BEYOND ADVERSARIAL EXAMPLES: SAMPLING AND REPAIRING DIVERSE FAILURES WITH RADIUM

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent years have seen large numbers of learning-enabled autonomous systems deployed in the real world. Unfortunately, increased deployment has seen a corresponding increase in accidents involving these systems. We must be able to predict the ways in which these systems might fail and take steps to prevent those failures *before* deployment. Existing tools for failure prediction struggle to search over high-dimensional environmental parameters and provide little guidance on how to mitigate failures once they are discovered. In this paper, we develop a novel framework to efficiently predict failures and propose policy parameter updates to mitigate those failures. By re-framing adversarial optimization as a sequential inference problem, our approach is able to generate a more diverse set of challenging failures, which in turn lead to more robust repaired policies. We propose both gradient-free and gradient-based approaches to solving this inference problem, achieving state-of-the-art performance for policy repair, and we include a theoretical and empirical evaluation of the trade-offs between the two.

1 INTRODUCTION

Assuring the reliability of learning-enabled systems is a challenging open problem, particularly in domains like robotics where learned components are deployed in an uncertain environment. Nevertheless, it is critical to understand how a system will behave when faced with uncertainty; for instance, are there environmental factors that might cause the system to fail, and are there ways to repair the system to avoid such failures in the first place?

Adversarial testing methods propose to solve this problem by searching for counterexamples where the learned system performs poorly, then retraining on those counterexamples (Madry et al., 2018; Salman et al., 2021; Hanselmann et al., 2022; Ding et al., 2020; Corso et al., 2019; Wang et al., 2021). Adversarial methods are typically greedy, using gradient-based or gradient-free optimization to seek out the most severe or most likely failures, but this leads to a critical issue: a loss of diversity in the counterexamples. If the counterexamples over-represent certain cases, then the retraining process will over-fit to those cases, reducing robustness.

The challenge of finding diverse counterexamples has motivated recent work in *rare-event prediction* using methods like Markov Chain Monte Carlo (MCMC) and importance sampling (Sinha et al., 2020; Delecki et al., 2023; Zhou et al., 2021; O' Kelly et al., 2018; Corso et al., 2019). Unfortunately, existing rare-event prediction methods, particularly importance sampling, suffer from the curse of dimensionality as rare events become sparse in high-dimensional search spaces (Betancourt, 2017). Moreover, existing failure prediction methods provide little guidance on how to update the policy once failures have been discovered.

In this paper, we aim to close the gap between adversarial training and rare-event prediction with RADIUM: a framework that simultaneously predicts diverse, challenging failures and updates the control policy to repair those failures, as shown in Fig. 1. To efficiently explore the failure space, we start with highly likely failures and gradually expand our search to more severe rare counterexamples, continuously repairing the policy as the failure distribution shifts. We make the following contributions:

1. We reframe adversarial optimization as a sequential inference problem, leading to a novel framework for predicting and repairing a diverse set of failures.

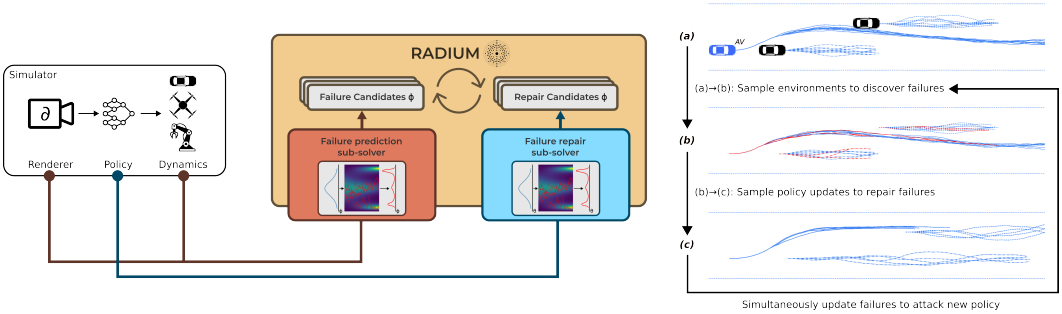

Figure 1: An overview of our approach for closed-loop rare-event prediction, which efficiently predicts and repairs failures in autonomous systems. Our framework alternates between failure prediction and repair sub-solvers, which use a simulated environment to efficiently sample from the distributions (2) and (3). We use differentiable rendering and simulation to accelerate our method with end-to-end gradients, but we also propose a gradient-free implementation.

2. We develop both gradient-free and gradient-based variants of our framework, relying on differentiable simulation and rendering for the latter, and evaluate the performance trade-offs when scaling to high-dimensional problems.
3. We provide a theoretical analysis of our sequential inference framework, proving correctness and asymptotic convergence for both variants. For the gradient-based variant, we also provide finite-sample convergence rates in a restricted setting.

In contrast with existing methods, many of which are specialized to autonomous driving (Ding et al., 2020; Riedmaier et al., 2020; Zhong et al., 2022), our method is general purpose and requires minimal reward shaping. Unlike previous gradient-based methods for rare-event prediction (Sinha et al., 2020; Dawson and Fan, 2023), which do not support systems with vision in the loop, our use of differentiable rendering allows us to test and repair more complex autonomous systems. Empirically, experiments on a range of robotics tasks show that our method outperforms existing methods (gradient-based adversarial training and adversarial RL) for failure prediction and policy repair, generating repaired policies with 2-3x lower failure rates. We include code for our method in the supplementary materials and plan to open-source our framework.

1.1 Related work

Multiple prior works deal with the problem of failure mode prediction for autonomous systems, often in the context of autonomous vehicles, or AVs (Xu et al., 2022; Riedmaier et al., 2020; O' Kelly et al., 2018; Corso et al., 2019; Wang et al., 2021; Sun et al., 2021; Zhong et al., 2022; Corso and Kochenderfer, 2020; Zhang et al., 2022; Hanselmann et al., 2022; Zhong et al., 2022)). However, the large majority of these methods rely on black-box methods that struggle to scale to high-dimensional search spaces. Since our goal is to not only predict but also repair failures (thus searching over high-dimensional policy space), we are interested instead in gradient-based methods with improved scalability. Unfortunately, existing gradient-based failure prediction methods cannot handle systems with vision in the loop. Zhong et al. (2022) uses gradients through temporal logic driving rules to generate test cases, but does not consider agents with vision-based policies. Hanselmann et al. (2022) find that skipping the rendering step during backpropagation is sufficient for gradient-based failure-case generation, but this method is not able to repair visual-feedback policies. Sinha et al. (2020) learn a proxy model for the end-to-end dynamics, including a vision-based policy, but end-to-end proxy modeling is generally not applicable to rare-event generation, since it is difficult to include enough failure examples in the training data for the proxy model.

2 Problem

We begin by considering a general autonomous system that receives observations $o \in \mathcal{O}$ and makes control decisions using a learned control policy $\pi_\theta : \mathcal{O} \mapsto \mathcal{A}$, where θ are the parameters of that policy. This agent operates in closed loop with an environment with dynamics $f_\phi : \mathcal{X} \mapsto \mathcal{X}$ and

rendering function $R_\phi : \mathcal{X} \mapsto \mathcal{O}$, where uncertainty in the environment manifests as uncertain parameters ϕ with probability density $p_{\phi,0}$. Without loss of generality, we assume that randomness in the environment has been "factored out" into ϕ (so that f and R are deterministic given ϕ). We assign a cost $J(\theta, \phi)$ to a pair of policy and environmental parameters by rolling out for a fixed T-step horizon:

$$o_t = R_\phi(x_t); \quad x_{t+1} = f_\phi(x_t, \pi_\theta(o_t)); \quad J(\theta, \phi) = J(x_0, \ldots, x_T) \tag{1}$$

In this context, **failure prediction** involves finding environmental parameters ϕ that induce high cost for given policy parameters θ, i.e. finding multiple solutions $\phi^*(\theta) = \text{find}_\phi J(\theta, \phi) \geq J^*$ for failure threshold J^*, while **failure repair** involves modifying the initial policy parameters θ_0 to achieve low costs despite possible variation in ϕ, i.e. finding a nearby $\theta^* = \min_\theta ||\theta_0 - \theta||^2$ s.t. $E_\phi[J(\theta, \phi)] \leq J^*$.

2.1 Failure prediction

Simply optimizing for the highest-severity or most-likely failure will give an incomplete picture of the system's performance and lead to more conservative behavior. Instead, we balance the prior likelihood of a disturbance with the severity of the induced failure by sampling failures from the pseudo-posterior

$$p_{\text{failure}}(\phi; \theta) \propto p_{\phi,0}(\phi) e^{-[J^* - J(\theta,\phi)]_+} \tag{2}$$

where J^* is the cost threshold for a failure event and $[\cdot]_+$ is the exponential linear unit. Intuitively, we can interpret this likelihood as a posterior over environmental parameters ϕ conditioned on a failure occurring (Sinha et al., 2020; Zhou et al., 2021; Ma et al., 2019; Dawson and Fan, 2023). By framing the search for failures prediction as a sampling problem, rather than the traditional adversarial optimization, we gain a number of advantages. First, we are able to generate a more diverse set of failure modes than would be discovered by an extremum-seeking approach, as shown by our experiments in Section 5. Second, we are able to draw on a rich literature of theoretically well-motivated sampling algorithms to develop our approach, as we discuss in Sections 3 and 4.

2.2 Failure repair

This is not the first paper to take a sampling-based approach to failure prediction; for example, O' Kelly et al. (2018), Sinha et al. (2020), and Zhou et al. (2021) also approach failure prediction using this lens. Our insight is that this sampling framework can be extended to not only predict failures but also repair the underlying policy, thus mitigating the impact of the failure. Given initial policy parameters θ_0 and a population of anticipated failure modes $\phi_1, \ldots, \phi_n$, we can increase the robustness of our policy by sampling from a corresponding repair pseudo-posterior, similar to Eq. (2),

$$p_{\text{repair}}(\theta; \theta_0, \phi_1, \ldots, \phi_n) \propto p_{\theta,0}(\theta; \theta_0) e^{-\sum_{\phi_i} [J(\theta,\phi_i) - J^*]_+ / n} \tag{3}$$

where the prior likelihood $p_{\theta,0}$ regularizes the search for repaired policies that are close to the original policy. Intuitively, this distribution of repaired policies can be seen as a posterior over policies conditioned on the event that a failure *does not* occur in the given scenarios. Sampling from this posterior can be seen as a form of regularized re-training on the set of predicted failures, since maximizing the log of (3) is equivalent to minimizing the empirical risk $\sum_{\phi_i} [J(\theta, \phi_i) - J^*]_+ / n$ with regularization $||\theta - \theta_0||_2^2$ (assuming a Gaussian prior). This connection helps motivate our use of (3), but we find empirically in Section 5 that the increased diversity from sampling rather than policy gradient optimization yields better policies in practice.

3 Approach

Previous works have shown that sampling from a failure distribution like Eq. (2) can generate novel failures (Zhou et al., 2021; Sinha et al., 2020; Delecki et al., 2023), but several challenges have prevented these works from considering end-to-end policy repair as well. Our main contribution is a framework for resolving these challenges and enabling simultaneous failure prediction and repair, which we call **RADIUM** (Robustness via Adversarial Diversity using MCMC, illustrated in Fig. 1). We have designed this framework to take advantage of problem structure (e.g. differentiability) when possible, but we provide the ability to swap gradient-based subsolvers for gradient-free ones when needed, and we include a discussion of the associated trade-offs.

Challenge 1: Distribution shift during retraining Previous methods have proposed generating failure examples for use in retraining, but there is an inherent risk of distribution shift when doing so. Once we repair the policy, previously-predicted failures become stale and are no longer useful for verification (i.e. the distribution of likely failures has shifted). In the worst case, this can lead to overconfidence if the system claims to have repaired all previously-discovered failures while remaining vulnerable to other failures. To address this issue, we interleave failure and repair steps to continuously update the set of predicted failures as we repair the policy, creating an adversarial sampling process that generates a robust repaired policy along with a set of salient failure cases.

Challenge 2: Exploring diverse failure modes Traditional methods like Markov chain Monte Carlo (MCMC) are able to sample from non-normalized likelihoods like (2) and (3), but they struggle to fully explore the search space when the likelihood is highly multi-modal. To mitigate this issue, we take inspiration from the recent success of diffusion processes (Song et al., 2023; Zhong et al., 2022) and sequential Monte Carlo algorithms (Rubino and Tuffin, 2009) that interpolate between an easy-to-sample prior distribution and a multi-modal target distribution. Instead of sampling directly from the posterior, we begin by sampling from the unimodal, easy-to-sample prior and then smoothly interpolate to the posterior distributions (2)-(3). This process yields the tempered likelihood functions:

$$\tilde{p}_{\text{failure}} \propto p_{\phi,0}(\phi) e^{-\tau[J^* - J(\theta,\phi)]_+} \quad (4) \qquad \tilde{p}_{\text{repair}} \propto p_{\theta,0}(\theta,\theta_0) e^{-\frac{\tau}{n}\sum_{\phi_i}[J(\theta,\phi_i)-J^*]_+} \quad (5)$$

where the tempering parameter τ is smoothly varied from 0 to 1. When $\tau = 0$, this is equivalent to sampling from the prior distributions, and when $\tau \to 1$ we recover the full posteriors (2)-(3). This tempering process reduces the risk of overfitting to one particular mode of the failure distribution and encourages even exploration of the failure space.

Challenge 3: Efficiently sampling in high dimension Previous works have proposed a wide variety of sampling algorithms that might be used as sub-solvers in our framework, including MCMC methods like random-walk Metropolis-Hastings (RMH; Hastings (1970)), Hamiltonian Monte Carlo (HMC; Neal (2011)), and the Metropolis-adjusted Langevin algorithm (MALA; Bresag (1994)), variational inference methods like Stein Variational Gradient Descent (SVGD; Liu and Wang (2016)), and other black-box methods like adaptive importance sampling (O' Kelly et al., 2018). RADIUM is able to use any of these sampling methods as sub-solvers for either the prediction or repair. Generally, these sampling methods can be classified as either gradient-free or gradient-based. Theoretical and empirical evidence suggests that gradient-based methods can enjoy faster mixing time in high dimensions on certain classes of sufficiently smooth (but non-convex) problems (Ma et al., 2019), but autonomous systems with visual feedback have historically been treated as black-boxes due to an inability to backpropagate through the rendering step (Zhou et al., 2021; O' Kelly et al., 2018; Sinha et al., 2020). To enable the use of gradient-based samplers in RADIUM, we draw upon recent advances in differentiable simulation and rendering (Hu et al., 2019; Le Lidec et al., 2021) provide end-to-end gradients. In Sections 4 and 5, we provide theoretical and empirical evidence of a performance advantage for gradient-based samplers, but in order to make RADIUM compatible with existing non-differentiable simulators we also conduct experiments where RADIUM uses gradient-free sampling subroutines.

3.1 RADIUM

Pseudocode for RADIUM is provided in Algorithm 1. The algorithm maintains separate populations of candidate repaired policies $[\theta_1, \ldots, \theta_n]$ and failures $[\phi_1, \ldots, \phi_n]$ that are updated over N sampling rounds. In each round, we sample a set of new candidate policies from the repair generating process (5), then sample a new set of failures that attack the current population of policies. In practice, we average the tempered failure log probability (4) over the population of candidate designs, which results in a smoother distribution.

RADIUM supports a wide range of subroutines for sampling candidate failures and repaired policies. In our experiments and the provided implementation, we include RMH and MALA (gradient-free and gradient-based MCMC algorithms, respectively); we choose these particular methods to provide a direct comparison between similar algorithms with and without gradients (MALA reduces to RMH when the gradient is zero).

Algorithm 1: RADIUM: Robustness via Adversarial Diversity using MCMC

Input: N rounds, m steps per round, stepsize λ, population size n, tempering rate α, sampling algorithm (e.g. MALA, RMH, HMC, ...)

1 Sample initial failures and policies using priors: $[\phi_1, \ldots, \phi_n]_0 \overset{\text{iid}}{\sim} p_{\phi,0}$, $[\theta_1, \ldots, \theta_n]_0 \overset{\text{iid}}{\sim} p_{\theta,0}$;
2 **for** $i = 1, \ldots, N$ **do**
3 $\quad \tau \leftarrow 1 - e^{-\alpha i/N}$; `// Tempering schedule`
4 $\quad$ Sample $[\theta_1, \ldots, \theta_n]_i \overset{\text{iid}}{\sim}$ (5); `// Sample repaired policies`
5 $\quad$ Sample $[\phi_1, \ldots, \phi_n]_i \overset{\text{iid}}{\sim}$ (4); `// Generate failures attacking` θ_i^*
6 **end**

Return: Repaired policy $\theta_N^* = \arg\max_i$ (5) and failures $[\phi_1, \ldots, \phi_n]_N$ attacking that policy.

4 THEORETICAL ANALYSIS

The iterative adversarial sampling process defined in Alg. 1 raises a few theoretical questions. First, when can we expect the individual sampling steps on lines 4 and 5 to converge, and under what conditions might we expect a gradient-based sampling sub-routine to converge faster than a gradient-free one? Second, assuming that these individual samplers converge, what sort of policies will result from the adversarial sampling loop in Alg. 1?

Convergence and gradient acceleration RADIUM inherits the asymptotic convergence guarantees of the particular subsolvers used for each sampling step. For example, when using an MCMC sampler, so long as that sampler can propose arbitrarily large steps with non-zero probability and satisfies detailed balance (e.g. through the use of a Metropolis adjustment), then the sampler will produce samples asymptotically close to the target sampling distribution. Since the conditions for asymptotic convergence of MCMC samplers are relatively weak Hastings (1970), it is more interesting to ask about finite-sample convergence rates; in particular, under what conditions can we expect gradient-based samplers like MALA to accelerate convergence to the target distribution?

In many robotics problems, even when analytical gradients are available, it is unclear whether these gradients are useful for optimization (i.e. low empirical bias and variance; Suh et al. (2022)). Here, we build on recent theoretical results by Ma et al. (2019) to provide sufficient conditions for fast, polynomial-time convergence of gradient-based samplers in our setting.

Theorem 4.1. *Let J be a L-Lipschitz smooth cost function (i.e. ∇J is L-Lipschitz continuous), let the log prior distributions $\log p_{\phi,0}$ and $\log p_{\theta,0}$ be Lipschitz smooth everywhere and m-strongly convex outside a ball of finite radius R, and let $d = \max(\dim\theta, \dim\phi)$ be the dimension of the search space. If $m > L$, then MALA with appropriate step size will yield samples within ϵ total variation distance of the target distributions* (4) *and* (5) *with total number of sampling steps $\leq \widetilde{\mathcal{O}}\left(d^2 \ln \frac{1}{\epsilon}\right)$.*

A proof is given in the appendix, which also provides the required step size for the MALA sampler. The key idea of the proof is to rely on the log-concavity of the prior distributions to dominate the non-convexity of the cost function sufficiently far from the central modes. Theorem 4.1 requires smoothness assumptions on the cost; we recognize that this assumption is difficult to verify in practice and does not hold in certain domains (notably when rigid body contact is involved). However, in the problems we consider it is possible to smooth both the renderer and scene representation (by blurring the scene and using smooth signed distance functions), thus smoothing the gradients of J. The smoothness and convexity conditions hold for many common prior distributions, such as Gaussian and smoothed uniform distributions.

Adversarial Joint Distribution Even if the samplers for both policy and environmental parameters converge within each round of Alg. 1, it is not clear what will be the effect of running these samplers repeatedly in an adversarial manner. Our next theoretical result defines the joint distribution of θ and ϕ as a result of this adversarial sampling loop. To simplify the theoretical analysis, we consider the case when population size $n = 1$, and we replace the smooth ELU with a ReLU in (2) and (3).

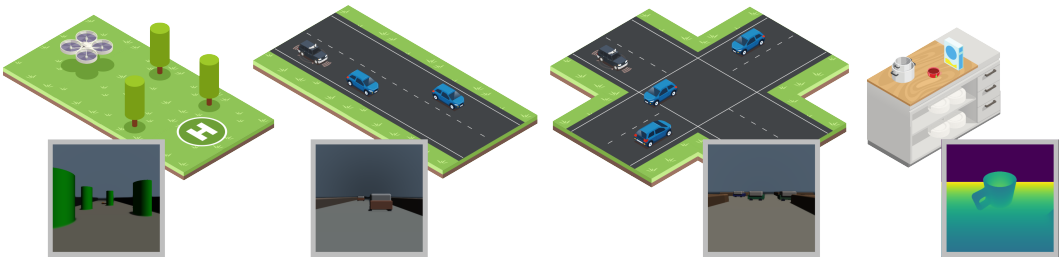

Figure 2: Environments for drone navigation, AV control, and household object manipulation. Our method is applicable in a wide range of settings and is agnostic to the type of learned policy used. Inset: a robot's-eye-view of each scene rendered using our differentiable renderer.

Theorem 4.2. *The iterative adversarial sampling procedure in Alg. 1 yields policies drawn from a marginal distribution with density function*

$$f_\theta(\theta^*) = p_{\theta,0}(\theta^*)\left(\frac{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}|J(\theta^*,\phi)\leq J^*\right]}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]}+\frac{\mathbb{P}[J(\theta^*,\phi)>J^*]}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]}\right) \tag{6}$$

where $\mathbb{P}(J(\theta^*,\phi)>J^* = \mathbb{E}_{\phi\sim p_{\phi,0}}[\mathbb{1}(J(\theta^*,\phi)\geq J^*)]$ *is the probability of failure when* ϕ *is sampled from the prior distribution.*

The proof is included in the appendix and follows from the Hammersley-Clifford theorem for Gibbs samplers (Robert and Casella, 2004). The first term in the parenthesis in (6) is bounded above by 1 and maximized when the policy does not experience failure (in which case the conditional and unconditional expectations will be equal). The numerator of the second term bounded $[0, 1]$, while the denominator grows exponentially large when a failure occurs. As a result, the marginal distribution of θ^* assigns higher probability (relative to the prior) for policies that avoid failure.

5 Experiments

Next, we evaluate our proposed method using experiments in a range of robotics environments: autonomous vehicle control, visual navigation of an aerial robot in a cluttered environment, and grasp identification for robotic manipulation. We compare the the performance and scalability of both variants of RADIUM (with gradient-free and gradient-based MCMC sampling subroutines) with state-of-the-art adversarial training methods.

5.1 Experimental design

Environments We consider three distinct environments with two tasks in each, as shown in Fig. 2. *AV (highway):* An autonomous vehicle must overtake two other vehicles. *AV (intersection):* the autonomous vehicle must navigate an uncontrolled intersection with crossing traffic. In both tasks, the actions of the non-ego vehicles are uncertain, and the AV observes RGBd images from a front-facing camera as well as its speed. *Drone (static):* A drone must safely navigate through a cluttered environment in windy conditions. There is uncertainty in the wind speed and location of all obstacles. *Drone (dynamic):* the same but the obstacles move with uncertain velocities. Initial policies θ_0 for drone and intersection environments were pretrained using behavior cloning, and policies for the highway environment were pretrained using PPO Schulman et al. (2017). *Grasp (box/mug):* a robot must locate and grasp an object using a depth image of the scene. There is uncertainty in the location of the objects and in the location of a nearby distractor object. The grasp detector is trained with labels of ground-truth grasp affordances.

The dimension of the failure space is 20 for the highway task, 30 for the intersection task, 12 for the static drone task, 22 for the dynamic drone task, and 4 for the grasping tasks. The dimension of the policy space is 64k for the highway and intersection tasks, 84k for the drone tasks, and 266k for the grasping tasks. More details on each task are in the supplementary material, including cost functions and failure thresholds.

Table 1: Failure rate of policies optimized using different adversarial methods on a test set of 1,000 ϕ sampled i.i.d. from the prior. θ_0 indicates the performance of the original policy prior to retraining. Mean and standard deviation (subscript) across 4 seeds are shown; lower is better.

	Task					
	AV (hw.)	AV (int.)	Drone (st.)	Drone (dyn.)	Grasp (box)	Grasp (mug)
θ_0	0.11	0.92	0.17	0.30	0.0088	0.011
GD	$0.44_{0.40}$	$1.0_{0.0}$	$0.27_{0.46}$	$0.26_{0.17}$	$0.0088_{0.0012}$	$0.011_{0.0025}$
L2C	$0.12_{0.018}$	$0.82_{0.20}$	$0.17_{0.013}$	$0.27_{0.013}$	$0.0088_{0.0012}$	$0.011_{0.0025}$
R_0	$\mathbf{0.056}_{0.10}$	$0.51_{0.34}$	$0.21_{0.013}$	$0.31_{0.13}$	$0.0040_{0.0034}$	$0.0045_{0.0061}$
R_1	$\mathbf{0.056}_{0.068}$	$\mathbf{0.43}_{0.38}$	$\mathbf{0.078}_{0.029}$	$\mathbf{0.16}_{0.070}$	$\mathbf{0.0028}_{0.0038}$	$\mathbf{0.0035}_{0.0044}$

Simulator Since we require a differentiable renderer and simulation engine for our work, we were not able to use an off-the-shelf simulator like CARLA (Dosovitskiy et al., 2017). Instead, we wrote our own simulator and basic differentiable renderer using JAX.

Baselines We compare with two representative baselines. *Gradient descent (GD)* predicts failure modes that locally maximize the posterior likelihood; this is meant to represent any number of gradient-based adversarial optimization schemes (Hanselmann et al., 2022). *Learning to collide (L2C)* uses black-box optimization (REINFORCE) to search for failure cases (Ding et al., 2020). We denote the gradient-free and gradient-based variants of RADIUM as R_0 and R_1, respectively. All methods were run on the same GPU model with the same sample budget for each task. Hyperparameters for all experiments are given in the appendix.

Metrics To measure the robustness of the optimized policies, we report the failure rate (FR) on a test set of 1,000 i.i.d. samples of ϕ from the prior $p_{\phi,0}$. To assess the quality of the counterexamples, we compare the difference between the failure rate predicted on the test set with that predicted on the counterexamples (a more accurate estimate is better). Finally, for each task, we report the time required for simulating a rollout both with and without reverse-mode differentiation.

5.2 Results

Table 1 shows the failure rate of policies optimized using each method, along with the failure rate of the policy prior to any adversarial optimization and repair, and Fig. 3 shows examples of failure cases and repaired policies generated using R_1. We observe a wide range of prior failure rates across tasks due to the difficulty of pre-training successful policies in all domains; this range of prior failure rates allows us to compare tasks where failures are easy to find with those where failures are more rare. Across tasks, we find that the gradient-based variant of RADIUM consistently yields the most robust repaired policy. In some cases, we find that adversarial training using GD or R_0 can lead to catastrophic forgetting where overfitting to the counterexamples leads to a repaired policy with worse performance than the original, but we observe that R_1 avoids this issue.

Table 2 compares the failure rate on the counterexamples found by each method with the failure rate seen on the test set, indicating how adversarial the counterexamples are. As expected, we find that all methods find counterexamples that are at least as difficult as those sampled randomly from the prior.

Table 3 reports the time required to simulate a single rollout for each task, both with and without AD. We observe that computing derivatives through the simulator and renderer results in a slowdown of 2-5x relative to a simulation without derivatives, and gradient-based methods GD and R_1 typically run 2x more slowly than gradient-free methods L2C and R_0.

5.3 Discussion

Comparing these adversarial optimization methods, we find that our proposed gradient-based method, R_1, consistently finds more robust repaired policies than both gradient-free and gradient-based baselines. This pattern holds when failures are relatively easy to find (AV int.) and when failures are relatively rare (AV hw. and grasping). We attribute the improved performance of our method to two

Table 2: Mean relative difference between failure rate estimated using counterexamples and failure rate on a test set of 1,000 ϕ sampled i.i.d. from the prior. Mean and standard deviation (subscript) across 4 seeds are shown; positive indicates an overestimate.

	Task					
	AV (hw.)	AV (int.)	Drone (st.)	Drone (dyn.)	Grasp (box)	Grasp (mug)
GD	$-0.09\%_{0.31}$	$0.00\%^{*}_{0.00}$	$-0.09\%_{0.13}$	$-0.05\%_{0.71}$	$-1.28\%_{0.25}$	$-0.80\%_{0.10}$
L2C	$0.24\%_{1.51}$	$0.06\%_{0.18}$	$0.78\%_{0.96}$	$-0.44\%_{0.65}$	$-1.28\%_{0.25}$	$-0.80\%_{0.10}$
R_0	$-0.10\%_{0.11}$	$-0.02\%_{0.01}$	$2.10\%_{0.64}$	$0.95\%_{0.79}$	$-1.12\%_{1.33}$	$-0.89\%_{0.65}$
R_1	$-0.10\%_{0.80}$	$-0.22\%_{0.45}$	$-0.36\%_{1.14}$	$1.21\%_{0.56}$	$-1.25\%_{1.35}$	$-0.79\%_{0.93}$

* GD fails on 100% of test cases and counterexamples, with trivially zero error.

Table 3: Time required for simulating a rollout with and without autodiff (AD) for each task. Average and standard deviation (subscript) reported across 100 trials on an NVIDIA RTX A4000.

	Task				
	AV (hw.)	AV (int.)	Drone (sts)	Drone (dyn.)	Grasp (all)
Without AD	$0.70_{0.003}$ s	$2.22_{0.01}$ s	$0.39_{0.002}$ s	$0.39_{0.001}$ s	$0.0045_{5.1\times10^{-5}}$ s
With AD	$1.72_{0.003}$ s	$6.65_{0.14}$ s	$1.77_{0.06}$ s	$1.83_{0.04}$ s	$0.0049_{3.8\times10^{-5}}$

factors. Relative to gradient-free methods, R_1 is able to more efficiently explore the space of failures and repaired policies by using end-to-end gradients of the sampling distribution; since the space of policy repairs can be quite high, this is particularly helpful during the repair step. In particular, we observed that the gradient-free method L2C did not substantially change policy performance in any task; this is likely because this method is based on REINFORCE and requires a much larger sample budget than that used in these experiments (we limited all methods to 1000 total simulator queries).

Relative to adversarial gradient-based optimization, RADIUM benefits from its posterior sampling-based approach: not only does sampling improve the diversity of the counterexamples, leading to a better repaired policy, but MCMC is also more robust than GD to poorly-conditioned gradients, since it is able to reject large repair steps that decrease the performance of the policy, even when we apply the same gradient normalization to both GD and R_1.

Of course, there are two significant downsides to relying on gradients. First, automatic differentiation increases the computational cost of each simulation query (typically by 2-5x, as shown in Table 3), which in turn caused GD and R_1 to run 2x more slowly than L2C and R_0 on the same hardware. Despite this computational cost, our results in Table 1 indicate that gradient-based methods like R_1 are able to find higher-quality solutions, making this trade-off worthwhile in some contexts. The second downside is the lack of widely available differentiable simulation environments, particularly including differentiable rendering. In this work, we developed our own simple simulation and renderer, but we hope that increased interest in downstream uses for end-to-end gradients (like R_1) motivates further development of differentiable photorealistic renderers (Jakob et al., 2022).

6 Conclusion

In this paper, we have proposed a novel framework for predicting the ways in which a learning-based system might fail and repairing the learned policy to preemptively mitigate those failures. Our framework reframes traditional adversarial optimization as an iterative sampling process to prioritize diversity in the predicted failures, yielding more robust repaired policies and avoiding overfitting to a narrow set of predicted failures. We present both gradient-free and gradient-based variants of our framework, and we examine the tradeoff in computation time and solution quality between these methods, using end-to-end gradients through differentiable simulation and rendering for the gradient-based variant. We find that our method yields more robust repaired policies than prior methods on a range of problems.

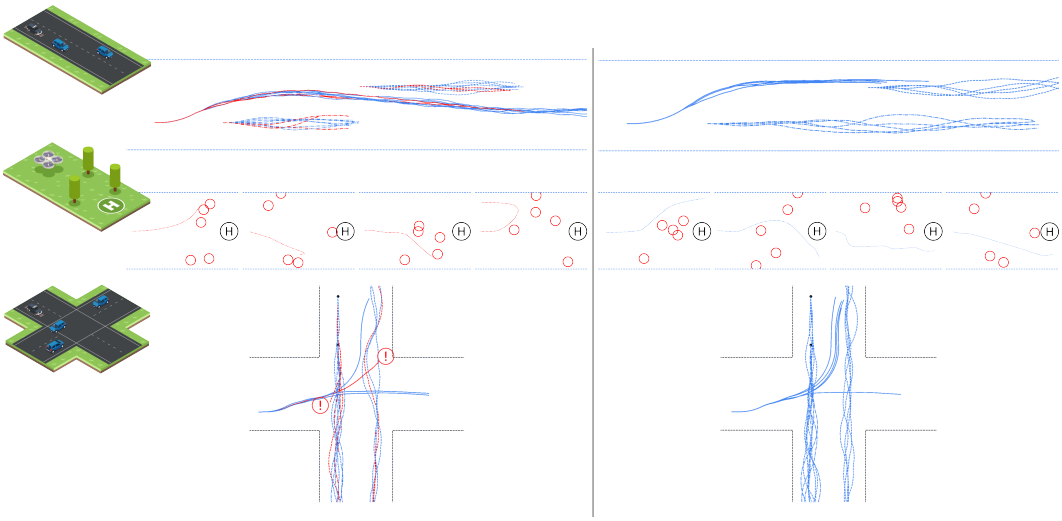

Figure 3: Examples of failure cases (left) and repaired policies (right) generated using our method.

6.1 Limitations & future work

The main limitation of our approach is that it requires a simulator with enough fidelity to predict the failures of interest; moreover, the gradient-based version of our framework requires a differentiable simulator, which is even more difficult to come by. This means that our method cannot predict any failures not modeled by the simulator (e.g. loose cables). In future, we hope to explore algorithms for combining accelerated testing in simulation with limited, but higher fidelity, testing in hardware.

In future work, we also hope to integrate with emerging photorealistic differentiable renderers like Mitsuba (Jakob et al., 2022). As of this writing, ease of use and interoperability with machine learning libraries is still difficult with these renderers, but we hope that additional engineering effort in this area will result in an easy-to-use, high-quality differentiable renderer that can be integrated with existing robotics simulation frameworks.

7 Ethics Statement

Verification methods like ours are inherently dual-use: an algorithm that an engineer can use to test her design can also be used by a malicious actor to break an already-deployed system. We mitigate this risk in two ways. First, our approach requires a simulator model of the system under test; without access to this model, bad actors cannot use our method to generate attacks. Second, our failure prediction method comes with a failure repair method attached, so system designers can harden their systems against any attacks generated using our method. On balance, we hope that our method will help engineers fine-tune their designs to mitigate the risk of failure prior to deployment, enabling them to more confidently design, debug, and deploy autonomous systems.

8 Reproducibility Statement

We have included code to reproduce our experiments in the supplementary materials, including the specific commands and seeds used to generate our results.

References

Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. January 2017. doi: 10.48550/arxiv.1701.02434.

Julian Bresag. Comments on U. Grenadier, M. Miller, "Representations of Knowledge in Complex Systems". *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(4):549–603, 1994. ISSN 0035-9246.

Anthony Corso and Mykel J. Kochenderfer. Interpretable Safety Validation for Autonomous Vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, September 2020. doi: 10.1109/ITSC45102.2020.9294490.

Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168, October 2019. doi: 10.1109/ITSC.2019.8917242.

Charles Dawson and Chuchu Fan. A Bayesian approach to breaking things: Efficiently predicting and repairing failure modes via sampling. In *7th Annual Conference on Robot Learning*, August 2023.

Harrison Delecki, Anthony Corso, and Mykel Kochenderfer. Model-based Validation as Probabilistic Inference. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, pages 825–837. PMLR, June 2023.

Wenhao Ding, Baiming Chen, Minjun Xu, and Ding Zhao. Learning to Collide: An Adaptive Safety-Critical Scenarios Generating Method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2243–2250, October 2020. doi: 10.1109/IROS45743.2020.9340696.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas 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, October 2022. Springer-Verlag. ISBN 978-3-031-19838-0. doi: 10.1007/978-3-031-19839-7_20.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. ISSN 0006-3444. doi: 10.1093/biomet/57.1.97.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Fredo Durand. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations*, December 2019.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.Jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022. doi: 10.1145/3528223.3530099.

Quentin Le Lidec, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Differentiable rendering with perturbed optimizers. In *Advances in Neural Information Processing Systems*, volume 34, pages 20398–20409. Curran Associates, Inc., 2021.

Qiang Liu and Dilin Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yi An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I. Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences of the United States of America*, 116(42):20881–20885, October 2019. ISSN 10916490. doi: 10.1073/PNAS.1820003116/-/DCSUPPLEMENTAL.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, February 2018.

Radford M. Neal. MCMC Using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011. ISBN 978-0-429-13850-8.

Matthew O' Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John 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.

Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. Survey on Scenario-Based Safety Assessment of Automated Vehicles. *IEEE Access*, 8:87456–87477, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2993730.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York, NY, 2004. ISBN 978-1-4419-1939-7 978-1-4757-4145-2. doi: 10.1007/978-1-4757-4145-2.

Gerardo Rubino and Bruno Tuffin. Introduction to Rare Event Simulation. In *Rare Event Simulation Using Monte Carlo Methods*, chapter 1, pages 1–13. John Wiley & Sons, Ltd, 2009. ISBN 978-0-470-74540-3. doi: 10.1002/9780470745403.ch1.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Sai Vemprala, Aleksander Madry, and Ashish Kapoor. Unadversarial Examples: Designing Objects for Robust Vision. In *Advances in Neural Information Processing Systems*, volume 34, pages 15270–15284. Curran Associates, Inc., 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

Aman Sinha, Matthew O'Kelly, Russ Tedrake, and John Duchi. Neural bridge sampling for evaluating safety-critical autonomous systems. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, pages 6402–6416, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-71382-954-6.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, January 2023.

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do Differentiable Simulators Give Better Policy Gradients? In *Proceedings of the 39th International Conference on Machine Learning*, pages 20668–20696. PMLR, June 2022.

Haowei Sun, Shuo Feng, Xintao Yan, and Henry X. Liu. Corner Case Generation and Analysis for Safety Assessment of Autonomous Vehicles. *Transportation Research Record*, 2675(11):587–600, November 2021. ISSN 0361-1981. doi: 10.1177/03611981211018697.

Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. AdvSim: Generating Safety-Critical Scenarios for Self-Driving Vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021.

Chejian Xu, Wenhao Ding, Weijie Lyu, Zuxin Liu, Shuai Wang, Yihan He, Hanjiang Hu, Ding Zhao, and Bo Li. SafeBench: A Benchmarking Platform for Safety Evaluation of Autonomous Vehicles. *Advances in Neural Information Processing Systems*, 35:25667–25682, December 2022.

Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z. Morley Mao. On Adversarial Robustness of Trajectory Prediction for Autonomous Vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15159–15168, 2022.

Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided Conditional Diffusion for Controllable Traffic Simulation, October 2022.

Yilun Zhou, Serena Booth, Nadia Figueroa, and Julie Shah. RoCUS: Robot Controller Understanding via Sampling. In *5th Annual Conference on Robot Learning*, November 2021.

SUPPLEMENTARY MATERIAL

SAMPLING USING METROPOLIS-ADJUSTED LANGEVIN ALGORITHM (MALA)

For reference, we include the pseudocode in Algorithm 2 for sampling from an unnormalized likelihood function using the Metropolis-Adjusted Langevin Algorithm (MALA; Ma et al. (2019)). MALA can be interpreted as a special case of Hamiltonian Monte Carlo (HMC; Betancourt (2017)) where the Hamiltonian dynamics are simulated for only a single step, and setting the gradient to zero recovers RMH sampling. Compared with HMC, MALA generates samples that are more highly auto-correlated but requires fewer evaluations of the likelihood and its gradient. In our case, since each step requires a forward and backward pass through the simulator and renderer, we find that this trade-off favors MALA.

In Algorithm 1, we use MALA to sample N i.i.d. samples. We do this by running MALA N times in parallel from different starting conditions; each of the N samples from the previous sampling round is used to initialize a different parallel MALA instance.

Algorithm 2: Metropolis-Adjusted Langevin Algorithm (MALA; Ma et al. (2019))

Input: Number of steps m, stepsize λ, unnormalized likelihood $p(x)$, initial guess x_0

for $i = 1, \ldots, m$ **do**

- $\eta \sim \mathcal{N}(0, 2\lambda I)$; `// Gaussian diffusion`
- $x_i = x_{i-1} + \lambda \nabla \log p(x_{i-1}) + \eta$; `// Add gradient drift`
- $P_{accept} \leftarrow \frac{p(x_i) e^{-||x_{i-1} - x_i - \lambda \nabla \log p(x_i)||^2/(4\lambda)}}{p(x_{i-1}) e^{-||x_i - x_{i-1} - \lambda \nabla \log p(x_{i-1})||^2/(4\lambda)}}$;
- With probability $1 - \min(1, P_{accept})$: $x_i \leftarrow x_{i-1}$; `// Accept/reject`

end

Return: A sample drawn with the given likelihood $x_m \sim p(x)$.

PROOF OF THEOREM 4.1

We will show the proof for sampling from the failure generating process with likelihood given by Eq. (4); the proof for the repair generating process follows similarly. The log-likelihood for the failure generating process is

$$\log p_{\phi,0}(\phi) - \tau[J^* - J(\theta, \phi)]_+ \tag{7}$$

Ma et al. (2019) show that MALA sampling enjoys the convergence guarantees of Theorem 4.1 so long as the target log likelihood is strongly convex outside of a ball of finite radius R (see Theorem 1 in Ma et al. (2019)). Since $\log p_{\phi,0}(\phi)$ is assumed to be strongly m-convex, it is sufficient to show that as $||\phi|| \to \infty$, the strong convexity of the log-prior dominates the non-convexity in $\tau[J^* - J(\theta, \phi)]_+$.

For convenience, denote $f(\phi) = -\tau[J^* - J(\theta, \phi)]_+$ and $g(\phi) = \log p_{\phi,0}(\phi)$. We must first show that $f(\phi) + g(\phi)$ is $(m - L)$-strongly convex, for which it suffices to show that $f(\phi) + g(\phi) - (m - L)/2||\phi||^2$ is convex. Note that

$$f(\phi) + g(\phi) - \frac{m - L}{2}||\phi||^2 = f(\phi) + \frac{L}{2}||\phi||^2 + g(\phi) - \frac{m}{2}||\phi||^2 \tag{8}$$

$g(\phi) - \frac{m}{2}||\phi||^2$ is convex by m-stong convexity of g, so we must show that the remaining term, $f(\phi) + L/2||\phi||^2$, is convex. Note that the Hessian of this term is $\nabla^2 f(\phi) + LI$. Since we have assumed that J is L-Lipschitz smooth (i.e. its gradients are L-Lipschitz continuous), it follows that the magnitudes of the eigenvalues of $\nabla^2 f$ are bounded by L, which is sufficient for $\nabla^2 f(\phi) + LI$ to be positive semi-definite, completing the proof.

PROOF OF THEOREM 4.2

We can treat Alg. 1 as a two-stage Gibbs sampling procedure and apply the Hammersley-Clifford Theorem (Robert and Casella, 2004) to get the joint distribution

$$f_{\theta,\phi}(\theta^*, \phi^*) = p_{\theta,0}(\theta^*)p_{\phi,0}(\phi^*)\frac{e^{-[J^* - J(\theta^*,\phi^*)]_+}}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]} \tag{9}$$

Integrating over ϕ yields the marginal distribution of θ, completing the proof.

$$f_{\theta^*} = \int_\phi f_{\theta,\phi}(\theta^*, \phi)d\phi = \frac{p_{\theta,0}(\theta^*)}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]}\int_\phi p_{\phi,0}(\phi)e^{-[J^* - J(\theta^*,\phi^*)]_+}d\phi \tag{10}$$

$$= p_{\theta,0}(\theta^*)\frac{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{-[J^* - J(\theta^*,\phi)]_+}\right]}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]} \tag{11}$$

$$= p_{\theta,0}(\theta^*)\frac{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{-(J^* - J(\theta^*,\phi))}|J^* - J(\theta^*,\phi) \geq 0\right] + \mathbb{E}_{\phi\sim p_{\phi,0}}\left[1|J^* - J(\theta^*,\phi) < 0\right]}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]} \tag{12}$$

$$= p_{\theta,0}(\theta^*)\frac{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{-(J^* - J(\theta^*,\phi))}|J^* \geq J(\theta^*,\phi)\right] + \mathbb{E}_{\phi\sim p_{\phi,0}}\left[1|J^* < J(\theta^*,\phi)\right]}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]} \tag{13}$$

$$= p_{\theta,0}(\theta^*)\frac{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{-(J^* - J(\theta^*,\phi))}|J^* \geq J(\theta^*,\phi)\right] + \mathbb{P}[J(\theta^*,\phi) > J^*]}{\mathbb{E}_{\phi\sim p_{\phi,0}}\left[e^{J(\theta^*,\phi)-J^*}\right]} \tag{14}$$

$$\tag{15}$$

CODE

We include anonymized code implementing our method, all baselines, and all environment/task combinations in the attached `.zip` archive, which also includes instructions for installing this code and reproducing our experiments. We plan to open-source this code, which also includes our simple differentiable rendering engine, upon publication.

VIDEO

We provide videos of the predicted failures and repaired policies in the attached video file. These videos were generated using the same rendering engine used in our experiments, to provide a sense of the scene geometry and rendering quality used in our experiments. Note that the videos were rendered at a higher resolution than used for our experiments.

ENVIRONMENT DETAILS

GRASPING

Each task in the grasping environment involved a target object (either a box or a mug) on a table along with a distractor object (a cube). A 64x64 depth image was rendered from a fixed camera position and passed to a grasp identification policy, which was structured as an auto-encoder with 3 convolutional layers and 3 transposed convolutional layers (each with kernel size 7, stride 2, and 32 channels, and ReLU activations). The policy was trained to identify grasp affordances on the object in the form of an image of the same size as the output highlightling the "graspable" regions of the object (this is a common strategy in robot manipulation). θ includes all parameters of the autoencoder, and the environmental parameters ϕ included the 2D pose $[x, y, \psi]$ of the target object and the x position of the distractor object, all treated as Gaussian random variables. The affordance autoencoder network was trained using ground-truth affordances from hand-labelling of the target object.

Cost The cost function for all grasping tasks was the mean square error between the predicted and ground-truth grasp affordances.

Hyperparameters We ran experiments for $N = 5$ rounds of $m = 10$ steps with population size $n = 5$. All methods used learning rate $\lambda = 10^{-3}$ for ϕ and 10^{-5} for θ, with tempering parameter $\alpha = 40$.

DRONE

All drone environments included a corridor $8\,\mathrm{m}$ wide and $30\,\mathrm{m}$ long, with the drone starting roughly $10\,\mathrm{m}$ away from the target (placed at the origin and marked by a black square and red "H"). In both the static and dynamic tasks, there are 5 obstacles in the scene, modeled as green cylinders with radius $0.5\,\mathrm{m}$. The drone has Dubins car dynamics restricted to the xy plane, with control actions for velocity and yaw rate. Control actions are predicted by a policy that takes 32x32 RGB and depth images as input, encodes the images using three convolutional layers (kernel size 7, stride 1, 32 channels), and predicts control actions using a fully connected network with 4 hidden layers of 32 units each (all layers used ReLU activation). θ includes all parameters of the policy, and the environmental parameters ϕ include the initial position of the drone (treated as a Gaussian random variable centered $10\,\mathrm{m}$ away from the target) and the initial positions of all obstacles (treated as uniformly distributed throughout the corridor between the starting point and the target). The dynamic case adds initial velocities to ϕ in x and y for each obstacle, sampled from a Gaussian distribution. The drone's initial policy was trained to mimic an oracle with perfect state information, which we implemented as an optimization-based receding-horizon path planner with perfect information about the state and velocity of the drone and all obstacles.

Cost The cost for both drone examples was the (soft) minimum reward over a trajectory plus the distance to the goal at the end of the trajectory:

$$J = -\text{logsumexp}(-r_t) + \frac{1}{2}\sqrt{x_T^2 + y_T^2} \tag{16}$$

$$r_t = -10\sigma(5 \min_i d_i) \tag{17}$$

where the reward at each timestep r_t is based on the minimum distance $\min_i d_i$ to any obstacle in the scene and $sigma$ is the sigmoid function (used as a smooth approximation of the indicator function).

Hyperparameters We ran experiments for $N = 6$ rounds of $m = 10$ steps with $n = 5$ and learning rate $\lambda = 10^{-2}$ for ϕ and $\lambda = 10^{-5}$ for θ, with tempering parameter $\alpha = 40$.

AV

All AV examples use bicycle kinematics for all agents, with state $[x, y, \psi, v]$, including position, heading, and velocity, and control actions $[\delta, a]$ for steering angle and acceleration. The continuous time dynamics

$$\dot{x} = v\cos\psi \tag{18}$$

$$\dot{y} = v\sin\psi \tag{19}$$

$$\dot{\psi} = \frac{v}{l}\tan\delta \tag{20}$$

$$\dot{v} = a \tag{21}$$

were discretized with timestep $0.1\,\mathrm{s}$. The parameter l denotes axle length and was set to $1\,\mathrm{m}$. Control actions were predicted based on 32x32 RGB and depth images using the same structure as the drone policy (3 convolutional layers and 4 fully connected layers, but the convolutional layers had kernel size 6). In both the highway and intersection tasks, θ includes all trainable parameters of the policy network.

HIGHWAY

The highway example included 2 lanes of traffic, with total width $15\,\mathrm{m}$. We placed one non-ego agent in each lane; these agents track a series of waypoints using an LQR controller. The environmental parameters ϕ include all of these waypoints (5 2D waypoints per non-ego agent), which we modeled

as drawn from a Gaussian distribution about a straight-line path. Future work could explore using a generative model as the prior for non-ego driving agents. The task was simulated for 60 timesteps. The driving policy was pre-trained using proximal policy optimization (PPO) with the same reward as used for testing.

Cost The cost was the soft minimum reward observed over the course of the trajectory:

$$J = -\text{logsumexp}(-r_t) \tag{22}$$

$$r_t = -10\sigma(5 \min_i d_i) + 0.1 v_t \tag{23}$$

where the reward at each timestep r_t is based on the minimum distance $\min_i d_i$ to any obstacle in the scene and the forward velocity ; $sigma$ is the sigmoid function (used as a smooth approximation of the indicator function).

Hyperparameters We ran experiments for $N = 10$ rounds of $m = 10$ steps with population size $n = 5$. All methods used learning rate $\lambda = 10^{-2}$ for ϕ and 2×10^{-5} for θ, with tempering parameter $\alpha = 20$.

INTERSECTION

The intersection example included a 4-way intersection, with 2 lanes of traffic in each $15\,\mathrm{m}$ wide road. We placed two non-ego agent moving left to right in the crossing street and one non-ego agent crossing right to left. Similarly to the highway task, these agents were controlled via LQR to track uncertain trajectories centered about straigt lines. The task was simulated for 70 timesteps. The driving policy was trained using behavior cloning with supervision from an oracle with perfect information about the state of the other agents, which we implemented as a receding-horizon optimization-based path planner.

Cost The cost was the soft minimum reward observed over the course of the trajectory plus a term to test whether the ego agent successfully crosses the intersection.

$$J = -\text{logsumexp}(-r_t) + 0.1[x_T - x_{target}]_+ \tag{24}$$

$$r_t = -10\sigma(5 \min_i d_i) \tag{25}$$

where the reward at each timestep r_t is based on the minimum distance $\min_i d_i$ to any obstacle in the scene and $x_{target} = 20$; $sigma$ is the sigmoid function (used as a smooth approximation of the indicator function).

Hyperparameters Prediction and repair experiments were run for $N = 10$ rounds of $m = 10$ steps with $n = 5$ and learning rate $\lambda = 10^{-2}$ for ϕ and 10^{-4} for θ, with tempering parameter $\alpha = 20$.

BASELINES

GRADIENT DESCENT

This baseline is meant to represent the standard approach of adversarial training with counterexamples. We implement this method identically to Algorithm 1 except that 1) we omit tempering, and 2) instead of using MALA to sample new states, we simply take one step of gradient ascent on the log posterior: $x_i = x_{i-1} + \lambda \nabla \log p(x_{i-1})$ (this corresponds to gradient descent on the cost).

LEARNING TO COLLIDE

This method was utilized in Ding et al. (2020) for proposing safety-critical scenarios in autonomous driving setting. The key idea was to consider the adversarial scenario model as an agent and the driving algorithm as an environment with the reward function optimized using policy gradient method REINFORCE. The original work incorporates the optimization of a reward function designed specifically for the autonomous driving scenario which would yield risky scenarios. For the purpose of performance comparison with our method, we have adopted the core idea of using the policy

gradient method for optimizing a standardized optimization algorithm across all different scenarios. Our implementation of REINFORCE uses a Gaussian perturbation with standard deviation 0.05 and a moving average baseline with update rate 0.5.

COMPUTE REQUIREMENTS

Our experiments for failure prediction and repair required running 4 methods (2 of ours and 2 baselines) on 6 tasks with 4 random seeds each: a total of 96 experiments individual runs. Each of these experiments can be run in under 30 minutes on an NVIDIA GeForce RTX 2080 Ti, using approximately 1-2 GB of GPU memory. When running our experiments, we primarily used an AWS instance with 4 GPUs, each able to fit at least 4 simultaneous runs in memory at once. We estimate that producing the results in our paper required approximately 4-5 hours, with a roughly equal amount of additional time used for parameter tuning, with a cost of around 100 USD in GPU-equipped AWS EC2 instances (not counting any compute used for preliminary experiments during development and debugging).

LICENSE ACKNOWLEDGMENT

Figures in this paper were created using assets designed by Macrovector/Freepix.