Neural Bridge Sampling for Evaluating Safety-Critical Autonomous Systems

Aman Sinha[*]
Stanford University
amans@stanford.edu

Matthew O'Kelly[*]
University of Pennsylvania
mokelly@seas.upenn.edu

Russ Tedrake
Massachusetts Institute of Technology
russt@mit.edu

John Duchi
Stanford University
jduchi@stanford.edu

Abstract

Learning-based methodologies increasingly find applications in safety-critical domains like autonomous driving and medical robotics. Due to the rare nature of dangerous events, real-world testing is prohibitively expensive and unscalable. In this work, we employ a probabilistic approach to safety evaluation in simulation, where we are concerned with computing the probability of dangerous events. We develop a novel rare-event simulation method that combines exploration, exploitation, and optimization techniques to find failure modes and estimate their rate of occurrence. We provide rigorous guarantees for the performance of our method in terms of both statistical and computational efficiency. Finally, we demonstrate the efficacy of our approach on a variety of scenarios, illustrating its usefulness as a tool for rapid sensitivity analysis and model comparison that are essential to developing and testing safety-critical autonomous systems.

1 Introduction

Data-driven and learning-based approaches have the potential to enable robots and autonomous systems that intelligently interact with unstructured environments. Unfortunately, evaluating the performance of the closed-loop system is challenging, limiting the success of such methods in safety-critical settings. Even if we produce a deep reinforcement learning agent better than a human at driving, flying a plane, or performing surgery, we have no tractable way to certify the system's quality. Thus, currently deployed safety-critical autonomous systems are limited to structured environments that allow mechanisms such as PID control, simple verifiable protocols, or convex optimization to enable guarantees for properties like stability, consensus, or recursive feasibility (see *e.g.* [33, 69, 14]). The stylized settings of these problems and the limited expressivity of guaranteeable properties are barriers to solving unstructured, real-world tasks such as autonomous navigation, locomotion, and manipulation.

The goal of this paper is to *efficiently* evaluate complex systems that lack safety guarantees and/or operate in unstructured environments. We assume access to a simulator to test the system's performance. Given a distribution $X \sim P_0$ of simulation parameters that describe typical environments for the system under test, our governing problem is to estimate the probability of an adverse event

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma). \tag{1}$$

The parameter γ is a threshold defining an adverse event, and $f : \mathcal{X} \to \mathbb{R}$ measures the safety of a realization x of the agent and environment (higher values are safer). In this work, we assume P_0 is

[*] Equal contribution

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

known; the system-identification and generative-modeling literatures (*e.g.* [6, 82]) provide several approaches to learn or specify P_0. A major challenge for solving problem (1) is that the better an agent is at performing a task (*i.e.* the smaller p_γ is), the harder it is to confidently estimate p_γ—one rarely observes events with $f(x) \leq \gamma$. For example, when P_0 is light-tailed, the sample complexity of estimating p_γ using naive Monte Carlo samples grows exponentially [19].

Problem (1) is often solved in practice by naive Monte Carlo estimation methods, the simplest of which *explore* the search space via random samples from P_0. These methods are unbiased and easy to parallelize, but they exhibit poor sample complexity. Naive Monte Carlo can be improved by adding an adaptive component *exploiting* the most informative portions of random samples drawn from a sequence of approximating distributions $P_0, P_1, \ldots, P_K$. However, standard adaptive Monte Carlo methods (*e.g.* [20]), though they may use first-order information on the distributions P_k themselves, fail to use first-order information about f to improve sampling; we explicitly leverage this to accelerate convergence of the estimate through *optimization*.

Naive applications of first-order optimization methods in the estimation problem (1)—for example biasing a sample in the direction $-\nabla f(x)$ to decrease $f(x)$—also require second-order information to correct for the distortion of measure that such transformations induce. Consider the change of variables formula for distributions $\rho(y) = \rho(g^{-1}(y)) \cdot |\det J_{g^{-1}}(y)|$ where $y = g(x)$. When $g(x)$ is a function of the gradient $\nabla f(x)$, the volume distortion $|\det J_{g^{-1}}(y)|$ is a function of the Hessian $\nabla^2 f(x)$. Hessian computation, if even defined, is unacceptably expensive for high-dimensional spaces $\mathcal{X}$ and/or simulations that involve the time-evolution of a dynamical system; our approach avoids any Hessian computation. In contrast, gradients $\nabla f(x)$ can be efficiently computed for many closed-loop systems [1, 80, 107, 59] or through the use of surrogate methods [105, 28, 36, 8].

To that end, we propose *neural bridge sampling*, a technique that combines *exploration, exploitation*, and *optimization* to efficiently solve the estimation problem (1). Specifically, we consider a novel Markov-chain Monte Carlo (MCMC) scheme that moves along an adaptive ladder of intermediate distributions P_k (with corresponding unnormalized densities $\rho_k(x)$ and normalizing constants $Z_k := \int_{\mathcal{X}} \rho_k(x)dx$). This MCMC scheme iteratively transforms the base distribution P_0 to the distribution of interest $P_0 I\{f(x) \leq \gamma\}$. Neural bridge sampling adaptively balances exploration in the search space (via $\nabla \log \rho_0$) against optimization (via ∇f), while avoiding Hessian computations. Our final estimate $\hat{p}_\gamma$ is a function of the ratios Z_k/Z_{k-1} of the intermediate distributions P_k, the so-called "bridges" [10, 66]. We accurately estimate these ratios by warping the space between the distributions P_k using neural density estimation.

Contributions and outline Section 2 presents our method, while Section 3 provides guarantees for its statistical performance and overall efficiency. A major focus of this work is empirical, and accordingly, Section 4 empirically demonstrates the superiority of neural bridge sampling over competing techniques in a variety of applications: (i) we evaluate the sensitivity of a formally-verified system to domain shift, (ii) we consider design optimization for high-precision rockets, and (iii) we perform model comparisons for two learning-based approaches to autonomous navigation.

1.1 Related Work

Safety evaluation Several communities [27] have attempted to evaluate the closed-loop performance of cyber-physical, robotic, and embodied agents both with and without learning-based components. Existing solutions are predicated on the definition of the evaluation problem: verification, falsification, or estimation. In this paper we consider a method that utilizes interactions with a gradient oracle in order to solve the estimation problem (1). In contrast to our approach, the verification community has developed tools (*e.g.* [56, 24, 4]) to investigate whether any adverse or unsafe executions of the system exist. Such methods can certify that failures are impossible, but they require that the model is written in a formal language (a barrier for realistic systems), and they require whitebox access to this formal model. Falsification approaches (*e.g.* [40, 31, 5, 108, 34, 83]) attempt to find *any* failure cases for the system (but not the overall probability of failure). Similar to our approach, some falsification approaches (*e.g.* [1, 107]) utilize gradient information, but their goal is to simply minimize $f(x)$ rather than solve problem (1). Adversarial machine learning is closely related to falsification; the key difference is the domain over which the search for falsifying evidence is conducted. Adversarial examples (*e.g.* [61, 53, 95, 99]) are typically restricted to a p-norm ball around a point from a dataset, whereas falsification considers all possible in-distribution examples.

Both verification and falsification methods provide less information about the system under test than estimation-based methods: they return only whether or not the system satisfies a specification. When the system operates in an unstructured environment (*e.g.* driving in an urban setting), the mere existence of failures is trivial to demonstrate [93]. Several authors (*e.g.* [76, 104]) have proposed that it is more important in such settings to understand the overall frequency of failures as well as the relative likelihoods of different failure modes, motivating our approach.

Sampling techniques and density estimation When sampling rare events and estimating their probability, there are two main branches of related work: parametric adaptive importance sampling (AIS) [63, 75] and nonparametric sequential Monte Carlo (SMC) techniques [32, 30]. Both of these literatures are advanced forms of variance reduction techniques, and they are complementary to standard methods such as control variates [91, 46]. Parametric AIS techniques, such as the cross-entropy method [90], postulate a family of distributions for the optimal importance-sampling distribution. They iteratively perform heuristic optimization procedures to update the sampling distribution. SMC techniques perform sampling from a sequence of probability distributions defined nonparametrically by the samples themselves. The SMC formalism encompasses particle filters, birth-death processes, and smoothing filters [29]. Our technique blends aspects of both of these communities: we include parametric warping distributions in the form of normalizing flows [82] within the SMC setting.

Our method employs bridge sampling [10, 66], which is closely related to other SMC techniques such as umbrella sampling [23], multilevel splitting [16, 20], and path sampling [41]. The operational difference between these methods is in the form of the intermediate distribution used to calculate the ratio of normalizing constants. Namely, the optimal umbrella sampling distribution is more brittle than that of bridge sampling [23]. Multilevel splitting employs hard barriers through indicator functions, whereas our approach relaxes these hard barriers with smoother exponential barriers. Path sampling generalizes bridge sampling by taking discrete bridges to a continuous limit; this approach is difficult to implement in an adaptive fashion.

The accuracy of bridge sampling depends on the overlap between intermediate distributions P_k. Simply increasing the number of intermediate distributions is inefficient, because it requires running more simulations. Instead, we employ a technique known as *warping*, where we map intermediate distributions to a common reference distribution [102, 65]. Specifically, we use normalizing flows [86, 54, 81, 82], which efficiently transform arbitrary distributions to standard Gaussians through a series of deterministic, invertible functions. Normalizing flows are typically used for probabilistic modeling, variational inference, and representation learning. Recently, Hoffman et al. [47] explored the benefits of using normalizing flows for reparametrizing distributions within MCMC; our warping technique encompasses this benefit and extends it to the SMC setting.

Beyond simulation This paper assumes that the generative model P_0 of the operating domain is given, so all failures are in the modeled domain by definition. When deploying systems in the real world, anomaly detection [22] can discover distribution shifts and is complementary to our approach (see *e.g.* [26, 68]). Alternatively, the problem of distribution shift can be addressed offline via distributional robustness [39, 70, 84], where we analyze the worst-case probability of failure under an uncertainty set composed of perturbations to P_0.

2 Proposed approach

As we note in Section 1, naive Monte Carlo measures probabilities of rare events inefficiently. Instead, we consider a sequential Monte Carlo (SMC) approach: we decompose the rare-event probability p_γ into a chain of intermediate quantities, each of which is tractable to compute with standard Monte Carlo methods. Specifically, consider K distributions P_k with corresponding (unnormalized) probability densities ρ_k and normalizing constants $Z_k := \int_{\mathcal{X}} \rho_k(x)dx$. Let ρ_0 correspond to the density for P_0 and $\rho_\infty(x) := \rho_0(x)I\{f(x) \leq \gamma\}$ be the (unnormalized) conditional density for the region of interest. Then, we consider the following decomposition:

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) = \mathbb{E}_{P_K}\left[\frac{Z_K}{Z_0}\frac{\rho_\infty(X)}{\rho_K(X)}\right], \qquad \frac{Z_K}{Z_0} = \prod_{k=1}^{K} \frac{Z_k}{Z_{k-1}}. \tag{2}$$

Algorithm 1 Neural bridge sampling

```
Input: $N$ samples $x_i^0 \overset{\text{i.i.d.}}{\sim} P_0$, MCMC steps $T$, step size $\alpha \in (0,1)$, stop condition $s \in (0,1)$
Initialize $k \leftarrow 0$, $\beta_0 \leftarrow 0$, $\log(\hat{p}_\gamma) \leftarrow 0$
while $\frac{1}{N}\sum_i I\{f(x_i^k) \le \gamma\} < s$ do
    $\beta_{k+1} \leftarrow$ solve problem (8)
    for $i = 1$ to $N$, in parallel
        $x_i^{k+1} \overset{\text{i.i.d.}}{\sim} \text{Mult}(\{\rho_{k+1}(x_i^k)/\rho_k(x_i^k)\})$ // multinomial resampling
    for $t = 1$ to $T$
        for $i = 1$ to $N$, in parallel
            $x_i^{k+1} \leftarrow \text{WarpedHMC}(x_i^k, \theta_k)$ // Appendix A
    $\theta_{k+1} \leftarrow$ argmin problem (6) // train normalizing flow on $\{x_i^{k+1}\}$ via SGD
    $\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(Z_{k+1}/Z_k)$ // warped bridge estimate (5)
    $k \leftarrow k + 1$
$\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(\frac{1}{N}\sum_i I\{f(x_i^k) \le \gamma\})$
```

Although we are free to choose the intermediate distributions arbitrarily, we will show below that our estimate for each ratio Z_k/Z_{k-1} and thus p_γ is accurate insofar as the distributions sufficiently overlap (a concept we make rigorous in Section 3). Thus, the intermediate distributions act as bridges that iteratively steer samples from P_0 towards P_K. One special case is the multilevel splitting approach [50, 16, 104, 74], where $\rho_k(x) := \rho_0(x) I\{f(x) \le L_k\}$ for levels $\infty =: L_0 > L_1 \ldots > L_K := \gamma$. In this paper, we introduce an exponential tilting barrier [94]

$$\rho_k(x) := \rho_0(x) \exp\left(\beta_k \left[\gamma - f(x)\right]_-\right), \tag{3}$$

which allows us to take advantage of gradients $\nabla f(x)$. Here we use the "negative ReLU" function defined as $[x]_- := -[-x]_+ = x I\{x < 0\}$, and we assume that the measure of non-differentiable points, *e.g.* where $\nabla f(x)$ does not exist or $f(x) = \gamma$, is zero (see Appendix A for a detailed discussion of this assumption). We set $\beta_0 := 0$ and adaptively choose $\beta_k > \beta_{k-1}$. The parameter β_k tilts the distribution towards the distribution of interest: $\rho_k \to \rho_\infty$ as $\beta_k \to \infty$. In what follows, we describe an MCMC method that combines exploration, exploitation, and optimization to draw samples $X_i^k \sim P_k$. We then show how to compute the ratios Z_k/Z_{k-1} given samples from both P_{k-1} and P_k. Finally, we describe an adaptive way to choose the intermediate distributions P_k. Algorithm 1 summarizes the overall approach.

MCMC with an exponential barrier Gradient-based MCMC techniques such as the Metropolis-adjusted Langevin algorithm (MALA) [89, 88] or Hamiltonian Monte Carlo (HMC) [35, 73] use gradients $\nabla \log \rho_0(x)$ to efficiently explore the space $\mathcal{X}$ and avoid inefficient random-walk behavior [37, 25]. Classical mechanics inspires the HMC approach: HMC introduces an auxiliary random momentum variable $v \in \mathcal{V}$ and generates proposals by performing Hamiltonian dynamics in the augmented state-space $\mathcal{X} \times \mathcal{V}$. These dynamics conserve volume in the augmented state-space, even when performed with discrete time steps [58].

By including the barrier $\exp\left(\beta_k \left[\gamma - f(x)\right]_-\right)$, we combine exploration with optimization; the magnitude of β_k in the barrier modulates the importance of ∇f (optimization) over $\nabla \log \rho_0$ (exploration), two elements of the HMC proposal (see Appendix A for details). We discuss the adaptive choice for β_k below. Most importantly, we avoid any need for Hessian computation because the dynamics conserve volume. As Algorithm 1 shows, we perform MCMC as follows: given N samples $x_i^{k-1} \sim P_{k-1}$ and a threshold β_k, we first resample using their importance weights (exploiting the performance of samples that have lower function value than others) and then perform T HMC steps. In this paper, we implement split HMC [92] which is convenient for dealing with the decomposition of $\log \rho_k(x)$ into $\log \rho_0(x) + \beta_k[\gamma - f(x)]_-$ (see Appendix A for details).

Estimating Z_k/Z_{k-1} via bridge sampling Bridge sampling [10, 66] allows estimating the ratio of normalizing constants of two distributions by rewriting

$$E_k := \frac{Z_k}{Z_{k-1}} = \frac{Z_k^B/Z_{k-1}}{Z_k^B/Z_k} = \frac{\mathbb{E}_{P_{k-1}}[\rho_k^B(X)/\rho_{k-1}(X)]}{\mathbb{E}_{P_k}[\rho_k^B(X)/\rho_k(X)]}, \qquad \widehat{E}_k = \frac{\sum_{i=1}^N \rho_k^B(x_i^{k-1})/\rho_{k-1}(x_i^{k-1})}{\sum_{i=1}^N \rho_k^B(x_i^k)/\rho_k(x_i^k)}, \tag{4}$$

where ρ_k^B is the density for a bridge distribution between P_{k-1} and P_k, and Z_k^B is its associated normalizing constant. We employ the geometric bridge $\rho_k^B(x) := \sqrt{\rho_{k-1}(x)\rho_k(x)}$. In addition to being simple to compute, bridge sampling with a geometric bridge enjoys the asymptotic performance guarantee that the relative mean-square error scales inversely with the Bhattacharyya coefficient, $G(P_{k-1}, P_k) = \int_{\mathcal{X}} \sqrt{\frac{\rho_{k-1}(x)}{Z_{k-1}} \frac{\rho_k(x)}{Z_k}} dx \in [0, 1]$ (see Appendix B for a proof). This value is closely related to the Hellinger distance, $H(P_{k-1}, P_k) = \sqrt{2 - 2G(P_{k-1}, P_k)}$. In Section 3, we analyze the ramifications of this fact on the overall convergence of our method.

Neural warping Both HMC and bridge sampling benefit from warping samples x_i into a different space. As Betancourt [11] notes, HMC mixes poorly in spaces with ill-conditioned geometries. Girolami and Calderhead [42] and Hoffman et al. [47] explore techniques to improve mixing efficiency by minimizing shear in the corresponding Hamiltonian dynamics. One way to do so is to transform to a space that resembles a standard isotropic Gaussian [62].

Conveniently, transforming P_k to a common distribution (*e.g.* a standard Gaussian) also benefits the bridge-sampling estimator (4). As noted above, the error of the bridge estimator grows with the Hellinger distance between the distributions $H(P_{k-1}, P_k)$. However, normalizing constants Z_k are invariant to (invertible) transformations. Thus, transformations that warp the space between distributions reduce the error of the bridge-sampling estimator (4). Concretely, we consider invertible transformations W_k such that $y_i^k = W_k(x_i^k)$. For clarity of notation, we write probability densities over the space $\mathcal{Y}$ as ϕ, the corresponding distributions for Y^k as Q_k, and the inverse transformations $W_k^{-1}(y)$ as $V_k(y)$. Then we can write the bridge-sampling estimate (4) in terms of the transformed variables y. The numerator and denominator are as follows:

$$\mathbb{E}_{Q_{k-1}}\left[\frac{\phi_k^B(Y)}{\phi_{k-1}(Y)}\right] = \mathbb{E}_{Q_{k-1}}\left[\sqrt{\frac{\phi_k(Y)}{\phi_{k-1}(Y)}}\right] = \mathbb{E}_{Q_{k-1}}\left[\sqrt{\frac{\rho_k(V_k(Y))|\det J_{V_k}(Y)|}{\rho_{k-1}(V_{k-1}(Y))|\det J_{V_{k-1}}(Y)|}}\right], \tag{5a}$$

$$\mathbb{E}_{Q_k}\left[\frac{\phi_k^B(Y)}{\phi_k(Y)}\right] = \mathbb{E}_{Q_k}\left[\sqrt{\frac{\phi_{k-1}(Y)}{\phi_k(Y)}}\right] = \mathbb{E}_{Q_k}\left[\sqrt{\frac{\rho_{k-1}(V_{k-1}(Y))|\det J_{V_{k-1}}(Y)|}{\rho_k(V_k(Y))|\det J_{V_k}(Y)|}}\right]. \tag{5b}$$

By transforming all P_k into Q_k to resemble standard Gaussians, we reduce the Hellinger distance $H(Q_{k-1}, Q_k) \leq H(P_{k-1}, P_k)$. Note that the volume distortions in the expression (5) are functions of the transformation V_k, so they do not require computation of the Hessian $\nabla^2 f$. However, computing $\rho_k(V_k(y))$ requires evaluations of f (*e.g.* calls of the simulator). We consider the cost-benefit analysis of warping in Section 3.

Classical warping techniques include simple mean shifts or affine scaling [102, 65]. Similar to Hoffman et al. [47], we consider normalizing flows, a much more expressive class of transformations that have efficient Jacobian computations [82]. Specifically, given samples x_i^k, we train masked autoregressive flows (MAFs) [81] to minimize the empirical KL divergence between the transformed samples y_i^k and a standard Gaussian $D_{\mathrm{KL}}(Q_k \| \mathcal{N}(0, I))$. Parametrizing W_k by θ_k, this minimization problem is equivalent to:

$$\text{minimize}_\theta \sum_{i=1}^N - \log \left|\det J_{W_k}\left(x_i^k; \theta\right)\right| + \frac{1}{2}\left\|W_k\left(x_i^k; \theta\right)\right\|_2^2. \tag{6}$$

The KL divergence is an upper bound to the Hellinger distance; we found minimizing the former to be more stable than minimizing the latter. Furthermore, to improve training efficiency, we exploit the iterated nature of the problem and warm-start the weights θ_k with the trained values θ_{k-1} when solving problem (6) via stochastic gradient descent (SGD). As a side benefit, the trained flows can be repurposed as importance-samplers for the ladder of distributions from nominal behavior to failure.

Adaptive intermediate distributions Because we assume no prior knowledge of the system under test, we exploit previous progress to choose the intermediate β_k online; this is a key difference to our approach compared to other forms of sequential Monte Carlo (*e.g.* [71, 72]) which require a predetermined schedule for β_k. We define the quantities

$$a_k := \textstyle\sum_i^N I\{f(x_i^k) \leq \gamma\}/N, \quad b_k(\beta) := \textstyle\sum_{i=1}^N \exp\left((\beta - \beta_k)[\gamma - f(x_i^k)]_-\right)/N. \tag{7}$$

The first is the fraction of samples that have achieved the threshold. The second is an importance-sampling estimate of E_{k+1} given samples $x_i^k \sim P_k$, written as a function of β. For fixed fractions $\alpha, s \in (0, 1)$ with $\alpha < s$, β_{k+1} solves the following optimization problem:

$$\text{maximize } \beta \text{ s.t. } \{b_k(\beta) \geq \alpha, \;\; a_k/b_k(\beta) \leq s\}. \tag{8}$$

Since $b_k(\beta)$ is monotonically decreasing and $b_k(\beta) \geq a_k$, this problem can be solved efficiently via binary search. The constant α tunes how quickly we enter the tails of P_0 (smaller α means fewer iterations), whereas s is a stop condition for the last iteration. Choosing β_{k+1} via (8) yields a crude estimate for the ratio Z_{k+1}/Z_k as α (or a_{K-1}/s for the last iteration). The bridge-sampling estimate $\widehat{E}_{k+1}$ corrects this crude estimate once we have samples from the next distribution P_{k+1}.

3 Performance analysis

We can write the empirical estimator of the function (2) as

$$\hat{p}_\gamma = \prod_{k=1}^{K} \widehat{E}_k \frac{1}{N} \sum_{i=1}^{N} \frac{\rho_\infty(x_i^K)}{\rho_K(x_i^K)}, \tag{9}$$

where $\widehat{E}_k$ is given by the expression (4) without warping, or similarly, as a Monte Carlo estimate of the expression (5) with warping. We provide guarantees for both the time complexity of running Algorithm 1 (*i.e.* the iterations K) as well as the overall mean-square error of $\hat{p}_\gamma$. For simplicity, we provide results for the asymptotic (large N) and well-mixed MCMC (large T) limits. Assuming these conditions, we have the following:

Proposition 1. *Let $K_0 := \lfloor \log(p_\gamma)/\log(\alpha) \rfloor$. Then, for large N and T, $s \geq 1/3$, and $p_\gamma < s$, the total number of iterations in Algorithm 1 approaches $K \overset{\text{a.s.}}{\to} K_0 + I\{p_\gamma/\alpha^{K_0} < s\}$. Furthermore, for the non-warped estimator, the asymptotic relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ is*

$$\frac{2}{N}\sum_{k=1}^{K}\left(\frac{1}{G(P_{k-1},P_k)^2} - 1\right) - \frac{2}{N}\sum_{k=1}^{K-1}\left(\frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})} - 1\right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right). \tag{10}$$

In particular, if the inverse Bhattacharyya coefficients are bounded such that $\frac{1}{G(P_{k-1},P_k)^2} \leq D$ (with $D \geq 1$), then the asymptotic relative mean-square error satisfies $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2] \leq 2KD/N$. For the warped estimator, replace $G(P_i, P_j)$ with $G(Q_i, Q_j)$ in the expression (10).

See Appendix B for the proof. We provide some remarks about the above result. Intuitively, the first term in the bound (10) accounts for the variance of $\widehat{E}_k$. The denominator of $\widehat{E}_{k-1}$ and numerator of $\widehat{E}_k$ both depend on x_i^k; the second sum in (10) accounts for the covariance between those terms. Furthermore, the quantities in the bound (10) are all empirically estimable, so we can compute the mean-square error from a single pass of Algorithm 1. In particular,

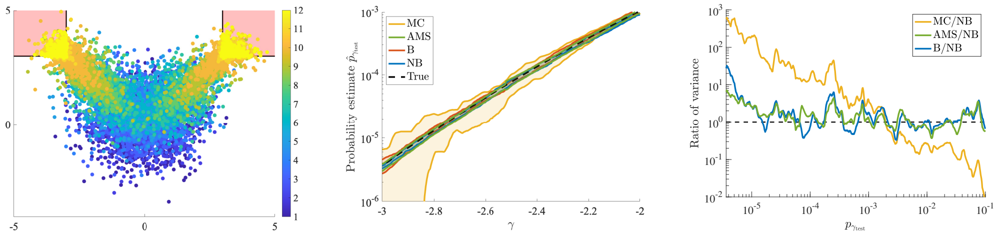
$$G(P_{k-1},P_k)^2 = \frac{Z_k^B}{Z_{k-1}}\frac{Z_k^B}{Z_k}, \qquad \frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})} = \frac{Z_k^C}{Z_k}\frac{Z_k}{Z_k^B}\frac{Z_k}{Z_{k+1}^B}, \tag{11}$$

where $Z_k^C/Z_k = \mathbb{E}_{P_k}\left[\rho_k^B(X)\rho_{k+1}^B(X)/\rho_k(X)^2\right]$. The last term in the bound (10) is the relative variance of the final Monte Carlo estimate $\sum_i I\{f(x_i^K) \leq \gamma\}/N$.

Overall efficiency The statistical efficiency outlined in Proposition 1 is pointless if it is accompanied by an overwhelming computational cost. We take the atomic unit of computation to be a query of the simulator, which returns both evaluations of $f(x)$ and $\nabla f(x)$; we assume other computations to be negligible compared to simulation. As such, the cost of Algorithm 1 is $N(1+KT)$ evaluations of the simulator without warping and $N(1+KT)+2KN$ with warping. Thus, the relative burden of warping is minimal, because training the normalizing flows to minimize $D_{\mathrm{KL}}(Q_k\|\mathcal{N}(0,I))$ requires no extra simulations. In contrast, directly minimizing $D_{\mathrm{KL}}(Q_{k-1}\|Q_k)$ would require extra simulations at each training step to evaluate $\rho_k(V_k(y))$.

Our method can exploit two further sources of efficiency. First, we can employ surrogate models for gradient computation and/or function evaluation during the T MCMC steps. For example, using a surrogate model for a fraction $d \leq 1 - 1/T$ of the MCMC iterations reduces the factor T to $T_s := (1-d)T$ in the overall cost. Surrogate models have an added benefit of making our approach amenable for simulators that do not provide gradients. The second source of efficiency is parallel computation. Given C processors, the factor N in the cost drops to $N_c := \lceil N/C \rceil$.

The overall efficiency of the estimator (9)—relative error multiplied by cost [44]—depends on p_γ as $\log(p_\gamma)^2$. In contrast, the standard Monte Carlo estimator has cost N to produce an estimate with



(a) Samples colored by iteration

(b) $\hat{p}_{\gamma_{\text{test}}}$ vs. γ_{test}

(c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

Figure 1. Experiments on a synthetic problem. 10 trials are used to calculate the 99% confidence intervals in (b) and variance ratios in (c). All adaptive methods perform similarly in this well-conditioned search space except at very small γ, where NB performs the best.

relative error $\frac{1-p_\gamma}{p_\gamma N}$. Thus, the relative efficiency gain for our estimator (9) over naive Monte Carlo is $O(1/(p_\gamma \log(p_\gamma)^2))$: the efficiency gains over naive Monte Carlo increase as p_γ decreases.

4 Experiments

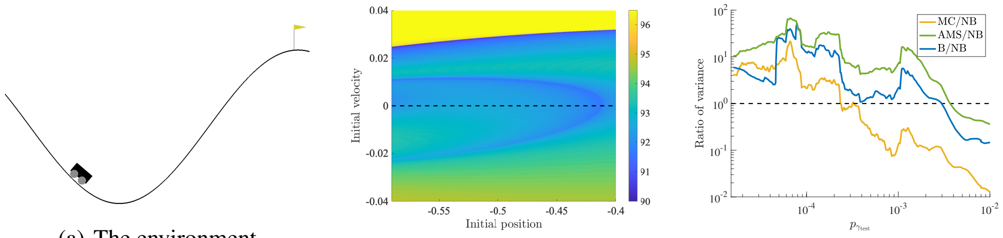
We evaluate our approach in a variety of scenarios, showcasing its use in efficiently evaluating the safety of autonomous systems. We begin with a synthetic problem to illustrate the methodology concretely as well as highlight the pitfalls of using gradients naively. Then, we evaluate a formally-verified neural network controller [48] on the OpenAI Gym continuous MountainCar environment [67, 17] under a domain perturbation. Finally, we consider two examples of using neural bridge sampling as a tool for engineering design in high-dimensional settings: (a) comparing thruster sizes to safely land a rocket [13] in the presence of wind, and (b) comparing two algorithms on the OpenAI Gym CarRacing environment (which requires a surrogate model for gradients) [55].

We compare our method with naive Monte Carlo (MC) and perform ablation studies for the effects of neural warping (denoted as NB with warping and B without). We also provide comparisons with adaptive multilevel splitting (AMS) [16, 104, 74]. All methods are given the same computational budget as measured by evaluations of the simulator. This varies from 50,000-100,000 queries to run Algorithm 1 as determined by p_γ (see Appendix C for details of each experiment's hyperparameters). However, despite running Algorithm 1 with a given γ, we evaluate estimates $\hat{p}_{\gamma_{\text{test}}}$ for all $\gamma_{\text{test}} \geq \gamma$. Larger γ_{test} require fewer queries to evaluate $\hat{p}_{\gamma_{\text{test}}}$ (as Algorithm 1 terminates early). Thus, we adjust the number of MC queries accordingly for each γ_{test}. Independently, we calculate the ground-truth values $p_{\gamma_{\text{test}}}$ for the non-synthetic problems using a fixed, very large number of MC queries.

Synthetic problem We consider the two-dimensional function $f(x) = -\min(|x_{[1]}|, x_{[2]})$, where $x_{[i]}$ is the i^{th} dimension of $x \in \mathbb{R}^2$. We let $\gamma = -3$ and $P_0 = \mathcal{N}(0, I)$ (for which $p_\gamma = 3.6 \cdot 10^{-6}$). Note that $\nabla^2 f(x) = 0$ almost everywhere, yet $\nabla f(x)$ has negative divergence in the neighborhoods of $x_{[2]} = |x_{[1]}|$. Indeed, gradient descent collapses $x_i \sim P_0$ to the lines $x_{[2]} = |x_{[1]}|$, and the ill-defined nature of the Hessian makes it unsuitable to track volume distortions. Thus, simple gradient-based transformations used to find adversarial examples (*e.g.* minimize $f(x)$) should not be used for estimation in the presence of non-smooth functions, unless volume distortions can be quantified.

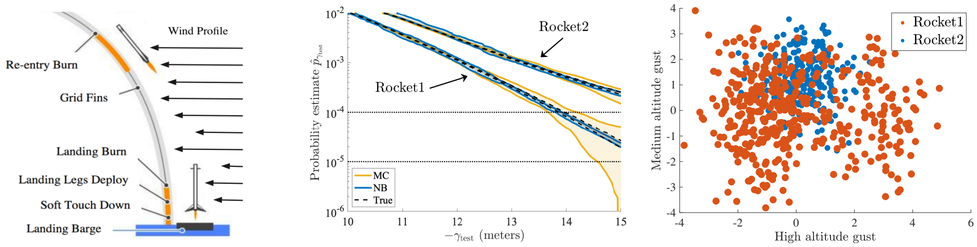
Figure 1(a) shows the region of interest in pink and illustrates the gradual warping of ρ_0 towards ρ_∞ over iterations of Algorithm 1. Figures 1(b) and 1(c) indicate that all adaptive methods outperform MC for $p_{\gamma_{\text{test}}} < 10^{-3}$. For larger $p_{\gamma_{\text{test}}}$, the overhead of the adaptive methods renders MC more efficient (Figure 1(c)). The linear trend of the yellow MC/NB line in Figure 1(c) aligns with the theoretical efficiency gain discussed in Section 3. Finally, due to the simplicity of the search space and the landscape of $f(x)$, the benefits of gradients and warping are not drastic. Specifically, as shown in Figure 1(c), all adaptive methods have similar confidence in their estimates except at very small $p_{\gamma_{\text{test}}} < 10^{-5}$, where NB outperforms AMS and B. The next example showcases the benefits of gradients as well as neural warping in a more complicated search space.

Sensitivity of a formally-verified controller under domain perturbation We consider a minimal reinforcement learning task, the MountainCar problem [67] (Figure 2(a)). Ivanov et al. [48] created a formally-verified neural network controller to achieve reward > 90 over all initial positions



(a) The environment (b) Contours of $f(x)$ (c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

Figure 2. Experiments on the MountainCar environment. The dashed horizontal line in (b) is the line along which the controller is formally verified. 10 trials are used for the variance ratios in (c). The irregular geometry degrades performance of AMS and B, but B benefits slightly from gradients over AMS. NB uses gradients and neural warping to outperform all other techniques.



(a) Rocket landing (b) Failure rates (c) Failure modes

Figure 3. Rocket design experiments. NB's high-confidence estimates enable quick design iterations to either increase the landing pad radius or consider a third rocket that fails with probability $< 10^{-5}$. Low-dimensional visualization shows that Rocket2's failure types are more concentrated than those of Rocket1, even though Rocket2 has a higher overall probability of failure.

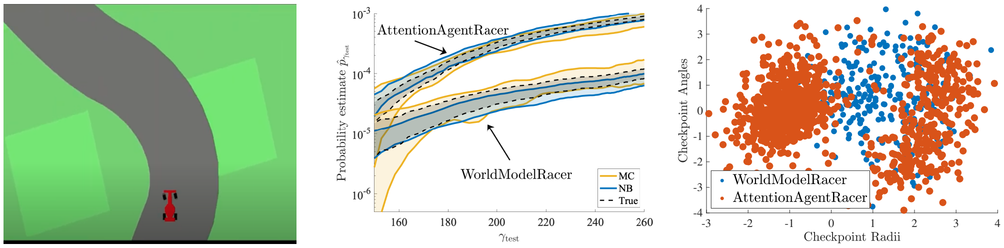
$\in [-0.59, -0.4]$ and 0 initial velocity (see Appendix C). The guarantees of formal verification hold only with respect to the specified domain; even small domain perturbations can affect system performance [49]. We illustrate this sensitivity by adding a small perturbation to the initial velocity $\sim \mathcal{N}(0, 10^{-4})$ and seek $p_\gamma := \mathbb{P}_0(\text{reward} \le 90)$ for $P_0 = \text{Unif}(-0.59, -0.4) \times \mathcal{N}(0, 10^{-4})$. We measure the ground-truth failure rate as $p_\gamma = 1.6 \cdot 10^{-5}$ using 50 million naive Monte Carlo samples.

Figure 2(b) shows contours of $f(x)$. Notably, the failure region (dark blue) is an extremely irregular geometry with pathological curvature, which renders MCMC difficult for AMS and B [11]. Quantitatively, poor mixing adversely affects the performance of AMS and B, and they perform even worse than MC (Figure 2(c)). Whereas gradients help B slightly over AMS, gradients and neural warping together help NB outperform all other methods. We next move to higher-dimensional systems.

Rocket design We now consider the problem of autonomous, high-precision vertical landing of an orbital-class rocket (Figure 3(a)), a technology first demonstrated by SpaceX in 2015. Rigorous system-evaluation techniques such as our risk-based framework are powerful tools for quickly exploring design tradeoffs. In this experiment, the amount of thrust which the rocket is capable of deploying to land safely must be balanced against the payload it is able to carry to space; stronger thrust increases safety but decreases payloads. We consider two rocket designs and we evaluate their respective probabilities of failure (not landing safely on the landing pad) for landing pad sizes up to 15 meters in radius. That is, $-f(x)$ is the distance from the landing pad's center at touchdown and $\gamma = -15$. We evaluate whether the rockets perform better than a threshold failure rate of 10^{-5}.

We let P_0 be the 100-dimensional search space parametrizing the sequence of wind-gusts during the rocket's flight. Appendix C contains details for this parametrization and the closed-loop simulation of the rocket's control law (based on industry-standard approaches [13, 87]). Figure 3(b) shows the estimated performance of the two rockets. We show only MC and NB for clarity; comparisons with other methods are in Table 1 (with ground-truth values calculated using 50 million naive Monte Carlo simulations). Whereas both NB and MC confidently estimate Rocket2's failure rate as higher than 10^{-4}, only NB confidently estimates Rocket1's failure rate as higher than 10^{-5}, letting engineers quickly judge whether to increase the size of the landing pad or build a better rocket.

We can also distinguish between the modes of failure for the rockets. Namely, Figure 3(c) shows a PCA projection of failures (with $\gamma_{\text{test}} = -15$) onto 2 dimensions. Analysis of the PCA modes



(a) The environment (b) Failure rates (c) Failure modes

Figure 4. CarRacing experiments. MC cannot distinguish between the policies below $\gamma_{\text{test}} = 160$. NB's high-confidence estimates enable model comparisons at extreme limits of failure. Low-dimensional visualization of the failure modes shows that the algorithms fail in distinct ways.

Table 1: Relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ over 10 trials

	Synthetic	MountainCar	Rocket1	Rocket2	AttentionAgentRacer	WorldModelRacer
MC	1.1821	0.2410	1.1039	0.0865	1.0866	0.9508
AMS	0.0162	0.5424	0.0325	0.0151	1.0211	0.8177
B	0.0514	0.3856	0.0129	0.0323	0.9030	0.7837
NB	**0.0051**	**0.0945**	**0.0102**	**0.0078**	**0.2285**	**0.1218**
p_γ	$3.6 \cdot 10^{-6}$	$1.6 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$\approx 2.5 \cdot 10^{-5}$	$\approx 9.5 \cdot 10^{-6}$

indicates that failures are dominated by high altitude and medium altitude gusts. Even though Rocket2 has a higher probability of failure, its failure mode is more concentrated than Rocket1's failures.

Car racing The CarRacing environment (Figure 4(a)) is a challenging reinforcement-learning task with a continuous action space and pixel observations. Similar observation spaces have been proposed for real autonomous vehicles (*e.g.* [7, 60, 103]). We compare two recent approaches, AttentionAgentRacer [98] and WorldModelRacer [43] that have similar average performance: they achieve average rewards of 903 ± 49 and 899 ± 46 respectively (mean ± standard deviation over 2 million trials). Both systems utilize one or more deep neural networks to plan in image-space, so neither has performance guarantees. We evaluate the probability of getting small rewards ($\gamma = 150$).

The 24-dimensional search space P_0 parametrizes the generation of the racing track (details are in Appendix C). This environment does not easily provide gradients due to presence of a rendering engine in the simulation loop. Instead, we fit a Gaussian process surrogate model to compute $\nabla f(x)$ (see Appendix C). As these experiments are extremely expensive (taking up to 1 minute per simulation), we only use 2 million naive Monte Carlo samples to compute the ground-truth failure rates. Figure 4(b) shows that, even though the two models have very similar average performance, their catastrophic failure curves are distinct. Furthermore, MC is unable to distinguish between the policies below rewards of 160 due to its high uncertainty, whereas NB clearly shows that WorldModelRacer is superior. Note that, because even the ground-truth has non-negligible uncertainty with 2 million samples, we only report the variance component of relative mean-square error in Table 1.

As with the rocket design experiments, we visualize the modes of failure (defined by $\gamma_{\text{test}} = 225$) via PCA in Figure 4(c). The dominant eigenvectors involve large differentials between radii and angles of consecutive checkpoints that are used to generate the racing tracks. AttentionAgentRacer has two distinct modes of failure, whereas WorldModelRacer has a single mode.

5 Conclusion

There is a growing need for rigorous evaluation of safety-critical systems which contain components without formal guarantees (*e.g.* deep neural networks). Scalably evaluating the safety of such systems in the presence of rare, catastrophic events is a necessary component in enabling the development of trustworthy high-performance systems. Our proposed method, neural bridge sampling, employs three concepts—exploration, exploitation, and optimization—in order to evaluate system safety with provable statistical and computational efficiency. We demonstrate the performance of our method on a variety of reinforcement-learning and robotic systems, highlighting its use as a tool for continuous integration and rapid engineering design. In future work, we intend to investigate how efficiently sampling rare failures—like we propose here for *evaluation*—could also enable the *automated repair* of safety-critical reinforcement-learning agents.

Broader Impact

This paper presents both foundational theory and methods for efficiently evaluating the performance of safety-critical autonomous systems. By definition, such systems can cause injury or death if they malfunction [15]. Thus, improving the tools that practitioners have to perform risk-estimation has the potential to provide a strong positive impact. On the other hand, the improved scalability of our method could be used to more efficiently find (zero-day) exploits and failure modes in P_0 (the model of the operational design domain). However, we note that adversarial examples or exploits can also be found via a variety of purely optimization-based methods [3]. The nuances of our method are primarily concerned with the frequency of adverse events, an extra burden; thus, we anticipate they will be of little interest to malicious actors who can manipulate the observations and sensor measurements of complex systems. Another potential concern about the use of our method is with respect to the identification of P_0, which we specifically assume to be known in this paper. The gap between P_0 in simulation and the real distribution of the environment could lead to overconfidence in the capabilities of the system under test. In Section 1.1 we outline complementary work in anomaly detection and distributionally robust optimization which could mitigate such risks. Still, more work needs to be done to standardize the operational domain of specific tasks by regulators and technology-stakeholders. Nevertheless, we believe that our method will enable the comparison of autonomous systems in a common language—risk—across the spectrum from engineers to regulators and the public.

The applications of our technology are diverse (*cf.* Corso et al. [27]), ranging from testing autonomous vehicles [76, 74] and medical devices [77] to evaluating deep neural networks [104] and reinforcement-learning agents [101]. In the case of autonomous vehicles, Sparrow and Howard [97] argue that it will be morally wrong not to deploy self-driving technology once performance exceeds human capabilities. Our work is an important tool for determining when this performance threshold is achieved due to the rare nature of serious accidents [51]. While the widespread availability of autonomy-enabled devices could narrowly benefit public health, there are many external risks associated with their development. First, many learning-based components of these systems will require massive and potentially invasive data collection [85]; preserving privacy of the public via federated learning [64] and differential privacy-based mechanisms [38] should remain important initiatives within the machine-learning community. A second potential negative consequence of the applications like autonomous vehicles is the use of the real-world as a “simulator” within a reinforcement-learning scheme by releasing “beta” autonomy features (*e.g.* Tesla Autopilot [52]). Unlike established industries such as aerospace [100], many potential applications currently lack regulation and standards; it is important to ensure that industry works with policy makers to develop safety standards in a way that avoids regulatory capture. If widely adopted in regulatory frameworks, our tool would enable rational decisions about the impact, positive or negative, of safety-critical autonomous systems before real lives are affected.

More broadly, the advent of autonomy could spark significant societal changes. For example, the autonomous applications described previously could become core components of weapons systems and military technology that are incompatible with (modern interpretations of) just war theory [96]. Similarly, the automation of the transportation industry has the potential to rapidly destroy the economics of public infrastructure and cost millions of jobs [97]. Thus, Benkler [9] highlights that there is a growing need for the academic community to take action on defining the broader performance criteria to which we will hold AI applications. Brundage et al. [18] and Wing [106] outline broad research agendas which are necessarily interdisciplinary. Still, much more work needs to be done to empower researchers to influence policy. These efforts will require systemic initiatives by research institutions and organizations to engage with local, national, and international governing bodies.

Acknowledgements

AS and JD were partially supported by the DAWN Consortium, NSF CAREER CCF-1553086, NSF HDR 1934578 (Stanford Data Science Collaboratory), ONR YIP N00014-19-2288, and the Sloan Foundation. MOK was supported by an NSF GRFP Fellowship. RT was supported by Lincoln Laboratory/Air Force Award No. PO# 7000470769 and Amazon Robotics Award No. CC MISC 00272683 2020 TR.

References

[1] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius. Functional gradient descent method for metric temporal logic specifications. In *2014 American Control Conference*, pages 2312–2317. IEEE, 2014.

[2] H. M. Afshar and J. Domke. Reflection, refraction, and hamiltonian monte carlo. In *Advances in neural information processing systems*, pages 3007–3015, 2015.

[3] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[4] M. Althoff. An introduction to cora 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[5] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

[6] K. J. Åström and P. Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[7] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[8] N. Baram, O. Anschel, I. Caspi, and S. Mannor. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pages 390–399, 2017.

[9] Y. Benkler. Don't let industry write the rules for ai. *Nature*, 569(7754):161–162, 2019.

[10] C. H. Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, 22(2):245–268, 1976.

[11] M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[12] C. M. Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[13] L. Blackmore. Autonomous precision landing of space rockets. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*. National Academies Press, 2017.

[14] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[15] J. Bowen and V. Stavridou. Safety-critical systems, formal methods and standards. *Software Engineering Journal*, 8(4):189–209, 1993.

[16] C.-E. Bréhier, T. Lelièvre, and M. Rousset. Analysis of adaptive multilevel splitting algorithms in an idealized case. *ESAIM: Probability and Statistics*, 19:361–394, 2015.

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[18] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong, et al. Toward trustworthy ai development: Mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020.

[19] J. Bucklew. *Introduction to rare event simulation*. Springer Science & Business Media, 2013.

[20] F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

[21] L. Chaari, J.-Y. Tourneret, C. Chaux, and H. Batatia. A hamiltonian monte carlo method for non-smooth energy sampling. *IEEE Transactions on Signal Processing*, 64(21):5585–5594, 2016.

[22] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[23] M.-H. Chen, Q.-M. Shao, and J. G. Ibrahim. *Monte Carlo methods in Bayesian computation*. Springer Science & Business Media, 2012.

[24] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification*, pages 258–263. Springer, 2013.

[25] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Fast mixing of metropolized hamiltonian monte carlo: Benefits of multi-step gradients. *arXiv preprint arXiv:1905.12247*, 2019.

[26] H. Choi, E. Jang, and A. A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

[27] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer. A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*, 2020.

[28] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[29] P. Del Moral. Feynman-kac formulae. In *Feynman-Kac Formulae*, pages 47–93. Springer, 2004.

[30] P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[31] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

[32] A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[33] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

[34] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification*, pages 432–442. Springer, 2019.

[35] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[36] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[37] A. Durmus, E. Moulines, et al. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

[38] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[39] P. M. Esfahani and D. Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *arXiv:1505.05116 [math.OC]*, 2015.

[40] J. M. Esposito, J. Kim, and V. Kumar. Adaptive rrts for validating hybrid robotic control systems. In *Algorithmic Foundations of Robotics VI*, pages 107–121. Springer, 2004.

[41] A. Gelman and X.-L. Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

[42] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2): 123–214, 2011.

[43] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[44] J. M. Hammersley and D. C. Handscomb. Monte carlo methods. 1964.

[45] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[46] T. C. Hesterberg and B. L. Nelson. Control variates for probability and quantile estimation. *Management Science*, 44(9):1295–1312, 1998.

[47] M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan. Neutralizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

[48] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178. ACM, 2019.

[49] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Case study: verifying the safety of an autonomous racing car with a neural network controller. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–7, 2020.

[50] H. Kahn and T. Harris. Estimation of particle transmission by random sampling. 1951.

[51] N. Kalra. *Challenges and Approaches to Realizing Autonomous Vehicle Safety*. RAND, 2017.

[52] A. Karpathy. Software 2.0. *Medium. com*, 2017.

[53] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv:1702.01135 [cs.AI]*, 1:1, 2017.

[54] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information*

processing systems, pages 4743–4751, 2016.

[55] O. Klimov. Carracing-v0. 2016. *URL https://gym. openai. com/envs/CarRacing-v0*.

[56] S. Kong, S. Gao, W. Chen, and E. Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015.

[57] S. Lan, B. Zhou, and B. Shahbaba. Spherical hamiltonian monte carlo for constrained target distributions. In *JMLR workshop and conference proceedings*, volume 32, page 629. NIH Public Access, 2014.

[58] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*, volume 14. Cambridge University Press, 2004.

[59] K. Leung, N. Arechiga, and M. Pavone. Back-propagation through stl specifications: Infusing logical structure into gradient-based methods.

[60] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[61] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[62] O. Mangoubi and A. Smith. Rapid mixing of hamiltonian monte carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*, 2017.

[63] A. W. Marshall. The use of multi-stage sampling schemes in monte carlo computations. Technical report, RAND CORP SANTA MONICA CALIF, 1954.

[64] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[65] X.-L. Meng and S. Schilling. Warp bridge sampling. *Journal of Computational and Graphical Statistics*, 11(3):552–586, 2002.

[66] X.-L. Meng and W. H. Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, pages 831–860, 1996.

[67] A. W. Moore. Efficient memory-based learning for robot control. Technical report, University of Cambridge, Computer Laboratory, 1990.

[68] B. Nachman and D. Shih. Anomaly detection with density estimation. *Physical Review D*, 101 (7):075042, 2020.

[69] S. Nakamoto and A. Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 2008.

[70] H. Namkoong and J. C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in neural information processing systems*, pages 2208–2216, 2016.

[71] R. M. Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

[72] R. M. Neal. Estimating ratios of normalizing constants using linked importance sampling. *arXiv preprint math/0511216*, 2005.

[73] R. M. Neal. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

[74] J. Norden, M. O'Kelly, and A. Sinha. Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*, 2019.

[75] M.-S. Oh and J. O. Berger. Adaptive importance sampling in monte carlo integration. *Journal of Statistical Computation and Simulation*, 41(3-4):143–168, 1992.

[76] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*, pages 9827–9838, 2018.

[77] M. O'Kelly, A. Sinha, J. Norden, and H. Namkoong. In-silico risk analysis of personalized artificial pancreas controllers via rare-event simulation. *arXiv preprint arXiv:1812.00293*, 2018.

[78] A. Pakman and L. Paninski. Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions. In *Advances in neural information processing systems*, pages 2490–2498, 2013.

[79] A. Pakman and L. Paninski. Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, 2014.

[80] Y. V. Pant, H. Abbas, and R. Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1235–1240. IEEE, 2017.

[81] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[82] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

[83] X. Qin, N. Aréchiga, A. Best, and J. Deshmukh. Automatic testing and falsification with dynamically constrained reinforcement learning. *arXiv preprint arXiv:1910.13645*, 2019.

[84] H. Rahimian and S. Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

[85] C. Ré. Software 2.0 and snorkel: beyond hand-labeled data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2876–2876, 2018.

[86] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1530–1538. JMLR. org, 2015.

[87] J. Ridderhof and P. Tsiotras. Minimum-fuel powered descent in the presence of random disturbances. In *AIAA Scitech 2019 Forum*, page 0646, 2019.

[88] G. O. Roberts and O. Stramer. Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

[89] P. J. Rossky, J. Doll, and H. Friedman. Brownian dynamics as smart monte carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 1978.

[90] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: A unified approach to Monte Carlo simulation, randomized optimization and machine learning*. Information Science & Statistics, Springer Verlag, NY, 2004.

[91] R. Y. Rubinstein and R. Marcus. Efficiency of multivariate control variates in monte carlo simulation. *Operations Research*, 33(3):661–677, 1985.

[92] B. Shahbaba, S. Lan, W. O. Johnson, and R. M. Neal. Split hamiltonian monte carlo. *Statistics and Computing*, 24(3):339–349, 2014.

[93] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[94] D. Siegmund. Importance sampling in the monte carlo study of sequential tests. *The Annals of Statistics*, pages 673–684, 1976.

[95] A. Sinha, H. Namkoong, and J. Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

[96] R. Sparrow. Killer robots. *Journal of applied philosophy*, 24(1):62–77, 2007.

[97] R. Sparrow and M. Howard. When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport. *Transportation Research Part C: Emerging Technologies*, 80:206–215, 2017.

[98] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. *arXiv preprint arXiv:2003.08165*, 2020.

[99] V. Tjeng and R. Tedrake. Verifying neural networks with mixed integer programming. *arXiv:1711.07356 [cs.LG]*, 2017.

[100] W. F. Tosney and P. G. Cheng. Space safety is no accident how the aerospace corporation promotes space safety. In *Space Safety is No Accident*, pages 101–108. Springer, 2015.

[101] J. Uesato, A. Kumar, C. Szepesvari, T. Erez, A. Ruderman, K. Anderson, N. Heess, P. Kohli, et al. Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures. *arXiv preprint arXiv:1812.01647*, 2018.

[102] A. F. Voter. A monte carlo method for determining free-energy differences and transition state theory rate constants. *The Journal of chemical physics*, 82(4):1890–1899, 1985.

[103] D. Wang, C. Devin, Q.-Z. Cai, P. Krähenbühl, and T. Darrell. Monocular plan view networks for autonomous driving. *arXiv preprint arXiv:1905.06937*, 2019.

[104] S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar. A statistical approach to assessing neural network robustness. 2018.

[105] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[106] J. M. Wing. Trustworthy ai. *arXiv preprint arXiv:2002.06276*, 2020.

[107] S. Yaghoubi and G. Fainekos. Falsification of temporal logic requirements using gradient based local search in space and time. *IFAC-PapersOnLine*, 51(16):103–108, 2018.

[108] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*, pages 1–10, 2014.

Algorithm 2 WarpedHMC

```
**Input:** Sample $x$, momentum $v \sim \mathcal{N}(0, I)$, transform $V_\theta$ and its inverse $W_\theta$, scale factor $\beta$, step size $\epsilon$
$y \leftarrow W_\theta(x)$
$\hat{v} \leftarrow v - 0.5\epsilon\beta I\{f(x) > \gamma\} J_{V_\theta}(y) \nabla f(x)$
$\hat{y} \leftarrow y\cos(\epsilon) + \hat{v}\sin(\epsilon)$
$\hat{v} \leftarrow \hat{v}\cos(\epsilon) - y\sin(\epsilon)$
$\hat{x} \leftarrow V_\theta(\hat{y})$
$\hat{v} \leftarrow \hat{v} - 0.5\epsilon\beta I\{f(\hat{x}) > \gamma\} J_{V_\theta}(\hat{y}) \nabla f(\hat{x})$
$\hat{v} \leftarrow -\hat{v}$
$x \leftarrow \hat{x}$ with probability $\min(1, \exp(-H(\hat{y}, \hat{v}) + H(y, v)))$
**Return** $x$
```

A Warped Hamiltonian Monte Carlo (HMC)

In this section, we provide a brief overview of HMC as well as the specific rendition, split HMC [92]. Given "position" variables x and "momentum" variables v, we define the Hamiltonian for a dynamical system as $H(x, v)$ which can usually be written as $U(x) + K(v)$, where $U(x)$ is the potential energy and $K(v)$ is the kinetic energy. For MCMC applications, $U(x) = -\log(\rho_0(x))$ and we take $v \sim \mathcal{N}(0, I)$ so that $K(v) = \|v\|^2/2$. In HMC, we start at state x_i and sample $v_i \sim \mathcal{N}(0, I)$. We then simulate the Hamiltonian, which is given by the partial differential equations:

$$\dot{x} = \frac{\partial H}{\partial v}, \quad \dot{v} = -\frac{\partial H}{\partial x}.$$

Of course, this must be done in discrete time for most Hamiltonians that are not perfectly integrable. One notable exception is when x is Gaussian, in which case the dynamical system corresponds to the evolution of a simple harmonic oscillator (*i.e.* a spring-mass system). When done in discrete time, a symplectic integrator must be used to ensure high accuracy. After performing some discrete steps of the system (resulting in the state (x_f, v_f)), we negate the resulting momentum (to make the resulting proposal reversible), and then accept the state $(x_f, -v_f)$ using the standard Metropolis-Hastings criterion: $\min(1, \exp(-H(x_f, -v_f) + H(x_i, v_i)))$ [45].

The standard symplectic integrator—the leap-frog integrator—can be derived using the following symmetric decomposition of the Hamiltonian (performing a symmetric decomposition retains the reversibility of the dynamics): $H(x, v) = U(x)/2 + K(v) + U(x)/2$. Using simple Euler integration for each term individually results in the following leap-frog step of step-size ϵ:

$$\begin{aligned} v_{1/2} &= v_i - \frac{\epsilon}{2}\frac{\partial U(x_i)}{\partial x} \\ x_f &= x_i + \epsilon\frac{\partial K(v_{1/2})}{\partial v} \\ v_f &= v_{1/2} - \frac{\epsilon}{2}\frac{\partial U(x_f)}{\partial x}, \end{aligned}$$

where each step simply simulates the individual Hamiltonian $H_1(x, v) = U(x)/2$, $H_2(x, v) = K(v)$, or $H_3(x, v) = U(x)/2$ in sequence. As presented by Shahbaba et al. [92], this same decomposition can be done in the presence of more complicated Hamiltonians. In particular, consider the Hamiltonian $H(x, v) = U_1(x) + U_0(x) + K(v)$. We can decompose this in the following manner: $H_1(x, v) = U_1(x)/2$, $H_2(x, v) = U_0(x) + K(v)$, and $H_3(x, v) = U_1(x)/2$. We can apply Euler integration to the momentum v for the first and third Hamiltonians and the standard leap-frog step to the second Hamiltonian (or even analytic integration if possible). For this paper, we have $U_0(x) = -\log \rho_0(x)$ and $U_1(x) = -\beta[\gamma - f(x)]_-$.

To account for warping, the modifications needed to the HMC steps above are simple. When performing warping, we simply perform HMC for a Hamiltonian $\hat{H}(y, v)$ that is defined with respect to the warped position variable y, where $x = V_\theta(y)$ for given parameters θ. By construction of the normalizing flows, we assume $y \sim \mathcal{N}(0, I)$, so that we can perform the dynamics for $\hat{H}_2(y, v)$ analytically. Furthermore, the Jacobian $J_{V_\theta}(y)$ is necessary for performing the Euler integration of $H_1(y, v)$ and $H_3(y, v)$. This is summarized in Algorithm 2. Note that we always perform the Metropolis-Hastings acceptance with respect to the true Hamiltonian H, rather than the Hamiltonian $\hat{H}$ that assumes perfect training of the normalizing flows.

HMC and non-smooth functions In Section 2, we assumed that the measure of non-differentiable points is zero for the energy potentials considered by HMC. As discussed by Afshar and Domke [2], the inclusion of the Metropolis-Hastings acceptance criterion as well as the above assumption ensures that HMC asymptotically samples from the correct distribution even for non-smooth potentials. An equivalent intuitive explanation for this can be seen by viewing the ReLU function $[x]_+$ as the limit of softplus functions $g_k(x) := \log(1 + \exp(kx))/k$ as the sharpness parameter $k \to \infty$. We can freely choose k such that, up to numerical precision, Algorithm 2 is the same whether we consider using a ReLU or sufficiently sharp (*e.g.* large k) softplus potential, because, with probability one, we will not encounter the points where the potentials differ. When further knowledge about the structure of the non-differentiability is known, the acceptance rate of HMC proposals can be improved [78, 57, 79, 2, 21].

B Performance analysis

B.1 Proof of Proposition 1

We begin with showing the convergence of the number of iterations. To do this, we first show almost sure convergence of β_k in the limit $N \to \infty$. We note that in the optimization problem (8), β_k is a feasible point, yielding $b_k(\beta) = 1$. Thus, $\beta_{k+1} \geq \beta_k \geq \beta_0 := 0$. Due to this growth of β_k with k, we have

$$\frac{Z_{k+1}}{Z_k} = \mathbb{E}_{P_k}\left[\frac{\rho_{k+1}(X)}{\rho_k(X)}\right] \leq 1,$$

$$\begin{aligned}\mathbb{P}_k(f(X) \leq \gamma) &= \mathbb{E}_{P_{k+1}}\left[\frac{Z_{k+1}}{Z_k}\frac{\rho_k(X)}{\rho_{k+1}(X)} I\{f(X) \leq \gamma)\}\right] \\ &= \frac{Z_{k+1}}{Z_k}\mathbb{E}_{P_{k+1}}\left[I\{f(X) \leq \gamma)\}\right] \\ &\leq \mathbb{P}_{k+1}(f(X) \leq \gamma).\end{aligned}$$

By the unfiorm convergence of empirical measures offered by the Glivenko-Cantelli Theorem, the value $a_k \to \mathbb{P}_k(f(X) \leq \gamma)$ almost surely. Then, the stop condition can be rewritten as $b_k(\beta) \geq a_k/s \to \mathbb{P}_k(f(X) \leq \gamma)/s \geq p_\gamma/s$. Since $b_k(\beta)$ is monotonically decreasing in the quantity $\beta - \beta_k$, this constraint gives an upper bound for β_{k+1}, and, as a result, all β_k are almost surely bounded from above and below. We denote this interval as $\mathcal{B}$.

Now, we consider the convergence of the solutions to the finite N versions of problem (8), denoted β_k^N, to the "true" optimizers β_k in the limit as $N \to \infty$. Leaving the dependence on β_k implicit for the moment, we consider the random variable $Y := g(X; \beta) := \exp\left((\beta - \beta_k)[\gamma - f(X)]_-\right)$. Then, since $\beta \in \mathcal{B}$ is bounded and g is continuous in β, we can state the Glivenko-Cantelli convergence of the empirical measure uniformly over $\mathcal{B}$: $\sup_{\beta\in\mathcal{B}} \|F^N(Y) - F(Y)\|_\infty \to 0$ almost surely, where F is the cumulative distribution function for Y. Note that the constraints in the problem (8) can be rewritten as expectations of this random variable Y. Furthermore, the function g is strictly monotonic in β (and therefore invertible) for non-degenerate $f(X)$ (*i.e.* $f(x) > \gamma$ for some non-negligible measure under P_0). Thus, we have almost sure convergence of the argmin β_{k+1}^N to β_{k+1}.

Until now, we have taken dependence on β_k implicitly. Now we make the dependence explicit to show the final step of convergence. In particular, we can write β_{k+1} as a function of β_k (along with their empirical counterparts), For concreteness, we consider the following decomposition for two iterations:

$$|\beta_2^N(\beta_1^N) - \beta_2(\beta_1)| \leq |\beta_2^N(\beta_1^N) - \beta_2(\beta_1^N)| + |\beta_2(\beta_1^N) - \beta_2(\beta_1)|.$$

We have already shown above that the first term on the right hand side vanishes almost surely. By the same reasoning, we know that $\beta_1^N \to \beta_1$ almost surely. The second term also vanishes almost surely since $\beta_{k+1}(\beta)$ is a continuous mapping. This is due to the fact that the constraint functions in problem (8) are continuous functions of both β and β_k along with the invertibility properties discussed previously. Then, we simply extend the telescoping series above for any k and similarly show that all terms vanish almost surely. This shows the almost sure convergence for all β_k up to some K.

Now we must show that K is bounded and almost surely converges to a constant. To do this we explore the effects of the optimization procedure. Assuming the stop condition (the second constraint) does not activate, the first constraint in problem (8) has the effect of making $\mathbb{Z}_{k+1}/Z_k = \alpha$ (almost surely), which implies $\mathbb{P}_{k+1}(f(X) \leq \gamma) = \mathbb{P}_k(f(X) \leq \gamma)/\alpha$. In other words, we magnify the event of interest by a factor of $1/\alpha$. The second constraint can be rewritten as $\mathbb{P}_{k+1}(f(X) \leq \gamma) \leq s$. Thus, we magnify the probability of the region of interest by factors of α unless doing so would increase the probability to greater than s. In that case, we conclude with setting the probability to s (since $\mathbb{P}_\beta(f(X) \leq \gamma)$ is monotonically increasing in β). In this way, we have 0 iterations for $p_\gamma \in [s, 1]$, 1 iteration for $p_\gamma \in [\alpha s, s)$, 2 iterations for $p_\gamma \in [\alpha^2 s, \alpha s)$, and so on. Then, the total number of iterations is (almost surely) $\lfloor \log(p_\gamma)/\log(\alpha) \rfloor + I\{p_\gamma/\alpha^{\lfloor \log(p_\gamma)/\log(\alpha) \rfloor} < s\}$.

Now we move to the relative mean-square error of $\hat{p}_\gamma$. We employ the delta method, whereby, for large N, this is equivalent to Var$(\log(\hat{p}_\gamma))$ (up to terms $o(1/N)$). For notational convenience, we decompose $\widehat{E}_k$ into its numerator and denominator:

$$A_k(X) := \rho_k^B(X)/\rho_{k-1}(X), \qquad \widehat{A}_k := \frac{1}{N}\sum_{i=1}^N A_k(x_i^{k-1})$$

$$B_k(X) := \rho_k^B(X)/\rho_k(X), \qquad \widehat{B}_k := \frac{1}{N}\sum_{i=1}^N B_k(x_i^k).$$

By construction (and assumption of large T), Algorithm 1 has a Markov property that each iteration's samples x_i^k are independent of the previous iterations' samples x_i^{k-1} given β_k. For shorthand, let $\beta_{0:k}$ denote all $\beta_0, \ldots, \beta_k$. Conditioning on $\beta_{0:k}$, we have

$$\text{Var}(A_k) = \text{Var}\left(\mathbb{E}[A_k|\beta_{0:k}]\right) + \mathbb{E}\left[\text{Var}\left(A_k|\beta_{0:k}\right)\right].$$

Since $\beta_{0:k}$ approaches constants almost surely as $N \to \infty$, the first term vanishes and the second term is the expectation of a constant. In particular, the second term is as follows:

$$\begin{aligned}
\text{Var}\left(A_k|\beta_{0:k}\right) &= \mathbb{E}\left[A_k^2|\beta_{0:k}\right] - \left(\mathbb{E}\left[A_k|\beta_{0:k}\right]\right)^2 \\
&= \mathbb{E}_{P_{k-1}}\left[\frac{\rho_k(X)}{\rho_{k-1}(X)}\right] - \left(\mathbb{E}_{P_{k-1}}\left[\sqrt{\frac{\rho_k(X)}{\rho_{k-1}(X)}}\right]\right)^2 \\
&= \frac{Z_k}{Z_{k-1}} - \left(\frac{Z_k^B}{Z_{k-1}}\right)^2.
\end{aligned}$$

Similarly, Var$(B_k|\beta_{0:k}) = Z_{k-1}/Z_k - (Z_k^B/Z_k)^2$. Next we look at the covariance terms:

$$\text{Cov}(A_{k-1}, A_k) = \text{Cov}\left(\mathbb{E}[A_{k-1}|\beta_{0:k}], \mathbb{E}[A_k|\beta_{0:k}]\right) + \mathbb{E}\left[\text{Cov}\left(A_{k-1}, A_k|\beta_{0:k}\right)\right].$$

Again, the first term vanishes since $\beta_{0:k}$ approach constants as $N \to \infty$. By construction, the second term is also 0 since the quantities are conditionally independent. Similarly, Cov$(B_{k-1}, B_k) = 0$ and Cov$(A_i, B_j) = 0$ for $j \neq i - 1$. However, there is a nonzero covariance for the quantities that depend on the same distribution:

$$\begin{aligned}
\text{Cov}\left(B_k, A_{k+1}|\beta_{0:k+1}\right) &= \mathbb{E}\left[B_k A_{k+1}|\beta_{0:k+1}\right] - \mathbb{E}\left[B_k|\beta_{0:k+1}\right]\mathbb{E}\left[A_{k+1}|\beta_{0:k+1}\right] \\
&= \mathbb{E}_{P_k}\left[\frac{\sqrt{\rho_{k-1}(X)\rho_{k+1}(X)}}{\rho_k(X)}\right] - \frac{Z_{k+1}^B}{Z_k}\frac{Z_k^B}{Z_k} \\
&= \frac{Z_k^C}{Z_k} - \frac{Z_{k+1}^B}{Z_k}\frac{Z_k^B}{Z_k}.
\end{aligned}$$

By the large T assumption, the samples x_i^k and x_j^k are independent for all $i \neq j$ given β_k. Then we have

$$\text{Var}(\widehat{A}_k|\beta_{0:k}) = \text{Var}(A_k|\beta_{0:k})/N, \;\; \text{Var}(\widehat{B}_k|\beta_{0:k}) = \text{Var}(B_k|\beta_{0:k})/N,$$

$$\text{Cov}(\widehat{B}_k, \widehat{A}_{k+1}|\beta_{0:k+1}) = \text{Cov}(B_k, A_{k+1}|\beta_{0:k+1})/N.$$

The last term in $\hat{p}_\gamma$, $\frac{1}{N}\sum_{i=1}^N \frac{\rho_\infty(x_i^K)}{\rho_K(x_i^K)}$, reduces to a simple Monte Carlo estimate since $\frac{\rho_\infty(X)}{\rho_K(X)} = I\{f(X) \leq \gamma\}$. Furthermore, this quantity is independent of all other quantities given $\beta_{0:K}$ and, as noted above, approaches s almost surely as $N \to \infty$.

Putting this all together, the delta method gives (as $N \to \infty$ so that $\beta_{0:K}$ approach constants almost surely),

$$\text{Var}(\log(\hat{p}_\gamma)) \to \sum_{k=1}^{K} \left(\frac{\text{Var}(\widehat{A}_k)}{(Z_k^B/Z_{k-1})^2} + \frac{\text{Var}(\widehat{B}_k)}{(Z_k^B/Z_k)^2} \right) - 2 \sum_{k=1}^{K-1} \frac{\text{Cov}(\widehat{B}_k, \widehat{A}_{k+1})}{Z_{k+1}^B Z_k^B / Z_k^2} + \frac{1-s}{sN} + o\left(\frac{1}{N}\right).$$

The Bhattacharrya coefficient can be written as

$$G(P_{k-1}, P_k) = \int_{\mathcal{X}} \sqrt{\frac{\rho_{k-1}(x)}{Z_{k-1}} \frac{\rho_k(x)}{Z_k}} dx = \frac{Z_k^B}{\sqrt{Z_{k-1} Z_k}}.$$

Furthermore, we have

$$\frac{G(P_{k-1}, P_{k+1})}{G(P_{k-1}, P_k) G(P_k, P_{k+1})} = \frac{Z_k^C}{\sqrt{Z_{k-1} Z_{k+1}}} \frac{\sqrt{Z_{k-1} Z_k}}{Z_k^B} \frac{\sqrt{Z_k Z_{k+1}}}{Z_{k+1}^B} = \frac{Z_k^C Z_k}{Z_k^B Z_{k+1}^B},$$

yielding this final result

$$\text{Var}(\log(\hat{p}_\gamma)) \to \frac{2}{N} \sum_{k=1}^{K} \left(\frac{1}{G(P_{k-1}, P_k)^2} - 1 \right) - \frac{2}{N} \sum_{k=1}^{K-1} \left(\frac{G(P_{k-1}, P_{k+1})}{G(P_{k-1}, P_k) G(P_k, P_{k+1})} - 1 \right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right). \quad (12)$$

We remark that a special case of this formula is for $K = 1$ and $s = 1$ (so only the first term survives), which is the relative mean-square error for a single bridge-sampling estimate $\widehat{E}_k$.

Now, since $G(P, Q) \geq 0$, the terms in the second sum are ≥ -1 so that the second sum is $\leq 2(K-1)/N$. Furthermore, since $s \geq 1/3$, the last term is also $\leq 2/N$. Thus, if we have $\frac{1}{G(P_{k-1}, P_k)^2} \leq D$ (with $D \geq 1$), then the asymptotic relative mean-square error (12) is $\leq 2KD/N$ (up to terms $o\left(\frac{1}{N}\right)$).

When performing warping, we follow the exact same pattern as the above results, conditioning on both $\beta_{0:k}$ and $W_{0:k}$, where W_0 is defined as the identity mapping. We follow the same almost-sure convergence proof for W_k as above for β_k, which requires compactness of $\theta \in \Theta$, continuity of W with respect to θ and x, and that we actually achieve the minimum in problem (6). Although the first two conditions are immediate in most applications, the last condition can be difficult to satisfy for deep neural networks due to the nonconvexity of the optimization problem.

C Experimental setups

C.1 Hyperparameters

The number of samples N affects the absolute performance of all of the methods tested, but not their relative performance with respect to each other. For all experiments, we use $N = 1000$ for B and NB to have adequate absolute performance given our computational budget (see below for the computing architecture used). Other hyperparameters were tuned on the synthetic problem and fixed for the rest of the experiments (with the exception of the MAF architecture for the rocket experiments). The hyperparameters were chosen as follows.

When performing Hamiltonian dynamics for a Gaussian variable, a time step of 2π results in no motion and time step of π results in a mode reversal, where both the velocity and position are negated. The π time step is in this sense the farthest exploration that can occur in phase space (which can be intuitively understood by recognizing that the phase diagram of a simple spring-mass system is a unit circle). Thus, we considered $T = 4, 8, 12,$ and 16 with time steps π/T. We found that $T = 8$ provided reasonable exploration (as measured by autocorrelations and by the bias of the final estimator $\hat{p}_\gamma$) and higher values of T did not provide much more benefit. For B, we allowed 2 more steps $T = 10$ to keep the computational cost the same across B and NB. Similarly, for AMS, we set $T = 10$. We also performed tuning online for the time step to keep the accepatance ratio between 0.4 and 0.8. This was done by setting the time step to $\sin^{-1}(\min(1, \sin(t)\exp((p - C)/2))$, where t is the current time step, p is the running acceptance probability for a single chain and $C = 0.4$ if $p < 0.4$ or 0.8 if $p > 0.8$. This was done after every T HMC steps.

For the step size of the bridge, we considered $\alpha \in \{0.01, 0.1, 0.3, 0.5\}$. Smaller α results in fewer iterations and better computational efficiency. However, we found that very small α made MAF training difficult (see below for the MAF architectures used). We settled on $\alpha = 0.3$, which provided reasonable computational efficiency (no more than 11 iterations for the synthetic problem) as well as stable MAF training. For AMS, we followed the hyperparameter settings of Webb et al. [104]. Namely, we chose a culling fraction of $\alpha_{\text{AMS}} = 10\%$, where α_{AMS} sets the fraction of particles that are removed and rejuvenated at each iteration [104].

The MAF architectures for the synthetic, MountainCar, and CarRacing experiments were set at 5 MADE units, each with 1 hidden layer of 100 neurons. Because the rocket search space is very high dimensional, we decreased the MAF size for computational efficiency: we set it at 2 MADE units, each with hidden size 400 units. We used 100 epochs for training, a batch size of 100, a learning rate of 0.01 and an exponential learning-rate decay with parameter 0.95.

Given the above parameters, the number of simulations for each experiment varies based on the final probability in question p_γ (smaller values result in more simulations due to having a higher number of iterations K). We had runs of 111000, 101000, 91000, 71000, 91000, and 101000 simulations respectively for the synthetic, MountainCar, Rocket1, Rocket2, AttentionAgentRacer, and WorldModelRacer environments. We used these values as well as the ground truth p_γ values to determine the number of particles allowed for AMS, $N_{\text{AMS}} = 920, 910, 820, 780, 820, 910$ respectively, as AMS has a total cost of $N_{\text{AMS}}(1 + \alpha_{\text{AMS}} T K_{\text{AMS}})$, where $K_{\text{AMS}} \approx \log(p_\gamma)/\log(1 - \alpha_{\text{AMS}})$.

For the surrogate Gaussian process regression model for CarRacing, we retrained the model on the most recent N simulations after every NT simulations (*e.g.* after every T HMC iterations). This made the amortized cost of training the surrogate model negligible compared to performing the simulations themselves. We used a Matern kernel with parameter $\nu = 2.5$. We optimized the kernel hyperparameters using an L-BFGS quasi-Newton solver.

Computing infrastructure and parallel computation Experiments were carried out on commodity CPU cloud instances, each with 96 Intel Xeon cores @ 2.00 GHz and 85 GB of RAM. AMS, B, and NB are all designed to work in a Map-Reduce paradigm, where a central server orchestrates many worker jobs followed by synchronization step. AMS requires more iterations and fewer parallel worker threads per iteration than B and NB. In particular, whereas B and NB perform N parallel jobs per iteration, AMS only performs $\alpha_{\text{AMS}} N_{\text{AMS}}$ parallel jobs per iteration. Thus, B and NB take advantage of massive scale and parallelism much more than AMS.

C.2 Environment details

C.2.1 MountainCar

The MountainCar environment considers a simple car driving on a mountain road. The car can sense horizontal distance s as well as its velocity v, and may send control inputs u (the amount of power applied in either the forward or backward direction). The height of the road is given by: $h(s) = 0.45 \sin(3s) + 0.55$. The speed of the car, v, is a function of s and u only. Thus, the discrete time dynamics are: $s_{k+1} = s_k + v_{k+1}$ and $v_{k+1} = v_k + 0.0015 u_k - 0.0025 \cos(3 s_k)$. For a given episode the agent operating the car receives a reward of $-0.1 u_k^2$ for each control input and 100 for reaching the goal state.

In this experiment we explore the effect of domain shift on a formally verified neural network. We utilize the neural network designed by Ivanov et al. [48]; it contains two hidden layers, each of 16 neurons, for a total of 337 parameters. For our experiments we use the trained network parameters available at: `https://github.com/Verisig/verisig`. Ivanov et al. [48] describe a layer-by-layer approach to verification which over-approximates the reachable set of the combined dynamics of the environment and the neural network. An encoding of this system (network and environment) is developed for the tool Flow* [24] which constructs the (overapproximate) reachable set via a Taylor approximation of the combined dynamics.

The MountainCar environment is considered solved if a policy achieves an average reward of 90 over 100 trials. The authors instead seek to prove that the policy will achieve a reward of at least 90 for any initial condition. By overapproximating the reachable states of the system, they show that the car always receives a total reward greater than 90 and achieves the goal in less than 115 steps for a subset of the intial conditions $\hat{p}_0 \in [-0.59, -0.4]$.

C.2.2 Rocket design

The system under test is a rocket spacecraft with dynamics $m\ddot{p} = f - mge_3$, where $m > 0$ is the mass, $p(t) \in \mathbf{R}^3$ is the position, and e_3 is the unit vector in the z-direction. While it is possible to synthesize optimal trajectories for an idealized model of the system, significant factors such as wind and engine performance (best modeled as random variables) are unaccounted for [13]. Without feedback control, even small uncorrected tracking errors result in loss of the vehicle. In the case of disturbances the authors suggest two approaches: (1) a feedback control law which tracks the optimal trajectory (2) receding horizon model predictive control. The system we consider tracks an optimal trajectory using a feedback control law. Namely, the optimal trajectory is given by the minimum fuel solution to a linearized mode of the dynamics. Specifically, we consider the thrust force discretized in time with a zero-order hold, such that f_k applied for time $t \in [(k-1)h, kh]$ for a time step $h = 0.2$. Then, the reference thrust policy solves the following convex optimization problem

$$\begin{aligned} \text{minimize} & \sum_{i=1}^{K} \|f_k\|_2 \\ \text{such that } & p_K = v_K = 0, \|f_k\| \leq F_{\max}, \\ & v_{k+1} - v_k = \frac{h}{m} f_k - hge_3, \\ & p_{k+1} - p_k = \frac{h}{2}\left(v_k + v_{k+1}\right), \\ & (p_3)_k \geq 0.5\|((p_1)_k, (p_2)_k)\|_2, \end{aligned}$$

where the last constraint is a minimum glide slope and $F_{\max}$ is a maximum thrust value for the nominal thrusters. This results in the thrust profile $f^\star$. The booster thrusters correct for disturbances along the flight. The disturbances at every point in time follow a mixture of Gaussians. Namely, we consider 3 wind gust directions, $w_1 = (1,1,1)/\sqrt(3)$, $w_2 = (0,1,0)$, and $w_3 = (1,0,0)$. For every second in time, the wind follows a mixture:

$$W \sim \mathcal{N}(0, I) + w_1 B + w_2 \hat{B} + (1 - \hat{B}) w_3,$$

where $B \sim$ Bernoulli$(1/3)$ and $\hat{B} \sim$ Bernoulli$(1/2)$. This results in 5 random variables for each second, or a total of 100 random variables since we have a 20 second simulation. The wind intensity experienced by the rocket is a linear function of height (implying a simplistic laminar boundary layer): $f_w = CWp_3$ for a constant C. Finally, the rocket has a proportional feedback control law for the booster thrusters to the errors in both the position p_k and velocity v_k:

$$f_{\text{feedback,k}} = \text{clip-by-norm}(f_k^\star - K_p(p_k - p_k^\star) - K_v(v_k - v_k^\star)).$$

The maximum norm for clip-by-norm is aF_{max}, where $a = 1.15$ for Rocket1 and $a = 1.1$ for Rocket2, indicating that the boosters are capable of providing 15% or 10% of the thrust of the main engine.

C.2.3 Car Racing

We compare the failure rate of agents solving the car-racing task utilizing the two distinct approaches ([43] and [98]). The car racing task differs from the other experiments due to the inclusion of a (simple) renderer in the system dynamics. At each the step the agent recieves a reward of $-0.1 + \mathcal{I}_{newtile}(1000/N) - \mathcal{I}_{offtrack}(100)$ where N is the total number of tiles visited in the track. The environment is considered solved if the agent returns an average reward of 900 over 100 trials. The search space P_0 is the inherent randomness involved with generating a track. The track is generated by selecting 12 checkpoints in polar coordinates, each with radian value uniformly in the interval $[2\pi i/12, 2\pi(i+1)/12)$ for $i = 0, \ldots 11$, and with radius uniformly in the interval $[R/3, R]$, for a given constant value R. This results in 24 parameters in the search space. The policies used for testing are described below (with training scripts in the code supplement).

AttentionAgent Tang et al. [98] utilize a simple self-attention module to select patches from a 96x96 pixel observation. First the input image is normalized then a sliding window approach is used to extract N patches of size $M \times M \times 3$ which are flattened and arranged into a matrix of size $3M^2 \times N$. The self-attention module is used to compute the attention matrix A and importance vector (summation of each column of A). A feature extraction operation is applied to the top K elements of the sorted importance vector and the selected features are input to a neural network controller. Both the attention module and the controller are trained together via CMA-ES. Together, the two modules contain approximately 4000 learnable parameters. We use the pre-trained model available here: https://github.com/google/brain-tokyo-workshop/tree/master/AttentionAgent.

WorldModel The agent of Ha and Schmidhuber [43] first maps a top-down image of the car on track via a variational autoencoder to a latent vector z. Given z, the world model M utilizes a recurrent-mixture density network [12] to model the distribution of future possible states $P(z_{t+1} \mid a_t, z_t, h_t)$. Note that h_t, the hidden state of the RNN. Finally, a simple linear controller C maps the concatenation of z_t and h_t to the action, a_t. We use the pre-trained model available here: https://github.com/hardmaru/WorldModelsExperiments/tree/master/carracing.