7$, t_{1p}, t_{2p} are used
 711 to specify the initial and final locations for the movement of pedestrian crossing from left to right,
 712 and similarly t_{1c}, t_{2c} for the cycle moving from right to left, given by:

$$\begin{aligned} x &= 40\alpha t_{ik} \\ y &= r \sin(\alpha t_{ik}) \pm 0.5w, \end{aligned} \quad (7)$$

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

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

729 **Choosing δ_1, δ_2 :** From the design of scenario, it can be observed that **Mode 1** occurs in a specific
 730 range of scenario parameters, hence we choose $\delta_1 = 0.5$ to reduce the search volume. We set
 731 $\delta_2 = 0.1$ as we observe a range of values for **Mode 2** failures. With the high dimension of the
 732 space, to encourage better exploration, radius was set to $r_p, r_m = 0.2$. Table 6 shows the results of
 733 all ECI variants against Random sampling. As we can see, $\lambda = 1.0$ has overall good performance,
 734 with all ECI methods performing better than Random sampling in all metrics. We also provide a

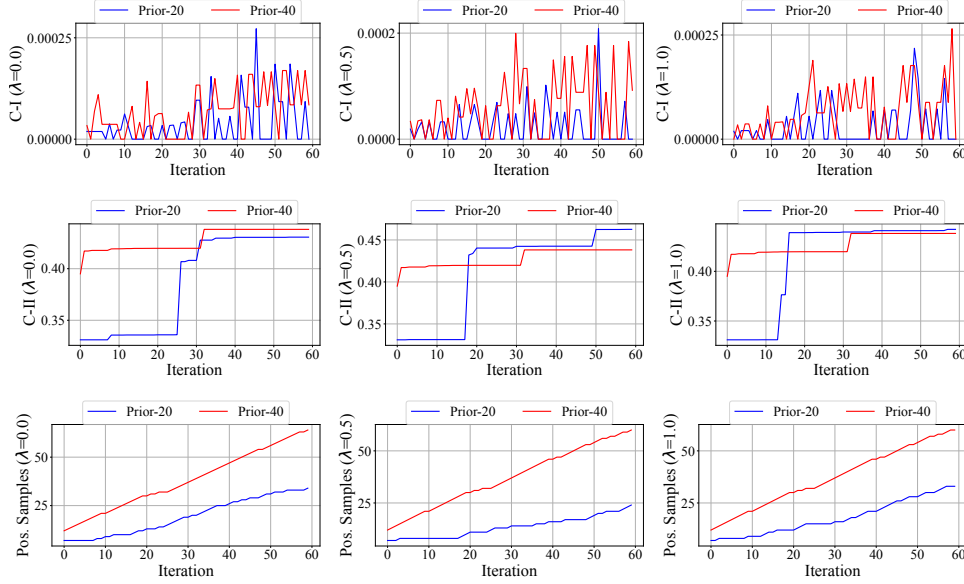


Figure 13: Performance metrics (C-I, C-II, Positive Samples- top to bottom) for ECI with $\lambda = 0, 0.5, 1.0$ (left to right), for an initial model trained with 20 and 40 initial evaluations, proceeded by 60 evaluations using our method.

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.

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

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

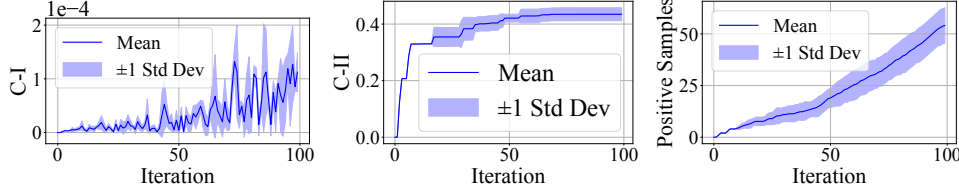


Figure 14: Metrics with experimental budget performance (AEB), $N = 100$, 1 seed. C-I reported as $C-I \times 10^{-4}$.

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. For AEB, the prompt only provides evaluation for Mode 2 failures and the overall evaluation strategy is a hybrid of LLM-based and conditional evaluation. Note that in both cases, a VLM can also be used instead of the proposed evaluation strategy.

C.7.1 LLM-based evaluation for CARLA

We use a pretrained Generative Image to Text model (GiT) [46], 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 provided 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 in front 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 **large distance**. If the list of *objects detected* has an object missing from the *objects to detect* list, look at the reason. The reason can have other components as well, but it can ‘only’ be considered as **bad light** if at least one of the objects was failed to detect strictly due to **bad light**. Similarly, the reason can have other components as well, but it can ‘only’ be considered as **large distance** if the reason contains the phrase ‘far away’. Follow the response instructions while responding.

Response Instructions: Response 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}

776

777 C.7.2 LLM based evaluation for AEB

778 Shown below is the LLM prompt used for **Mode 2** failure evaluation. **Mode 1** failure evaluation is
779 done using the condition $t_{1c} > t_p \wedge (\sum_{t=1}^T \text{AEB}_t > 0)$, for $T \leq 10$. Here AEB_t denotes the value
780 of AEB status at each timestep, and can take values 0,1,2 or 3, corresponding to inactive, partial
781 brakings and full brake respectively. If AEB leads to full brake, early stopping may happen. Text in
782 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 maneuvering) and 1 corresponds to AEB activated.
5. *Map1*: relative initial arrangement 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 atleast one of the timesteps
- 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}

783

784 C.8 Discussion on ECI variants

785 Across experiments, it has been observed that ECI with $\lambda = 1, 0.5$ performs better than $\lambda = 0.0$.
786 This indicates that for coverage, parameter space coverage plays a more important role than just
787 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 repairable, 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 successfully 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.

Additional References

45. Tran D, Ranganath R, Blei DM. The Variational Gaussian Process. *arXiv preprint arXiv:1511.06499*, 2015.
46. Wang J, Yang Z, Hu X, Li L, Lin K, Gan Z, Liu Z, Liu C, Wang L. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022.
47. Fremont, Daniel J., Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit 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.

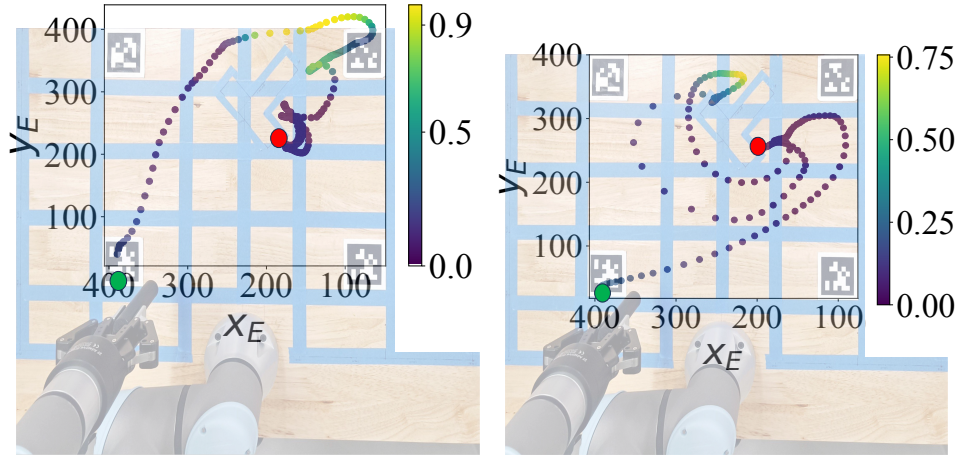


Figure 15: Push-T Hardware: **Mode 3** before repair (left) and after repair (right) respectively. Trajectories of the end effector generated using `pygame` environment shown. As seen from figure on left, trajectory of end effector extends beyond the workspace, leading to failure in real-time experimentation 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 additional 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.