SAMPLING AND ESTIMATION FOR THE VALIDATION OF SAFETY-CRITICAL AUTONOMOUS SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Harrison Delecki
May 2025

This dissertation is online at: https://purl.stanford.edu/tz122mx1710

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mykel Kochenderfer, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Art Owen

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mac Schwager

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Abstract

There is increasing interest in deploying autonomous systems to domains such as transportation, healthcare, finance, and robotics because of their potential to improve efficiency and safety. These domains are safety-critical; the consequences of system failures can cause loss of human life or other significant costs. Therefore, we require rigorous validation of autonomous systems before deployment in these domains. Traditional safety validation approaches like real-world testing may be extremely expensive and time-consuming. Formal verification techniques are not scalable to complex systems and environments. Falsification methods provide failure examples in simulation, but they do not provide a comprehensive understanding of the system's behavior.

In this thesis, we address these limitations by proposing techniques for failure sampling and estimation to validate safety-critical autonomous systems. Given a probabilistic model of system trajectories, our methods aim to sample from the distribution over trajectories that lead to failure (failure sampling) or to estimate the probability of failure (failure estimation). During system development, engineers may use failure sampling to identify failure modes and improve system safety. Failure estimation can be used to quantify the risk of system failure before deployment. Together, these approaches provide a comprehensive understanding of system behavior and enable rigorous safety validation. Previous methods for failure sampling and estimation have been based on Markov chain Monte Carlo, importance sampling, and dynamic programming. Although these methods have been successful in some applications, they have limitations in terms of generality, scalability, and efficiency.

Many existing methods for failure sampling and estimation are tailored to specific system models and failure criteria. This makes it difficult to apply these methods in new settings. We present a framework for failure sampling as probabilistic inference. To achieve this, we formulate the simulation of the system under test as a probabilistic program. The probabilistic program easily interfaces with general-purpose inference algorithms, making failure sampling techniques more accessible. Additionally, the probabilistic program easily incorporates some problem-specific knowledge like gradient information, which improves the efficiency of failure sampling.

Autonomous systems may have high-dimensional trajectories and exhibit complex failure modes, making failure sampling challenging. Furthermore, many industry-level systems only provide input-output access, meaning that algorithms cannot use any internal system knowledge like gradients to improve scalability. We propose a method for failure sampling in these high-dimensional black-box settings using deep generative models. Our method adaptively trains a diffusion model to sample trajectories that lead to failure. This method scales to problems with thousands of dimensions and multiple failure modes.

Methods for estimating the probability of failure often require many samples to achieve accurate estimates. Since failure events are rare and simulations may be complex, this can be computationally expensive. We propose a state-dependent importance sampling method that improves the efficiency of failure estimation. Our method learns a proposal distribution that builds trajectories in a state-dependent manner. The training process focuses on trajectories that are closer to failure.

The sampling and estimation algorithms in this work are tested on a variety of problems. A simple inverted pendulum system is used to illustrate important concepts and algorithms. A rule-based autonomous driving policy is tested in a scenario with a pedestrian in a crosswalk. Finally, we validate a model of the ground collision-avoidance autopilot of the F-16 aircraft. Through experiments, we show that the presented methods can improve the generality, scalability, and efficiency of failure sampling and estimation for safety validation of autonomous systems.

Acknowledgments

I am profoundly grateful to the many individuals who made this research possible. The journey to completing this thesis has been shaped by countless contributions of knowledge, time, and encouragement from mentors, colleagues, friends, and family.

First, I would like to thank my advisor Mykel Kochenderfer for providing me with the opportunity to be a part of SISL and for his incredible mentorship in research and in life. Thank you for inspiring me to think more critically and creatively, teaching me to communicate more effectively, and showing me how to live a life of kindness and enthusiasm. This thesis would not have been possible without your guidance. I simply could not imagine a better advisor or mentor. I would also like to thank my reading committee, Mac Schwager and Art Owen, and my defense committee, Scott Linderman and Somil Bansal. Thank you for your time, wisdom, and feedback.

I want to thank all of the sponsors and industry partners who made this research possible, including NASA, Ford, Nissan, Motional, and the National Science Foundation. Thank you to Jerry Lopez, who was my first contact at Motional and who has been a great supporter of my work. I also want to thank Marcell Vazquez-Chanlatte for sharing his expertise and deep technical knowledge with me.

I would like to thank my incredible fellow SISL members and Stanford collaborators. Each one of you has taught me something unique about research and life. I would like to thank my coauthors who ventured into the unknown with me on the ideas in this thesis: Anthony Corso, Marc Schlichting, Mansur Arief, and Sydney Katz. Anthony, your mentorship has been instrumental in my development as a researcher. Thank you for being available to work through problems and brainstorm

new ideas. Sydney, thank you for bringing your unique clarity of thought to our discussions and for helping me think bigger when it comes to communication. I would also like to thank Robert Moss, Liam Kruse, Esen Yel, Joshua Ott, Dylan Asmar, Bernard Lange, and Duncan Eddy who gave me invaluable advice and feedback on this research.

I am fortunate to have had many mentors that shaped my path towards this thesis. I would like to thank Jodi Ramirez, whose early guidance helped me navigate the beginnings of my scientific journey with confidence and curiosity. I would also like to thank Joseph Saleh for his mentorship during my undergraduate studies. Dr. Saleh instilled in me a deep appreciation for the study of safety-critical systems long before I knew I would pursue a PhD in this field.

To my girlfriend Emily, thank you not only for your support through the research process but also for ensuring I maintained perspective and balance in life outside the lab. Thank you for listening to me talk through the numerous failed experiments, celebrating successes of new ideas, and being silly and nerdy with me. Your own academic passion inspires me daily, and I'm excited to support you as you begin your PhD journey. I am so grateful to have you in my life.

Last but not least, I would like to thank my family. Thank you to my brother, who has always been there to share a laugh and listen to how things are going. Thank you to my parents, who have supported me in every step of my journey. I am so grateful for your love and encouragement. Thank you for encouraging me to pursue my passions and for always believing in me. Mom, thank you for always being there to listen, for your endless caring and reassuring presence, and for your unwavering support of my passions. Dad, thank you for encouraging me to work hard in the pursuit of my dreams, for sharing your love of science and engineering, and for always being there to help me when I needed it. I love you all.

Contents

1 Introduction

There is increasing interest in deploying autonomous systems to safety-critical domains, such as autonomous driving, aviation, and medicine. In these domains, the consequences of operational errors can be significant, including loss of property and life. It is critical to ensure the safe operation of autonomous systems before deployment, but this remains challenging due to the complexity of these systems, the environments in which they operate, and the ways they may fail. This thesis presents efficient and scalable methods for comprehensive failure analysis. In this first chapter, we discuss the safety validation of autonomous systems, arguing that failure sampling and probability estimation are key for comprehensive understanding of system safety. We present the challenges of safety validation, our contributions, and the outline of the thesis.

1.1 Safety Validation of Autonomous Systems

Autonomous systems in safety-critical domains have the potential to improve safety and efficiency. Autonomous aircraft collision avoidance systems, implemented following a series of devastating mid-air collisions [1], have been shown to reduce the already rare risk of such incidents by at least a factor of three [2]. The next generation of aircraft collision avoidance systems will improve safety and function in the higher-density airspaces of the future [3]. In the automotive domain, nearly 40,000 fatalities and over 2 million injuries occur annually due to traffic accidents in the United States [4]. Recent studies from Waymo suggest that their autonomous vehicles reduce injury-related crash rates by 80% [5].

Despite the promise of autonomous systems, significant risks remain in their deployment to safety-critical domains. The 2018 crash of an autonomous vehicle that killed a pedestrian in Tempe, Arizona, is a tragic example of the potential consequences of failures in autonomous systems [6]. In this incident, the vehicle observed a jaywalker but struggled to correctly classify the pedestrian as a person, leading to highly variable predictions about their future path. Due to the delay in classification and uncertain predictions, the vehicle did not apply emergency braking in time to avoid the fatal collision. In 2023, the California Department of Motor Vehicles suspended testing permits for the robotaxi company Cruise after incidents involving collisions or near misses with pedestrians [7]. Cruise's robotaxi operations were then shut down by its parent company in 2024. Another example of the potential risk of autonomous systems comes from aircraft collision avoidance systems. It was discovered through testing that the collision avoidance system could induce a small number of near-misses that would otherwise not have occurred [2]. However, the frequency of these near-misses was much lower than the frequency of mid-air collisions that the system prevented. These examples highlight the importance of understanding system weaknesses during development and risks prior to deployment.

To mitigate these risks, safety is incorporated into each stage of the development cycle. Initially, engineers may define system-level requirements that pose constraints on a system's behavior, such as maximum allowable failure probability or severity. These top-level requirements flow down to constrain the lower-level design of the system, such as perception and control. Using methods like certifiable perception [8], control barrier functions [9], or safe sets [10], engineers may design the system to be safe by construction. During early stages of development, engineers may use safety validation to search for potential failure modes and to ensure that the system meets its safety requirements. Later, validation methods may assist in deciding whether the system is ready for deployment.

Safety validation is performed on a prototype of the autonomous system. Here, we describe three broad classes of approaches to safety validation prior to deployment, including real-world testing, simulation-based testing, and formal verification.

This thesis focuses on *simulation-based testing*, where the system is tested in a simulated environment. Next, we discuss simulation-based testing and two other broad categories of safety validation.

1. **Real-world testing** involves testing the system in the real operational environment. This is the most direct way to evaluate the system's safety. Real-world testing is often used to validate the system's performance in the final stages of development. For autonomous vehicles, companies typically conduct extensive on-road testing with safety drivers [11]. Similarly, aircraft collision avoidance systems undergo rigorous flight tests in controlled airspace before certification [3], [12]. The main drawbacks of real-world testing include high operating costs, potential danger, limited coverage of edge cases, and the inability to easily reproduce specific scenarios.

2. **Simulation-based testing** involves evaluating the system in a simulated environment. This is a cost-effective way to evaluate the system's safety, allowing for comprehensive testing across numerous scenarios that would be impractical or dangerous to test in the real world [13]. However, simulation-based testing requires a high-fidelity simulation environment to ensure that insights transfer to real-world performance. Simulation-based testing often involves simulating the autonomous system in a stochastic model of the environment to discover failures. These simulations can be accelerated beyond real-time and parallelized across computing resources. The failures discovered in simulation may be used by engineers to inform decisions about the system's design or requirements. This process of collecting failures to inform future decisions is part of a process sometimes called *failure analysis*. Simulation-based approaches include falsification techniques that search for counterexamples to safety specifications [14], most-likely failure analysis that identifies the most probable failure scenarios, and failure probability estimation that quantifies the overall risk of system failure. One key challenge of simulation-based testing is to ensure that the simulated environment accurately represents the real-world environment.

3. **Formal verification** techniques provide mathematical proofs about the system's safety under a set of assumptions [15]. The system must be represented as a formal mathematical model such as a finite state machine, computer code, or neural network. Model checking is a common formal verification technique where a model of the system is exhaustively checked against a safety specification, typically expressed in temporal logic. For example, a model checker might verify that an autonomous vehicle controller always maintains a safe distance from other vehicles under all possible inputs within the model's constraints [16]. Recent advances have made formal verification more practical for certain components of autonomous systems, such as neural network controllers [17], [18]. Another approach is deductive verification, where a proof of safety is constructed by constructing a sequence of intermediate proofs about the behavior of the system [19]. Formal verification provides the strongest safety guarantees. However, the main drawback of formal verification is that it may not scale well to complex systems or those with large, continuous input spaces. Additionally, these methods may face challenges in potential gaps between verified models and their implementations [20].

All of these approaches to safety validation are important and may be used in conjunction to ensure the safety of the autonomous system. In this thesis, we focus on simulation-based testing. For the remainder of this thesis, we will use the term *validation* to refer to simulation-based testing.

1.2 *Safety Validation Tasks*

Simulation-based safety validation encompasses three complementary tasks that together provide a comprehensive safety assessment. These tasks represent different approaches for finding specific failure trajectories, characterizing the full distribution of failures, and quantifying failure probability. The results from these tasks help engineers identify system weaknesses, prioritize improvements, and make informed deployment decisions. We describe each task in more detail below.

Falsification. Falsification is the task of finding a counterexample to a specification. In the context of simulation-based testing, falsification involves finding a simulated trajectory that violates a safety requirement due to the stochastic components of the simulation. This process may focus on objectives like finding the most likely or the most severe failure example. The drawback of falsification is that it aims to discover a single failure trajectory, which may not be representative of the system's behavior.

Failure distribution sampling. Failure sampling aims to discover the distribution of failure events. Rather than finding just a single failure example, this approach attempts to characterize the space of failures by generating multiple failure samples weighted by their likelihood of occurrence. The output of this task is a collection of diverse failure events and their corresponding probabilities, providing insight into the different ways a system might fail. These samples can reveal distinct failure modes, which may require different mitigation strategies during system development. For example, in an autonomous vehicle system, failure sampling might reveal scenarios where the vehicle fails to detect pedestrians in low-light conditions, as well as separate scenarios where sensor noise causes unnecessary emergency braking. By sampling across the full distribution, engineers can identify the most common or concerning failure patterns, estimate the relative frequency of different failure types, and better understand the overall risk landscape. This comprehensive view enables more targeted improvements than what could be achieved through isolated counterexamples alone.

Failure probability estimation. Failure probability estimation aims to estimate the probability of a failure event occurring. This task involves computing the expected value of the indicator function that determines whether a simulated trajectory violates the safety specification. For safety-critical autonomous systems, failure probabilities are often required to be extremely low (e.g., 10^{-9} per hour of operation), making direct Monte Carlo estimation computationally intractable. Advanced techniques such as importance sampling and cross-entropy methods can provide more efficient estimates by focusing sampling efforts on regions of the input space

that are more likely to produce failures. The output of failure probability estimation is a numerical value that can be directly compared against safety requirements or certification standards. This quantitative measure is particularly valuable for regulatory approval and for making objective comparisons between different designs.

This thesis focuses on failure sampling and probability estimation because they provide a comprehensive evaluation of system safety and perform complementary roles during development. Failure sampling provides a distribution over failures, which designers may use to identify failure modes and take corrective action. While falsification provides only isolated examples, failure sampling reveals the full spectrum of potential failure scenarios, including their relative likelihoods. Probability estimation complements this by providing a quantitative measure of the system's overall safety, which may be used to compare different designs or to evaluate compliance with safety requirements. Together, these techniques enable both qualitative understanding of how a system might fail and quantitative assessment of how likely such failures are.

1.3 Challenges of Sampling and Estimation

There are many potential challenges in performing failure sampling and probability estimation for safety-critical systems. These challenges include generalization of algorithms to varying systems, scaling to very large problems, and sample efficiency. Although not discussed in this thesis, the challenge of system modeling and model mismatch is also a key challenge for safety validation. Next, we discuss these challenges in more detail.

Modeling and model mismatch. Simulation-based testing relies on creating computational models of the physical environment. If the model is not accurate, the results of the simulation may not accurately reflect the real-world behavior of the system. Accurate modeling of the physical world may require modeling large numbers of physical processes, like the dynamics of the autonomous system, the behaviors of other agents, and environment sensors. For example, a realistic simulation for an

autonomous vehicle may require modeling the dynamics of the vehicle, the decision-making of other vehicles, and the properties of camera sensors. In practice, these modeling challenges are often addressed by making simplified assumptions about the physical processes, constructing models from first principles, or by learning models from real-world data. Although there are many open challenges in creating accurate models, this thesis does not address the challenge of modeling and model mismatch. Instead, we assume there exists a sufficiently good simulation of the environment. When deploying systems in the real world, engineers may discover distribution shifts between the simulation and the real world using anomaly detection or out-of-distribution detection methods [21], [22]. These methods can be deployed online to flag situations where the system may be operating outside of its expected domain and potentially trigger safety measures. However, these methods are not the focus of this thesis.

Generalization. After creating a simulation for validation, engineers must formulate the validation task and choose an appropriate validation algorithm. A key challenge in failure sampling and probability estimation is developing algorithms that generalize across different types of autonomous systems and failure modes. Due to the complexity of safety-critical systems, existing sampling approaches often make restrictive assumptions or are tailored to specific applications, limiting their broader applicability. For example, algorithms designed for collision avoidance systems may rely on specific properties of vehicle dynamics or sensor models that do not translate to other domains [23]. The challenge is compounded by the inherent multimodality of failure distributions. Autonomous systems may fail in diverse and unexpected ways that are difficult to capture with a single sampling strategy, especially when the system involves machine learning-based components[24]. A self-driving car might fail due to sensor noise in some cases, control errors in others, or complex interactions between multiple subsystems. This lack of generalizability means that engineers may need to significantly modify existing approaches or develop entirely new sampling methods for their specific validation problems, making systematic safety validation more challenging and time-consuming. There is a need for more

flexible validation frameworks that can be readily adapted to different systems while effectively exploring diverse failure modes.

Scalability. A fundamental challenge in validation is discovering system failures due to variations in the environment over time. Autonomous systems operate by taking sequences of actions that form system trajectories. Therefore, failures are often associated with entire trajectories rather than individual states or actions. For example, a collision in an autonomous driving system may result from a complex sequence of sensor observations, vehicle state estimates, and control actions spanning multiple seconds. The validation task requires searching over trajectories of inputs to find those that lead to failure. This search becomes extremely challenging in modern autonomous systems due to both the dimensionality of the input space and the length of the time horizon. Consider an autonomous vehicle that processes high-dimensional sensor data like camera images – even modest trajectories of 10 timesteps with 100-dimensional inputs at each step create a 1000-dimensional search space. This curse of dimensionality makes exhaustive search intractable and causes many validation algorithms to scale poorly. For instance, methods based on Monte Carlo sampling become inefficient as the dimension grows, requiring an exponential number of samples to maintain coverage [25]. Similarly, optimization-based approaches struggle to explore high-dimensional spaces effectively, often converging to suboptimal local minima [26]. In practice, this challenge is often addressed by leveraging domain knowledge to reduce the search space or by making simplifying assumptions about the structure of inputs over time, though these workarounds can limit the generality of the validation.

Sample efficiency. Quantifying the probability of failure is crucial for deployment decisions. For example, the Federal Aviation Administration places requirements on the failure probability for aviation systems before deployment[27]. However, this presents a significant computational challenge. Safety-critical systems are often designed to be inherently safe, making failures extremely rare events. Consider an aircraft collision avoidance system that must demonstrate a failure rate below

10^{-9} per flight hour to meet certification requirements. Validation may require high-fidelity simulations, which are often computationally expensive. Using Monte Carlo sampling to verify this requirement would require over a billion flight hours of simulation to observe even a single failure on average, making this approach computationally intractable. More sophisticated sampling approaches like importance sampling can help by focusing computation on regions more likely to contain failures, but these methods often struggle with the high dimensionality and complex dynamics of autonomous systems.

The methods presented in this thesis make progress towards more general applicability, scalability, and efficiency for failure sampling and probability estimation. Overall, the goal of these methods is to provide engineers with safety validation tools to help develop safe autonomous systems.

1.4 Contributions

The methods presented in this thesis directly address these critical challenges of generalization, scalability, and sample efficiency in failure sampling and probability estimation for autonomous systems. We summarize the key contributions below.

Failure sampling as inference. To address the problem of generalization, we introduce a formulation of failure sampling as a Bayesian inference problem. We show how validation tasks like falsification and failure probability estimation can be unified under a single probabilistic framework. This formulation enables the use of powerful probabilistic inference algorithms that have been developed in other domains to serve as off-the-shelf algorithms for safety validation.

Failure sampling with generative models. We address challenges of scalability by proposing a failure sampling method that uses state-of-the-art generative models to represent the complex, high-dimensional distribution over trajectories. Our method adaptively trains a conditional diffusion model to generate disturbances that bring the system to a specified safety level (closeness to failure). Through iterative training

where each round's samples add to training data for the next, we guide the model toward the failure region. This strategy enables efficient failure sampling in systems with thousands of dimensions, where traditional methods often struggle.

Probability estimation using state-dependent importance sampling. To address the challenge of sample efficiency in failure probability estimation, we propose a method that uses state-dependent importance sampling. Rather than modeling the full distribution over failure trajectories directly, we decompose the problem by learning a proposal distribution that depends on the current system state. This decomposition dramatically reduces the dimensionality of the sampling problem while maintaining the ability to estimate failure probabilities accurately.

1.5 Outline

The rest of the thesis is organized as follows.

Chapter 2 formulates the problem of safety validation and presents notation used throughout the thesis. It also provides an overview of existing methods for failure sampling and probability estimation. The motivations, benefits, and challenges of these methods are discussed.

Chapter 3 describes three systems used as case studies throughout the thesis. The first example is a simple inverted pendulum problem, which is used to illustrate the core concepts of safety validation. The second example is a more complex autonomous driving system. The final example is a ground collision avoidance system for the F-16 aircraft, which features high input dimensionality and a long time horizon.

Chapter 4 presents a formulation of failure sampling as probabilistic inference. We present a general probabilistic framework to construct probabilistic programs for safety validation given a simulator and system under test. This formulation allows us easily incorporate system information like gradients. Furthermore, this formulation allows use to use any generic Bayesian inference algorithm for failure sampling. We demonstrate the benefits of the formulation on the inverted pendulum

and autonomous driving systems.

Chapter 5 presents a method for failure sampling that uses generative models to represent the distribution over failures in very high dimensional settings. Our algorithm adaptively trains a generative model to generate system disturbances that lead to failure. We demonstrate the effectiveness of this approach on several robotics validation problems, including an aircraft ground collision avoidance system with over 1000 dimensions.

Chapter 6 presents a method for efficiently estimating the probability of failure events in autonomous systems. We propose an adaptive importance sampling method that learns a state-dependent proposal distribution to reduce the dimensionality of the sampling problem.

Finally, Chapter 7 provides a summary of the contributions and discusses remaining challenges in failure sampling and probability estimation. We also discuss potential future directions that may address these challenges.

2 Background

In this chapter, we formulate the problem of safety validation in terms of a distribution over system trajectories that lead to failure. We introduce notation and describe the goals of the sampling and estimation tasks for safety validation with respect to the failure distribution. Finally, we review existing approaches for failure sampling and estimation, including Markov chain Monte Carlo (MCMC) methods, importance sampling, and dynamic programming.

2.1 Problem Formulation

System and environment. Suppose we have an autonomous system under test, which we will refer to as the *system*. The system takes actions $a \in A$ in an *environment* with state $s \in S$. The system selects actions based on observations $o \in O$ of the environment. We assume that interactions between the system and environment take place over a discrete time with finite horizon $t \in \{1, \ldots, d\}$. We denote the system's state trajectory up to time t as $s_{1:t} = [s_1, \ldots, s_t]$. Similarly, we denote the system's action trajectory up to time t as $a_{1:t} = [a_1, \ldots, a_t]$ and observation trajectory as $o_{1:t} = [o_1, \ldots, o_t]$.

We model the system as a distribution or policy that maps observation trajectories to actions, $a_t \sim \pi(\cdot \mid o_{1:t})$. As a concrete example, consider a self-driving car system that takes actions to control the car based on sensor observations of the environment. Note that this action may depend on all observations up until the current timestep. Similarly, we model the imperfect observations generated by sensors as a distribution known as an *observation model*, $o_t \sim O(\cdot \mid s_t)$. Finally, we

model potentially stochastic simulations using a transition model $s_{t+1} \sim T(\cdot \mid s_t, a_t)$. The state of the environment may include the position of other cars, pedestrians, and traffic signals. The observations may include images from cameras. Actions may correspond to steering and acceleration commands. Note that we generally assume that the transitions and observations only depend on the current states and actions. This is known as the *Markov property*. Given the current state, we can simulate the system under test to the next time step by sampling

$$o_t \sim O(\cdot \mid s_t), \qquad a_t \sim \pi(\cdot \mid o_{1:t}), \qquad s_{t+1} \sim T(\cdot \mid s_t, a_t). \tag{2.1}$$

We denote a simulated trajectory up to horizon d as $\tau = (s_1, a_1, o_1, \ldots, s_d, a_d, o_d)$.

Safety specifications. Safety specifications on the system describe the conditions under which the system is considered to be operating safely. These specifications are typically defined in terms of constraints on the state trajectories. For example, a safety specification for a self-driving car might require that the car maintains a safe distance from other vehicles and pedestrians at all times. These specifications may be designed ad hoc or written in a formal language like signal temporal logic [28]. We define the set of trajectories that satisfy the safety specification as ψ.Therefore, we write $\tau \notin \psi$ to denote an unsafe state trajectory. Often these specifications come with the ability to calculate a *robustness* value, which quantifies how close a system trajectory is to violating the specification. We denote the robustness metric of a state trajectory as $\rho(\tau)$, and failure occurs when $\rho(\tau) \leq 0$.

To facilitate validation, we often assume that *disturbances* x directly control the sources of randomness in the environment transitions, sensor observations, and agent actions. For example, the disturbances may directly correspond to the sensor noise or acceleration of other agents. Formally, we convert each distribution in the system simulation to a deterministic function that takes the disturbance as an input:

$$o_t = O(s_t, x_t), \qquad a_t = \pi(o_{1:t}, x_t), \qquad s_{t+1} = T(s_t, a_t, x_t) \tag{2.2}$$

Given a state s_t and disturbance x_t, we can simulate the system to the next time step

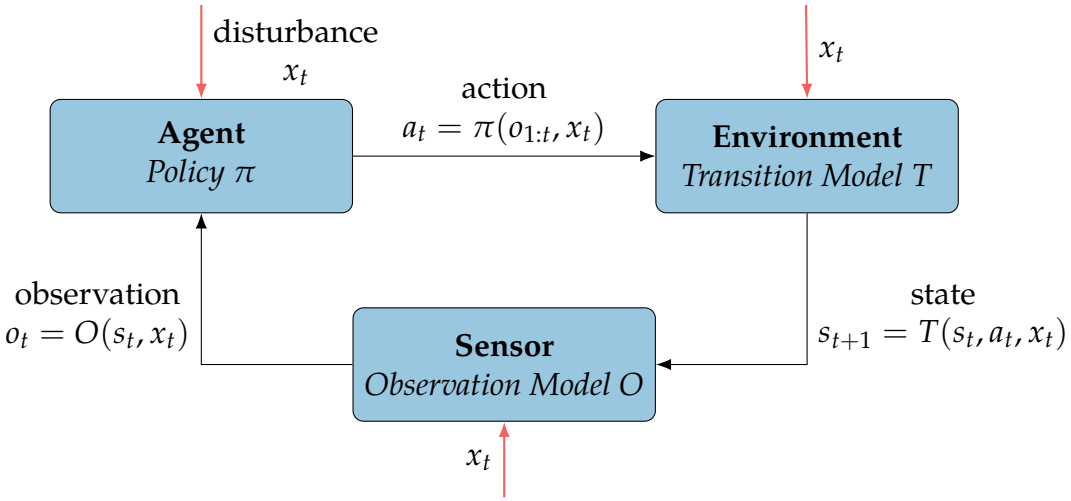


Figure 2.1: An agent takes actions in an environment after receiving observations through sensors. We inject disturbances to control sources of randomness in control, simulation, and observation.

by computing:

$$o_t \sim O(\cdot \mid s_t, x_t), \qquad a_t \sim \pi(\cdot \mid o_{1:t}, x_t), \qquad s_{t+1} \sim T(\cdot \mid s_t, a_t, x_t) \tag{2.3}$$

This formulation is illustrated in Figure 2.1. By simulating up to the end of the time horizon, we can generate a system trajectory $\tau = (s_1, a_1, o_1, x_t, \ldots, s_d, a_d, o_d, x_d)$. This formulation provides a deterministic way to simulate the system behavior under the influence of disturbances. Given a disturbance trajectory $x_{1:d}$, we denote the simulation of the system under this deterministic formulation as

$$\tau = f(x_{1:d}) \tag{2.4}$$

where $f(x_{1:d})$ simulates the system trajectory under the influence of disturbances $x_{1:d}$. In practice, this formulation may require engineering the ability to control randomness into existing simulators and sensor models if they are stochastic.

Disturbance models. We assume that the simulated system has an associated probabilistic model of disturbances. In the most general case, these disturbances may

depend on the current state, action, and observation. We denote this state-dependent *disturbance model* as $D(x_t \mid s_t, a_t, o_t)$. This means that we can compute the density of a trajectory τ as:

$$p(\tau) = p(s_1) \prod_{t=1}^{d} D(x_t \mid s_t, a_t, o_t) \tag{2.5}$$

where $p(s_1)$ is the initial state distribution. The disturbances may be independent of the state, action, and observation, in which case the distribution over system trajectories simplifies to:

$$p(\tau) = p(s_1) \prod_{t=1}^{d} D(x_t) \tag{2.6}$$

The disturbance model should capture the distribution of disturbances that the system is likely to encounter in the operational environment. Consequently, $p(\tau)$ will capture the distribution over system trajectories that we expect to see when the system is deployed. For this reason, we call $p(\tau)$ the *nominal trajectory distribution*.

The disturbance model can be either learned from data or specified by domain experts. The quality of safety validation results depends heavily on how accurately this model captures the actual distribution of disturbances in the operational environment. If the disturbance model is inaccurate, the safety validation results may be less representative of real-world failure events. However, even with an imperfect disturbance model, safety validation can still be used to understand potential weaknesses and vulnerabilities.

2.2 *Sampling and Estimation for Safety Validation*

In this section, we formalize three different tasks commonly encountered in safety validation: falsification, failure sampling, and estimation.

Falsification. The goal of falsification is to find a single disturbance trajectory that leads to a violation of the safety specification:

$$\text{Find} \quad \tau \quad \text{s.t.} \quad \tau \notin \psi \tag{2.7}$$

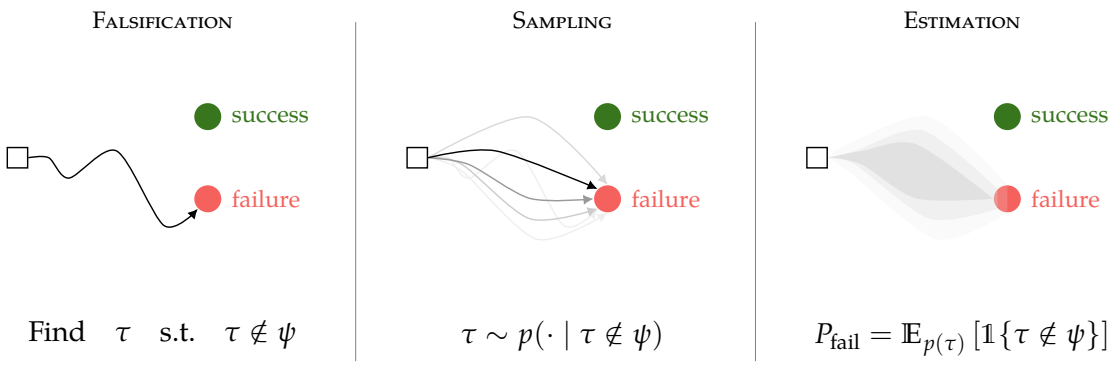


Figure 2.2: Three safety validation tasks. This thesis focuses on the sampling and estimation tasks.

Failure sampling. The goal of failure sampling is to sample from the distribution of disturbances that lead to a violation of the safety specification. This sampling problem is:

$$\tau \sim p(\cdot \mid \tau \notin \psi) \tag{2.8}$$

Failure probability estimation. In the failure probability estimation task, the goal is to estimate the probability that a system trajectory violates the safety specification. This probability of failure P_{fail} is an expectation over the nominal trajectory distribution:

$$P_{\text{fail}} = \mathbb{E}_{p(\tau)}[\mathbb{1}\{\tau \notin \psi\}] \tag{2.9}$$

These tasks are listed in order of increasing difficulty and utility for safety validation. Failure sampling is generally more challenging than falsification, as it requires sampling from the distribution of disturbances that lead to failures. To estimate the probability of failure, we must discover many likely failure examples. Therefore, if we can solve the failure probability task, we also solve the failure sampling and falsification tasks.

In general, the failure probability estimation task provides the most utility for safety validation, as it allows us to quantify the risk of a system violating safety specifications. However, each task may produce useful results at various stages of development. For example, falsification may be useful in the early stages of

development to identify potential safety issues. Failure sampling may be useful for generating a diverse set of failure examples for further analysis.

2.3 Existing Approaches

Both failure sampling and failure probability estimation tasks require sampling from the distribution of disturbances that lead to a violation of safety specifications. In this section, we review existing approaches for sampling and estimation.

Challenges. Algorithms for sampling and estimation face three key challenges. First, failures in safety-critical systems tend to be relatively rare events. Simple methods like Monte Carlo sampling, which draws trajectories directly from $p(\tau)$, may require an enormous number of samples to discover a failure. Consider a system with a failure probability of 10^{-6}. Sampling even one failure event with a Monte Carlo method would require 10^6 samples on average, and obtaining a reliable estimate of the failure probability would require many more. Second, the search space over failure trajectories is often high-dimensional. The space of disturbance trajectories scales exponentially with the size of the disturbance space and the trajectory length, so an exhaustive search is intractable. Third, the distribution of disturbances that lead to failures may be multimodal, with multiple distinct regions of the disturbance space that lead to failures. Reliably sampling multimodal distributions is often challenging, especially when the modes are not known a priori, as is often the case in safety validation [29].

2.3.1 Monte Carlo

A simple approach to sample failures and estimate the failure probability is using Monte Carlo (MC) sampling, which uses N independent samples from $p(\tau)$. After simulating these samples, we can approximate the distribution over failure trajectories by simply *rejecting* the samples that do not lead to failure. Additionally, we can estimate the failure probability by counting the number of samples that lead to

failure. The MC estimator of the failure probability is:

$$\hat{P}_{\text{fail}}^{\text{MC}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\tau_i \notin \psi\} \tag{2.10}$$

The MC estimator is unbiased, with variance that depends on the true failure probability:

$$\text{Var}[\hat{P}_{\text{fail}}^{\text{MC}}] = \frac{P_{\text{fail}}(1 - P_{\text{fail}})}{N} \tag{2.11}$$

For safety-critical systems, the true failure probability may be as low as 10^{-9}, making MC estimation infeasible. Sampling one failure trajectory would require 10^9 samples on average. Estimating the failure probability to within 1% accuracy would require 10^{11} samples. For these reasons, MC is often impractical for safety validation of complex systems. Instead, we may need to use more sophisticated methods that can more efficiently sample from the failure distribution.

2.3.2 *Markov Chain Monte Carlo*

Markov chain Monte Carlo (MCMC) methods generate samples from a target distribution by constructing a Markov chain whose stationary distribution matches the target distribution [29]. For safety validation, the target distribution is typically the failure distribution $p(\tau \mid \tau \notin \psi)$. MCMC methods begin with an initial sample τ_0 and iteratively propose new samples τ' from a proposal distribution that only depends on the previous sample $q(\cdot \mid \tau)$. The new sample is either accepted or rejected based on a probability that depends on the target distribution. If the new sample is accepted, it becomes the next sample in the Markov chain. The Markov chain is run for a large number of iterations, and the samples are used to estimate properties of the target distribution. The Markov chain is designed so that samples are asymptotically distributed according to the target distribution as the number of iterations approaches infinity.

One of the most common MCMC methods is the Metropolis-Hastings (MH) algorithm. Given a current sample τ, the algorithm proposes a new sample τ' from

a proposal distribution $q(\tau' \mid \tau)$. The proposed sample is accepted with probability:

$$\alpha = \min\left(1, \frac{p(\tau' \mid \tau' \notin \psi)q(\tau \mid \tau')}{p(\tau \mid \tau \notin \psi)q(\tau' \mid \tau)}\right) \tag{2.12}$$

Acceptance step. The proposal for Metropolis-Hastings is often chosen to be symmetric, meaning that $q(\tau' \mid \tau) = q(\tau \mid \tau')$. In this case, the acceptance probability simplifies to:

$$\alpha = \min\left(1, \frac{p(\tau' \mid \tau' \notin \psi)}{p(\tau \mid \tau \notin \psi)}\right) \tag{2.13}$$

Note that we do not need to know the normalizing constant of the target distribution, as it cancels out in the acceptance probability.

Limitations. While MCMC methods can theoretically sample from any target distribution, they face practical challenges in safety validation. The efficiency of MCMC methods depends heavily on the choice of proposal distribution. Poor proposal distributions can lead to slow mixing and long convergence times. Additionally, MCMC methods may struggle with multimodal failure distributions, as the Markov chain may have difficulty moving between modes.

2.3.3 *Importance Sampling*

Importance sampling can reduce the variance of failure probability estimates by focusing samples on regions of interest. The key idea is to sample from an alternative distribution $q(\tau)$ that makes failures more likely to occur. The distribution $q(\tau)$ is sometimes called a *proposal distribution*. The importance sampling estimate of the failure probability is:

$$\hat{P}_{\text{fail}}^{\text{IS}} = \frac{1}{N}\sum_{i=1}^{N} w(\tau_i)\mathbb{1}\{\tau_i \notin \psi\} \tag{2.14}$$

where the samples are weighted according to their importance weight $w(\tau) = p(\tau)/q(\tau)$. The estimator is unbiased if $q(x) > 0$ whenever $p(\tau)\mathbb{1}\{\tau \notin \psi\} > 0$.

The variance of the importance sampling estimator is:

$$\mathrm{Var}[\hat{P}_{\text{fail}}^{\text{IS}}] = \frac{1}{N}\mathbb{E}_{q(\tau)}\left[\frac{(p(\tau)\mathbb{1}\{\tau \notin \psi\} - q(\tau)P_{\text{fail}})^2}{q(\tau)}\right] \tag{2.15}$$

A good importance sampling distribution should minimize the variance of the estimator so that fewer samples are required to obtain a reliable estimate. The optimal importance sampling distribution, which minimizes the variance of the estimator, is proportional to the failure distribution itself:

$$q^\star(\tau) = \frac{p(\tau \mid \tau \notin \psi)}{P_{\text{fail}}} \tag{2.16}$$

In practice, it is impossible to sample directly from the optimal importance sampling distribution because it depends on the unknown failure probability. Instead, algorithms that use importance sampling often seek to estimate the optimal importance sampling distribution.

The choice of the proposal can significantly impact the efficiency of importance sampling for probability estimation. In practice, a poor choice of proposal can lead to estimates with much larger variance than direct Monte Carlo. The choice of proposal becomes especially difficult in high-dimensional, multimodal distributions. It is challenging to ensure adequate coverage of all potential failure modes, particularly when these modes are separated by low-probability regions within a high-dimensional parameter space. Poor proposals can be diagnosed by examining the importance weights of the samples. If the weights are very large, meaning that the proposal distribution is very different from the target distribution, the variance of the estimator will be high. In practice, Kim et al. [30] recommend clipping the importance weights to prevent the estimator from becoming unstable.

Adaptive multilevel splitting (AMS) [31], [32] is a method for rare event sampling and estimation that approximates the optimal importance sampling distribution using MCMC samples. AMS works by iteratively updating a group of samples using MCMC to be closer to the failure region. Previously, AMS has been used to evaluate the safety of autonomous driving policies [33], [34], to approximate the

probability of conflict in air traffic management [35], and to evaluate the robustness of neural networks for perception tasks [36].

One of the most popular methods for failure sampling and estimation is the Cross-Entropy Method (CEM) [37], [38]. CEM iteratively updates a parametric proposal distribution by minimizing the KL divergence to the optimal importance sampling distribution. Starting from a parametric distribution $q_\theta(\tau)$ where θ are the parameters, CEM solves the following optimization problem:

$$\theta^* = \arg\min_\theta D_{KL}(p(\tau \mid \rho(\tau) \leq 0) \parallel q_\theta(\tau)) \tag{2.17}$$

$$= \arg\max_\theta \int p(\tau) \mathbb{1}\{\rho(\tau) \leq 0\} \log q_\theta(\tau) d\tau \tag{2.18}$$

$$= \arg\max_\theta \mathbb{E}_{p(\tau)} \left[\mathbb{1}\{\rho(\tau) \leq 0\} \log q_\theta(\tau)\right] \tag{2.19}$$

$$= \arg\max_\theta \mathbb{E}_{q_\theta(\tau)} \left[\mathbb{1}\{\rho(\tau) \leq 0\} \frac{p(\tau)}{q_\theta(\tau)} \log q_\theta(\tau)\right] \tag{2.20}$$

$$\approx \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\rho(\tau) \leq 0\} \frac{p(\tau)}{q_\theta(\tau)} \log q_\theta(\tau) \tag{2.21}$$

When the proposal is a member of the exponential family, (e.g Gaussian), the optimization problem can be solved analytically. One challenge with this procedure is that because failure events are rare, all samples may be safe with $\rho(\tau) > 0$. In this case, the proposal will never converge to the failure region.

One solution is to iteratively update a target robustness threshold γ_k at iteration k to ensure that a minimum fraction α of samples are failures. The resulting algorithm takes the following steps:

1. Sample $\{\tau_1, \ldots, \tau_N\}$ from $q_{\theta_k}(\tau)$
2. Sort the samples by robustness $\rho(\tau_i)$
3. Update the threshold γ_k to be the α-th percentile of the robustness values
4. Set θ_{k+1} by solving the optimization problem with the new threshold
5. Repeat until $\gamma_k = 0$

CEM has been used to evaluate safety for aircraft collision avoidance systems [30] and autonomous driving policies [23], [39].

2.3.4 Dynamic Programming

Dynamic programming approaches for failure sampling and estimation make use of the temporal structure of the system to efficiently sample from the failure distribution. These methods often rely on observing the state of the environment during simulation to construct a state-dependent proposal distribution over disturbances. One of the key assumptions is that the state of the environment and system is *Markovian*, or that the current state contains all relevant information about the future. This assumption allows the method to construct a proposal that depends only on the current state of the system. In practice these dynamic programming problems can be formulated as decision-making problems, and may be solved using appropriate methods like value iteration or Q-learning [40], [41]. Dynamic programming methods can be effective for small, discrete problems. However, the dimensionality of the problem grows exponentially with the time horizon, making dynamic programming methods difficult to apply to long time horizons or high-dimensional problems. Dynamic programming methods have been used to evaluate the safety of autonomous driving policies in multi-agent scenarios [42] and to evaluate the safety of aircraft collision avoidance systems [43].

2.3.5 Strengths and weaknesses

Each category of approaches to sampling and estimation has its own strengths and weaknesses. Here, we summarize the key points for each approach.

Monte Carlo. Monte Carlo methods are simple, easy to implement, and require no knowledge about the environment state. However, they may be inefficient for rare failure events. Direct Monte Carlo sampling can require an enormous number of samples to find a failure, making it impractical for validation of complex systems.

Markov chain Monte Carlo. MCMC methods can theoretically sample from any target distribution. Like MC, they generally do not require knowledge of the environment state, which may be unavailable due to privacy or implementation concerns. Methods that rely on MCMC can be effective for very complex distributions because MCMC samples can represent the full distribution. However, MCMC methods may struggle with multimodal failure distributions, and the efficiency of MCMC methods depends heavily on the choice of proposal distribution.

Importance sampling. Importance sampling approaches require finding many failure examples to estimate the optimal importance sampling distribution. The efficiency of importance sampling depends heavily on the choice of proposal distribution. In high-dimensional, complex problems, it may be difficult to design or learn a good proposal distribution.

Dynamic programming. Dynamic programming methods can be effective for small, discrete problems where the state space is manageable. However, the dimensionality of the problem grows exponentially with the time horizon, making dynamic programming methods difficult to apply to problems with high-dimensional state and disturbance spaces.

2.4 Discussion

This chapter introduced a model of safety validation for autonomous systems. The system takes actions in an environment after receiving observations of the environment state through a sensor. We assume that the agent, environment, and sensor can be influenced through stochastic disturbances. The system and environment interact over a discrete time horizon, and the goal of safety validation is to find disturbances that lead to a violation of safety specifications.

We introduced three tasks for safety validation: falsification, failure sampling, and failure probability estimation. We argue that the failure sampling and estimation tasks are the most useful for safety validation, as they provide a distribution of

failures and a probability of failure, respectively. We reviewed existing approaches for failure sampling and estimation, including Monte Carlo, Markov chain Monte Carlo, importance sampling, and dynamic programming.

In the following chapter we introduce three sample systems that will be used to evaluate different sampling and estimation approaches in later chapters. These systems will help us understand the strengths and weaknesses of each approach.

3 Example Systems

In this chapter, we provide a mathematical formulation of safety validation problems for sampling and estimation. Next, we present example problem formulations for three different systems. First, we consider a simple inverted pendulum. The agent tries to keep a pendulum balanced vertically while being subjected to torque disturbances. The second system is a autonomous vehicle that tries to avoid colliding with a jaywalking pedestrian. The disturbances to the AV are the pedestrian's motion and the AV's sensor noise. Third, we describe a Ground Collision Avoidance (GCAS) autopilot for the F-16 aircraft. The aircraft must avoid collision with the ground while being subject to sensor noise.

3.1 Inverted Pendulum

Throughout this thesis, we consider the inverted pendulum as a simple, illustrative example system for safety validation. The inverted pendulum system is a classic nonlinear control problem. Figure 3.1 shows the system consisting of a pendulum that is free to rotate about a pivot point. The goal is to keep the pendulum balanced in the upright position by applying a torque to the pivot point. The state of the system is the angle θ of the pendulum with respect to the vertical axis and the angular velocity $\dot{\theta}$. The control input a is the applied torque. In this example, we consider a controller that perfectly observes the state of the system. The dynamics

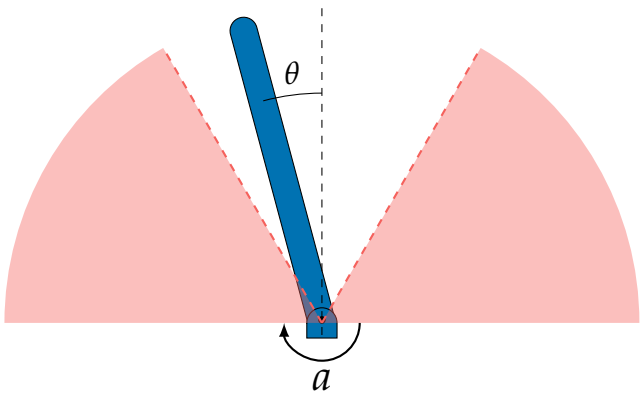


Figure 3.1: Inverted pendulum system. The agent controls the torque applied to the pivot point a to keep the pendulum balanced. Failure occurs when the angle of the pendulum exceeds a threshold, shown by the red shaded region.

of the system are given by the following equations of motion:

$$\dot{\theta}_{t+1} = \dot{\theta}_t + \left(\frac{3g}{2l} \sin\theta + \frac{3a}{ml^2} \right) \Delta t \tag{3.1}$$

$$\theta_{t+1} = \theta_t + \dot{\theta}_{t+1} \Delta t \tag{3.2}$$

where g is the acceleration due to gravity, l is the length of the pendulum, m is the mass of the pendulum, and Δt is the time step. The magnitude of the torque is bounded by $a_{\max}$. The initial state of the pendulum is always set to $\theta = 0$ and $\dot{\theta} = 0$.

Throughout this thesis, we will use additive torque disturbances on this system. The disturbances at each time step are drawn from a Gaussian distribution with mean 0 and standard deviation σ. The disturbance is added to the control input a before applying it to the system. This type of disturbance models the uncertainty in the control input due actuator noise or environmental factors like wind.

We note that the system is underactuated, meaning that the control input a does not have full control authority. If the angle of the pendulum exceeds the threshold, the force of gravity will cause the pendulum to fall over. More formally, the safety specification for the inverted pendulum requires that the magnitude of the pendulum's angle $|\theta|$ remains below a threshold $\theta_{\max}$ for all time steps.

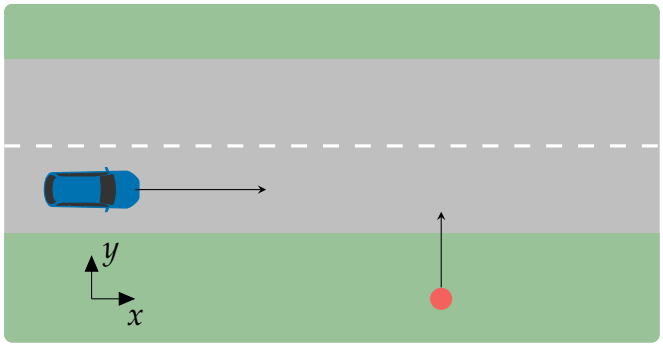


Figure 3.2: Autonomous vehicle and pedestrian scenario. The autonomous vehicle (AV) must avoid colliding with a jaywalking pedestrian. The disturbances are the AV's sensor noise and the pedestrian's motion.

3.2 *Autonomous Vehicle*

Autonomous driving has become an especially important domain for safety validation due to recent deployments of self-driving cars [44], [45], [46]. Safety validation of real-world autonomous vehicles is challenging due to the size of the state and disturbance spaces, complex interactions between agents, and rarity of failure events.

We consider a simplified scenario where an autonomous vehicle (AV) must avoid colliding with a jaywalking pedestrian. This scenario is pictured in Figure 3.2. The AV is equipped with sensors that provide noisy measurements of the pedestrian's position, heading, and speed. The AV must control its acceleration to avoid colliding with the pedestrian. There is only one lane for the AV to drive in, so the state of the AV can be represented by the longitudinal position $(r_x^{\text{AV}}, v_x^{\text{AV}})$. The state of the pedestrian is the lateral and longitudinal position and velocity $(r_x^{\text{ped}}, r_y^{\text{ped}}, v_x^{\text{ped}}, v_y^{\text{ped}})$. At each time step, both agents compute their acceleration. Then, their states are updated according to the following kinematic equations:

$$r_{t+\Delta t} = r_t + v_t \Delta t + \frac{1}{2} a_t \Delta t^2 \tag{3.3}$$

$$v_{t+\Delta t} = v_t + a_t \Delta t \tag{3.4}$$

where Δt is the simulation time step.

Table 3.1: Default IDM parameter values.

Parameter	Description	Default Value
k	Speed tracking constant	$1.0\,\mathrm{s}^{-1}$
δ	Acceleration exponent	4.0
T_{des}	Desired time headway	1.5 s
r_{min}	Minimum allowed gap	5.0 m
v_{des}	Desired velocity	15.0 m/s
a_{max}	Maximum acceleration	$3.0\,\mathrm{m/s}^2$
d_{comf}	Comfortable deceleration	$2.0\,\mathrm{m/s}^2$
d_{max}	Maximum deceleration	$9.0\,\mathrm{m/s}^2$

The AV's driving policy is based on the Intelligent Driver Model (IDM) [47], a rule-based model that describes car-following behavior. The IDM applies longitudinal acceleration to reach a desired velocity while maintaining a safe distance from obstacles. The IDM algorithm is shown in algorithm 3.1. The IDM takes as input the current distance to the lead vehicle Δr, the AV's velocity v_{ego}, and the lead vehicle's velocity v_{lead}. If there is no leading object ($c_{\mathrm{lead}} = 0$), the IDM accelerates the AV to reach its desired velocity. If there is a leading object, the IDM computes the desired gap r_{des} and computes the acceleration based on the desired gap and relative velocity. The IDM acceleration is bounded by the AV's maximum acceleration a_{max} and maximum deceleration d_{max}. The IDM default parameters are shown in Table 3.1.

Failures in this scenario occur when the AV collides with the pedestrian. The safety specification requires that the distance between the AV and the pedestrian remains above a threshold r_{min} for all time steps. The disturbances are the lateral and longitudinal acceleration of the pedestrian (a_x, a_y), and the lateral and longitudinal noise in the measurements of the pedestrian's position and speed (n_x, n_y, n_v). We assume that all five disturbances are independent, and model each with a zero-mean Gaussian distribution.

Algorithm 3.1 Intelligent Driver Model (IDM)

function IDMACCELERATION($\Delta r, v_{\text{ego}}, v_{\text{lead}}$)
  $\Delta v \leftarrow v_{\text{lead}} - v_{\text{ego}}$ ▷ Calculate relative velocity
  if $v_{\text{lead}} \neq \text{NaN}$ ▷ Check if lead vehicle is present
    $r_{\text{des}} \leftarrow r_{\text{min}} + v_{\text{ego}} \Delta T_{\text{des}} - \frac{v_{\text{ego}} \Delta v}{2\sqrt{a_{\text{max}} a_{\text{comf}}}}$ ▷ Calculate desired gap
    $a \leftarrow a_{\text{max}} \left(1 - \left(\frac{v_{\text{ego}}}{r_{\text{des}}}\right)^2 - \left(\frac{\Delta v}{\Delta r}\right)^2\right)$ ▷ IDM acceleration with lead vehicle
  else
    $a \leftarrow k \Delta v$ ▷ Free-road acceleration
  return CLAMP($a, -d_{\text{max}}, a_{\text{max}}$) ▷ Bound acceleration within limits

3.3 F-16 Ground Collision Avoidance System

The Ground Collision Avoidance System (GCAS) is an autopilot system designed to prevent aircraft from colliding with the ground. We consider validating a model of the GCAS system for the F-16 aircraft adapted from [48]. This represents a challenging validation problem due to the rarity of failures like ground collision and the long time horizon required to observe these failures. This problem is illustrated in Figure 3.3.

The F-16 aircraft model uses nonlinear 6 degree-of-freedom equations of motion comprising 13 state variables summarized in Table 3.2. The model represents both aerodynamic forces/moments using look-up tables based on wind tunnel data, and engine dynamics including afterburner effects. For a complete description of the F-16 dynamics and control system, see Stevens et al. [49] and Seto et al. [50].

The GCAS implements a recovery maneuver using a finite state machine with three primary phases: STANDBY, ROLL, and PULL. The default STANDBY phase applies no control effort as long as the aircraft's nose is sufficiently high and the aircraft is above the minimum safe altitude h_{min}, or flight deck. If the nose is not high enough or the aircraft is below the flight deck, the system enters a ROLL phase, which commands a roll maneuver to achieve wings-level flight. This phase uses a PD controller to compute a desired roll rate to level the wings. Once the wings are level, the system transitions to the PULL phase, which executes a 5-G pull-up maneuver

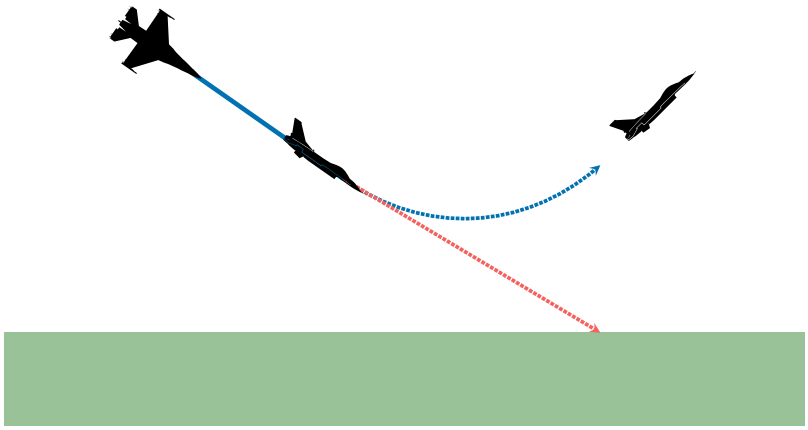


Figure 3.3: F-16 aircraft. The Ground Collision Avoidance System (GCAS) prevents the aircraft from colliding with the ground. The controller first rolls the wings level, then pulls up to prevent collision.

that is maintained until the aircraft's nose rises above the horizon. The controller must respect strict safety constraints: the total g-force must remain between -2 and 9 g's to prevent both structural damage and pilot physiological limits like g-force induced loss of consciousness. This algorithm is shown in algorithm 3.2. For more details on this model of the GCAS, see Heidlauf et al. [48].

Table 3.2: State variables in the F-16 aircraft model

Symbol	Meaning
Vt	Air Speed
α	Angle of Attack
β	Angle of Sideslip
ϕ	Roll
θ	Pitch
ψ	Yaw
P	Roll Rate
Q	Pitch Rate
R	Yaw Rate
Pn	Northward Displacement
Pe	Eastward Displacement
alt	Altitude
pow	Engine Power

Algorithm 3.2 GCAS Algorithm

Require: Current GCAS mode, mode $\in \{$STANDBY, ROLL, PULL$\}$
Require: Aircraft state vector x

```
function GcasControl(mode, x)
    if mode = Standby
        if Not IsNoseHighEnough(x) and Not IsAboveFlightDeck(x)
            mode ← Roll
    else if mode = Roll
        if IsRollRateLow(x) and AreWingsLevel(x)
            mode ← Pull
    else
        if IsNoseHighEnough(x)
            mode ← Standby
    if mode = Standby
        a ← StandbyCmd
    else if mode = Roll
        a ← RollWingsLevel(x)
    else
        a ← PullNoseLevel
    return mode, a
```

The high-level commands of desired roll-rate and pull-up acceleration are passed to an inner-loop controller that computes the control surface deflections necessary to achieve these commands. The inner-loop uses two decoupled linear quadratic regulators to handle longitudinal and lateral dynamics separately. The longitudinal controller uses elevator deflection to track commanded upward acceleration, while the lateral controller uses aileron and rudder to regulate the desired roll rate and stabilize combined side acceleration and yaw rate. The controller includes integral action on these three tracked variables to eliminate steady-state error. The details of this lower-level controller are provided in Seto et al. [50]

Failures in this system occur when the aircraft collides with the ground. The safety specification requires that the aircraft's altitude remains above 0 meters for all time steps. The disturbances in this system are sensor noise on the aircraft's roll, pitch, and yaw angles, and their rates. We assume that the sensor noise is independent and identically distributed zero-mean Gaussian.

3.4 Discussion

In this chapter, we introduce three safety validation problems that are used to evaluate the proposed methods in this thesis. The first problem is the inverted pendulum, a simple underactuated system that requires balancing a pendulum in the upright position while being subjected to torque disturbances. The second problem is an autonomous vehicle that must avoid colliding with a jaywalking pedestrian. The AV is subject to sensor noise and the pedestrian's motion. The third problem is the Ground Collision Avoidance System for the F-16 aircraft. The GCAS must prevent the aircraft from colliding with the ground while being subject to sensor noise.

The next chapter begins the discussion of the proposed methods for safety validation. We introduce a formulation of failure sampling as probabilistic inference and demonstrate it on the inverted pendulum and autonomous vehicle problems.

4 Failure Sampling as Probabilistic Inference

In this chapter, we develop an approach for failure sampling in autonomous systems that generalizes across different systems and sampling methods. First, we frame the problem as probabilistic inference of the distribution over failures due to disturbances in the environment. We then propose a modeling framework to represent the failure distribution in a probabilistic programming paradigm. This model-based representation enables us to easily use advanced inference algorithms for high-dimensional sampling problems, like Hamiltonian Monte Carlo [51], [52].

4.1 Motivation

Sequential decision-making systems in safety-critical domains require thorough validation for acceptance and safe deployment. Information about potential failure modes is critical for engineers and policymakers to determine whether certain failure modes require additional engineering attention and to estimate the risk associated with deployment [13]. Therefore, a key step of safety validation is to determine the distribution over failure trajectories, or search for possible failure modes and determine their associated likelihood. Searching for failures in safety-critical systems is challenging for several reasons. First, failures tend to be rare. Safety-critical autonomy is designed to be relatively safe from the outset, so naive methods like direct Monte Carlo sampling require an enormous number of samples to discover rare failures. Second, the search space is high dimensional. Autonomous systems typically have large state spaces and operate over long time horizon trajectories. Third, sequential systems can exhibit multimodal failures. Accurately estimating

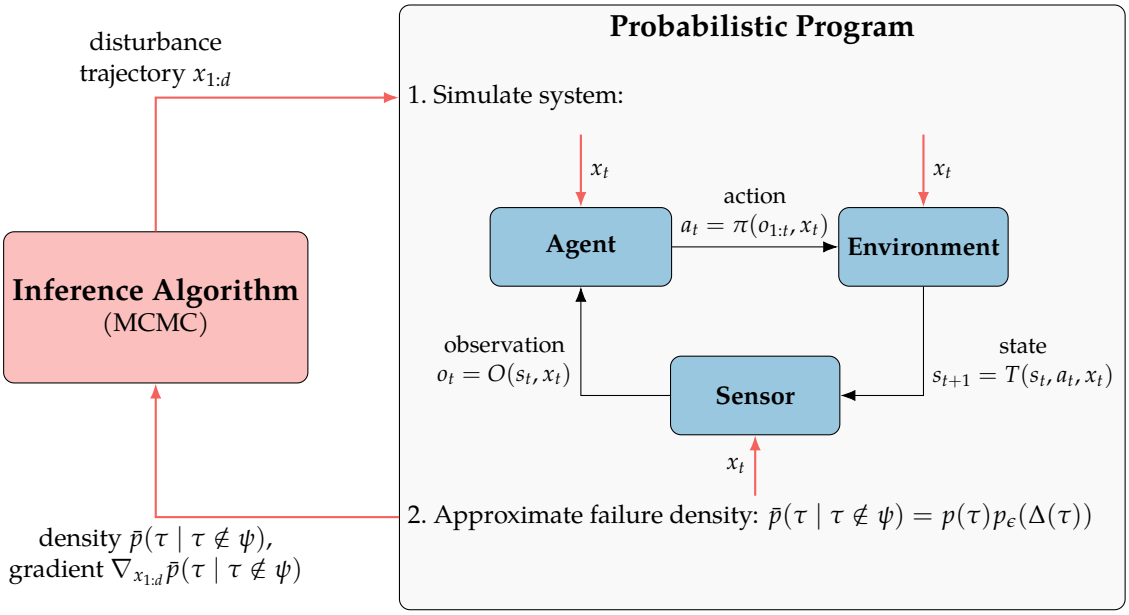


Figure 4.1: Overview of our approach for failure sampling as probabilistic inference. We wrap the simulation of the system under test with disturbances in a probabilistic program. The program is conditioned on the desired outcome of the simulation, that the trajectory is a failure. To facilitate inference on the discontinuous likelihood, we approximate the true failure distribution using a smoothing function that penalizes safe trajectories. If the system under test is differentiable, we can automatically compute gradients of the approximate failure distribution density. We can then use Hamiltonian Monte Carlo or any other generic inference algorithm to sample from the approximate failure distribution.

multimodal distributions requires methods that can explore the parameter space without converging to individual modes.

Several previous approaches to safety validation perform falsification [14], [53], [54], which aims to find a single failure example or failure mode. The main drawback of these methods is that they tend to converge to a single failure trajectory, such as the most extreme or most likely example [26]. The mode-seeking behavior of falsification methods is insufficient to capture the distribution over failure trajectories. Importance sampling [23], [30], [55] and Markov Chain Monte Carlo [25], [33] approaches have also been used to sample failure trajectories. These approaches aim to approximate the distribution over failures using samples. However, existing approaches are only applicable in low-dimensional spaces and rely on ad-hoc algorithmic modifications. These limitations make existing approaches less useful for the validation of sequential systems in a general setting.

In this contribution, we propose a general approach for sampling the distribution over failure trajectories in sequential systems. First, we frame the problem as probabilistic inference of the distribution over failures due to disturbances in the environment. We then propose a modeling framework to represent the failure distribution in a probabilistic programming paradigm. This model-based representation enables us to easily use advanced inference algorithms for high-dimensional sampling problems, such as Hamiltonian Monte Carlo [51], [52]. The proposed approach is illustrated in Figure 4.1. Our contributions are as follows:

- We frame sampling the distribution over failures in sequential systems as Bayesian inference.
- We propose a model-based approach to sample from the distribution over failures by incorporating system gradient information.
- We demonstrate the proposed approach on three sample validation problems.

4.2 Probabilistic Programming

This section provides a brief background on probabilistic programming to understand our approach for failure sampling. Probabilistic programming provides a framework for expressing and performing inference over probabilistic models. By extending traditional programming languages with random variables and conditioning, probabilistic programming languages allow developers to define complex generative models as executable programs. This approach allows for the separation of model specification from inference algorithms, making it easier to experiment with different models without reimplementing inference procedures.

Modern probabilistic programming languages like Stan [56], Pyro [57] and Turing.jl [58] provide a variety of sampling-based and variational inference techniques to efficiently estimate posterior distributions. These languages typically support automatic differentiation, provides gradients of the target density with respect to the input parameters. Gradient information can be critical for inference algorithms to scale to high-dimensional parameter spaces. The combination of flexible model specification and powerful inference algorithms makes probabilistic programming particularly well-suited for safety validation tasks in autonomous systems, where sampling from complex failure distributions often involves high-dimensional spaces with multiple failure modes.

4.3 Methodology

In this section, we present our framework for model-based probabilistic inference of the failure distribution in sequential systems. First, we introduce the necessary notation and definitions. Then, we describe our approach to estimate the distribution over failures through probabilistic inference.

We denote the simulation of the system under test with environment disturbances by a dynamics function f. Note that f represents the dynamics of both the system under test and the environment. The resulting state trajectory under disturbances is written as $\tau = f(x_{1:d})$. Since all stochasticity in the simulation is determined by $x_{1:d}$,

the disturbance model $p(x_{1:d})$ induces a distribution over state trajectories, $p(\tau)$. We seek the distribution over state trajectories that violate the safety property, the conditional distribution $p(\tau \mid \tau \notin \psi)$.

Environment dynamics or characteristics of the system under test often make it impossible to express the failure distribution analytically. However, it is usually easy to sample from these systems by simulating the system under test with disturbances. Bayesian inference methods such as MCMC approximate a target distribution by drawing samples. In this work, we frame the safety validation problem as Bayesian inference of the distribution over failures.

Sampling-based Bayesian inference requires a way to draw samples from a target distribution. We represent the state trajectory distribution as a probabilistic model using probabilistic programming [59]. Under this paradigm, we can describe the distribution over state trajectories as a function of random variables that represent the disturbance model. Modern probabilistic programming languages, such as Turing [58], let us wrap existing simulations in a probabilistic program with little modification of the underlying simulation. In general, these probabilistic programs take as input a set of parameters and return the log-likelihood of the parameters under the target distribution. This generic interface allows us to use a variety of inference algorithms to sample from the distribution over failures.

The probabilistic model represents the distribution over state trajectories induced by the disturbance model. We represent the distribution over failures by conditioning the probabilistic model on the desired outcome of the simulation — that the simulated trajectory is a failure. Any safe trajectory has zero likelihood under the conditional distribution. We represent this condition using an indicator function, whose value is one for failure trajectories and zero otherwise. Under this formulation, the log likelihood of a system trajectory $p(\tau \mid \tau \notin \psi)$ is:

$$\log p(\tau \mid \tau \notin \psi) = \log \mathbb{1}\{\tau \notin \psi\} + \log p(\tau) \tag{4.1}$$

The main challenges associated with sampling from $p(\tau \mid \tau \notin \psi)$ are the distribution's high dimensionality due to long time horizons, discontinuity due to the

indicator on failure trajectories, and potential multimodality. The following sections show how we address each of these challenges.

4.3.1 Model-based Approach with Automatic Differentiation

Trajectories in sequential systems of interest tend to be high dimensional. It is well established that black-box, gradient-free sampling from high-dimensional distributions becomes difficult with increasing dimensionality [51]. Gradient-based sampling algorithms such as Hamiltonian Monte Carlo (HMC) [60], [61] use information about the shape of the underlying distribution to take large steps in parameter space, allowing them to scale to higher dimensions.

To address the challenge of sampling high-dimensional failure trajectories in sequential systems, we take a model-based approach. Gradients of the dynamics function f with respect to disturbances are computed using automatic differentiation. The simulation environment and system under test are modeled in a framework compatible with automatic differentiation (AD). AD enables algorithms to compute exact gradients of the posterior likelihood density with respect to the input disturbances. We use Zygote [62], which performs source-to-source AD in Julia.

In contrast with some previous approaches that assume the system is Markov [42], our approach can handle non-Markov systems. We estimate the distribution over failures using rollouts of the system dynamics under disturbances, which fully characterize the system behavior and allows us to validate policies that depend on state estimates rather than true states.

4.3.2 Smoothing Approximation for Sharp Gradients

Gradients of the posterior density will be undefined in regions where the state trajectory is safe due to the indicator in Equation (4.1). Additionally, the boundary between safe and failing state trajectories is discontinuous. As a result, gradient-based sampling algorithms have no information about how to improve a proposal. We relax the discontinuity by approximating the indicator function using an exterior penalty function $p_\epsilon(\cdot)$ parameterized by ϵ. We also assume that we can calculate a

distance to failure to the failure region $\Delta(\tau)$ for any safe trajectory. The likelihood of the distance to failure under the approximation of the discontinuity is added to the likelihood of the sampled trajectory. The log-likelihood of a trajectory under this smoothing approximation is:

$$\log \bar{p}(\tau \mid \tau \notin \psi) = \log p_\epsilon(\Delta(\tau)) + \log p(\tau) \tag{4.2}$$

There are many possible choices for the exterior penalty function $p_\epsilon(\cdot)$. Any function that approximates the discontinuity between safe and failure regions can be used. Some possible choices include an exponential likelihood, a sigmoid, or a squared exponential (or Gaussian) likelihood function.

The role of AD is to calculate the gradient of the penalty density with respect to the input disturbance trajectory $\nabla_{x_{1:d}} \log p(\Delta(f(x_{1:d})))$, which is now defined for safe and failure trajectories. The exterior penalty smooths the discontinuity between the safe and failure regions of the state trajectory space while changing the density in the failure region by a constant. This smoothing approximation along with AD enables us to use gradient-based sampling methods.

We use a Gaussian distribution with mean zero and variance ϵ as the penalty function. The variance ϵ acts as a hyperparameter, with smaller values better approximating the discontinuity and larger values yielding smoother posterior densities. Consider an illustrative validation problem where the state distribution is the unit Gaussian and failure occurs when $|s| > 5$. Figure 4.2 illustrates the unnormalized density of the approximate failure distribution for a few values of ϵ. As ϵ decreases towards zero, the resulting distribution is a better approximation of the true failure distribution, shown in dark red. Larger values of ϵ smooth the gap between the separated failure modes on the far left and right tails of the distribution. Empirically, we find values in the range $\epsilon \in [0.01, 0.1]$ work well for the problems investigated in this work.

algorithm 4.1 shows the probabilistic program for sampling from the approximate failure distribution. The program samples a disturbance trajectory $x_{1:d}$ and simulates the system under test with the disturbance. The initial state may either

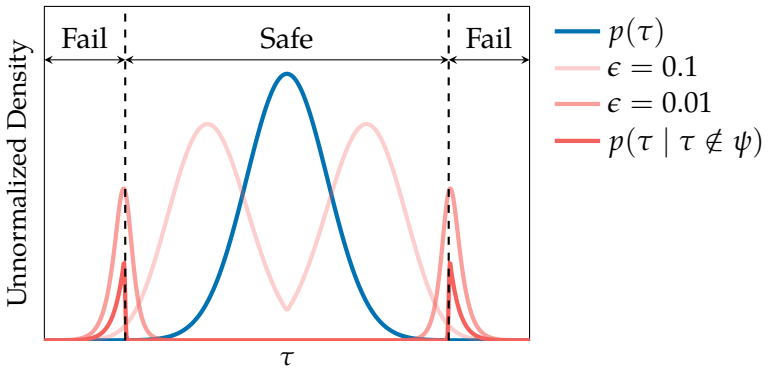


Figure 4.2: Illustration of the approximate failure distribution for different values of the smoothing hyperparameter ϵ. As ϵ decreases, the approximate distribution interpolates between the nominal distribution $p(\tau)$ and the true failure distribution.

be specified or sampled from an initial distribution. Each time a disturbance is sampled, the PPL internally accumulates the log-likelihood of the disturbance trajectory. After the trajectory is rolled out, we call AddLogProb($\log p_\epsilon(\Delta(\tau))$) to add the log-likelihood of the distance to failure under the smoothing approximation. The program returns the log-likelihood of the disturbance trajectory under the approximate failure distribution to the inference algorithm.

Algorithm 4.1 Failure Sampling Probabilistic Program

function FailureDistribution(π, T, O, D)
  $s_0 \sim p(s_0)$
  $\tau \leftarrow [s_0]$
  for $t \in 1 : t_{\max}$
    $x_t \sim D(x_t)$
    $o_t \leftarrow O(s_t, x_t)$
    $a_t \leftarrow \pi(o_{1:t}, x_t)$
    $s_{t+1} \leftarrow T(s_t, a_t, x_t)$
    $\tau \leftarrow$ append$(\tau, (o_t, a_t, x_t, s_{t+1}))$
  AddLogProb($\log p_\epsilon(\Delta(\tau))$)

4.3.3 *Sampling for Multimodality*

The generality of the approach makes available a wide variety of off-the-shelf Bayesian inference methods. We use HMC with No-U-Turn Sampling (NUTS) [52], which is a general HMC variant that excels at sampling from complex, high-dimensional posterior distributions.

Multimodality in the distribution over failure trajectories poses a challenge for many MCMC algorithms [63]. Sampling algorithms may converge to a single mode rather than mixing across the modes in parameter space. In our illustrative example, a single MCMC chain may only sample from the left or right mode of the failure distribution. One way to explore a multimodal space is to run many chains of MCMC from different starting points. In our sampling approach, we use multiple NUTS chains with different starting disturbance trajectories. The number of chains selected depends on the difficulty of the problem, with higher dimension and more modes generally requiring more chains.

4.4 *Experiments*

This section presents the experimental evaluation of the proposed failure sampling approach. First, we discuss our evaluation metrics and baseline sampling approaches. Then, we describe the three simulation environments and policies used in validation experiments.

Metrics. We evaluate the proposed approach based on failure rate, log-likelihood of failures discovered, and computation time. The failure rate is the proportion of drawn samples that result in failure. We also use a coverage metric based on the mean dispersion of failure trajectories in disturbance space [64]. The mean dispersion $C_{disp} \in [0, 1]$ is computed over a grid in trajectory space as

$$C_{disp} = 1 - \frac{1}{n} \sum_{j=1}^{n} \frac{\min(d_j, g)}{g} \tag{4.3}$$

where n is the number of grid points, g is the grid spacing, and d_j is the minimum distance from the grid point to a point in the sampled disturbance trajectories. Larger values of C_{disp} indicate that a sampler has discovered failures in more regions of disturbance trajectory-space.

Baselines. We compare the proposed gradient-based approach to direct Monte Carlo (MC) sampling and the black-box MCMC Particle Gibbs (PG) [65] algorithm with 1,000 particles. PG combines particle filtering with Gibbs sampling to sample from complex posterior distributions. It extends traditional Gibbs sampling by using a particle filter to handle high-dimensional trajectories.

4.4.1 Inverted Pendulum

The first system we validate is an inverted pendulum with a neural network controller. While this is not a safety-critical system, it demonstrates the ability of the proposed approach to validate policies with machine learning components. In the inverted pendulum, a policy exerts a control torque to keep the pendulum balanced upright. We optimize a neural network policy using the Proximal Policy Optimization reinforcement learning algorithm, which has been shown to perform well on the inverted pendulum [66]. The policy is represented by a two-layer neural network with 32 neurons per layer and tanh activations between hidden layers. The policy is trained for 10^6 time steps.

We sample failures of control policy by considering external torque disturbances from the environment. Torque disturbances are assumed to be distributed according to a zero-mean Gaussian with variance σ_n. This experiment uses $\sigma_n = 0.1\,\mathrm{N\,m}$ and $T_{\max} = 2\,\mathrm{N\,m}$ so that disturbances are likely to be much smaller than the maximum control torque. A failure occurs when the torque due to gravity overcomes the maximum control torque, which occurs when $|\theta| \geq 0.5\,\mathrm{rad}$. The distance to failure is the minimum absolute value of the angle of the pendulum over the trajectory. Pendulum simulations span a 5 s horizon with a 0.1 s time step. We sample 10,000 samples for each method, with 10 MCMC chains and 1,000 samples per chain.

4.4.2 *Autonomous Vehicle Scenario*

The second experiment involves an autonomous vehicle (AV) approaching a single pedestrian at a crosswalk. The disturbances control pedestrian acceleration and noise in the AV's sensors. The disturbances to the system are represented by a 6-tuple $x = (\delta r_x, \delta r_y, \delta v_x, \delta v_y, a_x, a_y)$ where $(\delta r_x, \delta r_y)$ is the noise in the pedestrian's observed position, $(\delta v_x, \delta v_y)$ is the noise in the pedestrian's observed velocity, and (a_x, a_y) is the pedestrian's acceleration. The noise components are added to the vehicle's observation of each pedestrian's position and velocity at each time step.

The disturbances are assumed to be distributed according to a multivariate Gaussian distribution with zero mean and diagonal covariance matrix. The diagonal components of covariance are $\sigma_r^2 = 0.1$ for position measurement noise, $\sigma_v^2 = 0.1$ for velocity measurement noise, $\sigma_{lat}^2 = 0.01$ over the pedestrian's lateral acceleration, and $\sigma_{lon}^2 = 0.1$ over the pedestrian's longitudinal acceleration. These values are chosen to reflect relatively small sensor noise, and that the pedestrian is less likely to accelerate longitudinally while crossing the street.

Failures of this scenario are defined by any collision between the pedestrian and vehicle. The distance to failure is the minimum Euclidean distance between the vehicle and the pedestrian. We sample 20,000 samples for each method, with 20 MCMC chains and 1,000 samples per chain.

4.4.3 *Partially Observable Lunar lander*

The final validation problem is a simulation of a lunar lander with partially observed state information. The objective of the lunar lander's policy is to guide the vehicle in a target area with a soft landing. The vehicle state is represented by the 6-tuple $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$, where x is the horizontal position, y is the vertical position, and θ is the vehicle's orientation. The vehicle's dynamics are deterministic. The vehicle makes noisy observations of its angular rate ω, horizontal speed v, and above ground level altitude (AGL). AGL is a noisy measurement of the distance from the vehicle's center of mass to the ground along the vehicle's longitudinal axis. The continuous actions are defined by $(T, F_x, \dot{d})$ where T is the main thrust acting along

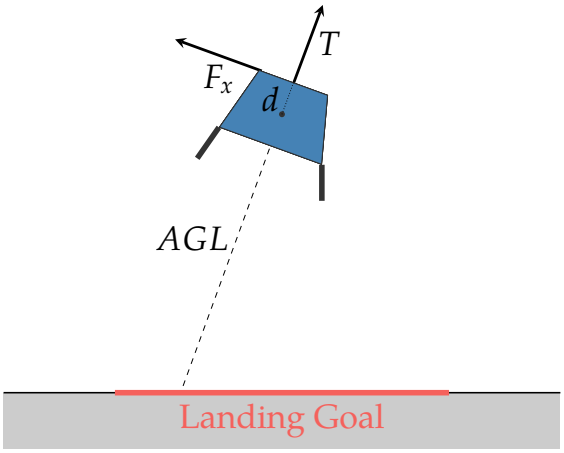


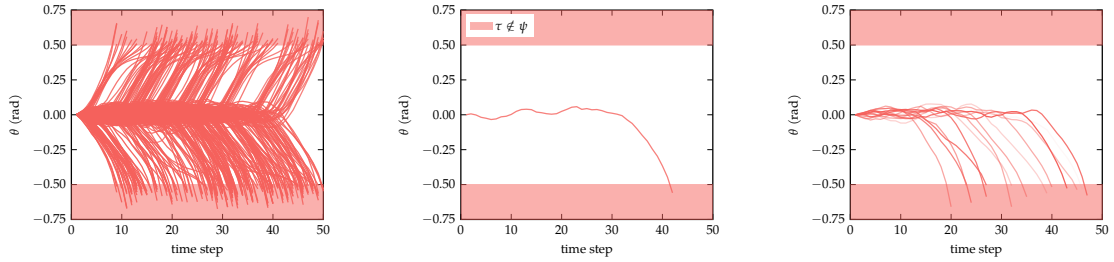
Figure 4.3: Illustration of the lunar lander system.

the longitudinal axis, F_x is lateral corrective thrust acting at an offset of d from the center of mass. The lander is illustrated in Figure 4.3.

We optimize a control policy by considering a discrete, fully observable version of the continuous, partially observable problem. We use approximate value iteration to solve for a table-based policy in the discrete problem. Next, we create a differentiable representation of the table-based policy by performing behavior cloning with a neural network approximator. The network is trained to minimize the mean squared error between the continuous output and the table-based action.

We assume that observation noise is identically and independently distributed at each time step according to a zero-mean Gaussian with a diagonal covariance matrix. We use $\sigma_\omega^2 = 0.02$, $\sigma_v^2 = 0.1$, and $\sigma_{AGL}^2 = 1.0$. An extended Kalman filter maintains a multivariate Gaussian belief over the true state in the partially observable problem. The policy uses the mean to calculate an action.

Our approach samples sequences of observation noise that lead to a hard landing, defined as the landing velocity being greater than $6\,\mathrm{m\,s^{-1}}$. Empirically, we discovered the lunar lander contained multiple isolated failure modes. We select initial conditions for the MCMC samplers by performing black-box optimization of the probabilistic model's posterior density. We sample 30 MCMC chains with 1,000 samples each to explore the different failure modes.



(a) HMC (b) PG (c) MC

Figure 4.4: Failure trajectories for the inverted pendulum with HMC and baseline black-box methods. HMC discovers many likely failures in the positive and negative directions. Greater opacity reflects a higher likelihood failure.

4.5 *Results*

This section presents experimental results for each validation problem. Probabilistic models of each system were written using Turing and sampled using the HMC-NUTS implementation provided by [58]. Gradients were computed using the Zygote [62] AD framework. Experiments were ran in serial on a desktop with 32GB of RAM and an Intel i7-7700K CPU.

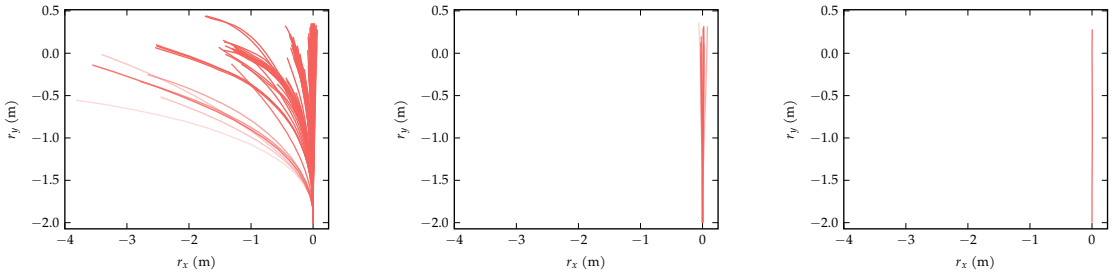
Section 4.5 shows the failure rate, failure trajectory log-likelihood, sampling time per failure, and mean dispersion for each sampling method and validation setting. The proposed approach consistently outperforms baselines in terms of failure rate. The gradient-based HMC is able to consistently sample from the high-dimensional failure region, while PG and MC samplers struggle. Additionally, the results suggest that HMC consistently samples from the high-likelihood failure region. The computational cost of computing gradients through the state trajectories is outweighed by the increase in sampling performance. HMC samples failures 5–10 times faster in wall clock time compared to baselines. Failures sampled using HMC also cover a significantly greater portion of the disturbance trajectory space across all experiments. Greater disturbance space coverage generally corresponds to a more diverse set of failures being sampled.

Table 4.1: Results for the inverted pendulum (IP), autonomous vehicle scenario (AV), and partially observable lunar lander (POL). Failure rate, trajectory mean and maximum log-likelihood (LL), sampling time per failure, mean dispersion C_{disp}, and total sampling time in minutes are reported for each algorithm. HMC outperforms black-box baselines in terms of failure rate, sampling time per failure, and disturbance-space coverage.

	Method	Failure Rate	Mean LL	Max. LL	Failures	C_{disp}	Time
IP	**HMC**	9.8×10^{-1}	-24.66 ± 9.59	-8.18	8.0	7.5×10^{-1}	22
	PG	2.0×10^{-4}	-21.42 ± 0.15	-19.89	1.1×10^{-3}	2.1×10^{-4}	27
	MC	3.0×10^{-3}	-18.44 ± 2.97	-13.81	1.2	2.4×10^{-2}	2
AV	**HMC**	7.8×10^{-1}	-2.01 ± 20.00	0.53	7.8	2.0×10^{-2}	40
	PG	1.7×10^{-1}	-0.71 ± 0.48	0.11	6.5×10^{-3}	1.0×10^{-5}	40
	MC	5.0×10^{-4}	-12.05 ± 0.06	-12.35	8.5×10^{-1}	1.0×10^{-9}	3
POL	**HMC**	1.6×10^{-1}	1.68 ± 1.69	3.19	4.1×10^{-1}	2.1×10^{-1}	375
	PG	2.4×10^{-4}	1.80 ± 0.21	2.12	3.3×10^{-5}	3.2×10^{-3}	810
	MC	3.0×10^{-5}	2.34 ± 1.09	2.61	1.7×10^{-1}	1.5×10^{-5}	13

Inverted pendulum. Figure 4.4 shows failures sampled using the HMC and baseline methods. The number of failure trajectories for HMC is downsampled for clarity. HMC finds a more diverse set of failure modes compared to PG and MC, with many failures in the positive- and negative- θ direction. The failures found by HMC span a wide range of trajectory lengths and failure modes. HMC uses the gradient of the trajectory with respect to both the pendulum dynamics and the neural network controller, enabling much broader coverage of the failure distribution than the black-box baselines.

HMC samples cover the disturbance space well and uncover asymmetric failure modes. The most likely failure trajectories found with each method lead to the pendulum falling in the negative-θ direction. HMC failure samples indicate that the neural network controller exhibits different failure modes in the positive- and negative-θ directions. Positive-θ failures have a lower mean log-likelihood of -28.1, while negative-θ failures have a mean of -17.4. The pendulum tends to fall more



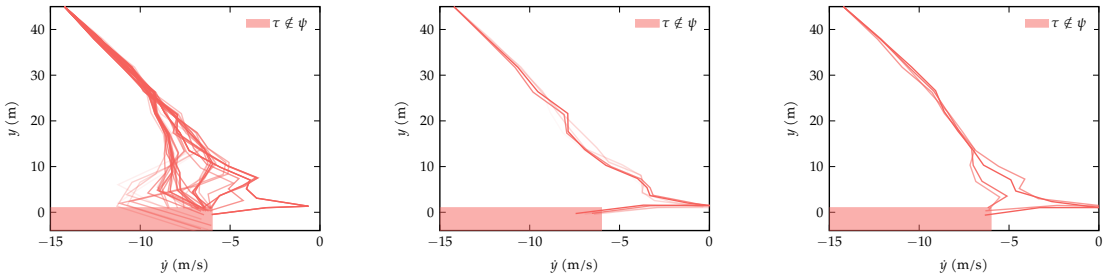
(a) HMC (b) PG (c) MC

Figure 4.5: Pedestrian trajectories resulting in collision for the autonomous vehicle scenario with HMC and baseline methods. HMC samples failures with a mixture of high sensor noise and lateral motion. Baseline methods only sample from the sensor noise failure mode. Greater opacity reflects a higher likelihood failure.

directly in the negative direction, while some failures in the positive direction hesitate before falling. This may indicate that this particular neural network controller learned using PPO is more robust to disturbances in the positive-θ direction than in the negative-θ direction.

AV scenario. Sampled pedestrian trajectories that result in a collision are shown in Figure 4.5. The HMC samples are subsampled to 500 trajectories for clarity. All trajectories end at the time of collision. HMC finds a wide variety of failure modes in terms of the pedestrian's trajectory while the baselines focus on a single mode.

The more likely failure modes involve little change in the acceleration of the pedestrian. These failure trajectories are concentrated towards the y-axis, and introduce sensor noise to cause the AV to incorrectly estimate the relative distance or velocity to the pedestrian. Other failure modes involve accelerations of the pedestrian in the negative-x direction. Baseline methods were unable to sample from these modes. In extreme cases, the AV comes to a complete stop in front of the crosswalk and the pedestrian walks into the vehicle. This failure mode has been observed in previous work [67], [68]. While this pedestrian-induced failure might not be crucial to the safety of the AV policy, it highlights important questions for system designers such as accident blame and scenario construction.



(a) HMC (b) PG (c) MC

Figure 4.6: Failure trajectories for the lunar lander in altitude y and vertical velocity $\dot{y}$ with HMC and baseline methods. Greater opacity reflects a higher likelihood.

Lunar lander. Failure trajectories for the lunar lander are shown in Figure 4.6. The failure region is indicated by the red shaded rectangle. In the most likely failure mode, there is relatively small noise during the initial segment of the descent. A small amount of noise is added to the AGL measurement in the negative direction, making the lander believe that it is slightly closer to the ground than it really is. Next, a larger amount of noise is added to AGL in the positive direction, preventing the lander from increasing thrust in a critical moment. This highlights the ability of the proposed approach to find failures in non-Markov systems. The model-based approach is able to find diverse-likely failures by computing gradients through sequences of observations to find belief states that lead to failure.

4.6 Discussion

In this chapter, we proposed a general framework for failure sampling in autonomous systems. The framework poses the failure sampling problem as probabilistic inference of the distribution over failures trajectories. We show how to represent the failure distribution for any system under test in a probabilistic programming language. Additionally, the probabilistic program allows us to easily take advantage of system-specific structure, such as the use of automatic differentiation to compute gradients of the posterior density. A wide variety of inference algorithms may be

used together with the probabilistic program to sample from the failure distribution. We demonstrate the proposed approach on three validation problems using both black-box and gradient-based sampling methods.

One of the key ideas of this work is to decouple the problem of failure sampling from the solution. By framing the problem as probabilistic inference, we open the door of general-purpose Bayesian inference algorithms to safety validation problems. Under this framework, engineers "plug and play" sampling algorithms into safety validation and choose methods according to the problem at hand. Additionally, as methods for Bayesian inference continue to improve, the proposed approach will benefit from these advances. There are already many promising methods for Bayesian inference that could be applied to safety validation problems, such as variational inference [69] and Stein-variational gradient descent [70]. We leave the exploration of these methods to future work.

In the next chapter, we present a method for black-box failure sampling in high-dimensional problems. As demonstrated in this chapter, black-box (gradient-free) methods often struggle to scale to very high-dimensional problems. In industry applications, it may not always be possible to compute gradient information. Next, we present an algorithm for black-box failure sampling that uses a deep generative model to sample from the distribution over failures.

5 Failure Sampling using Generative Models

In this chapter, we develop a method for approximately sampling from the distribution over failure trajectories in black-box systems, where the validation algorithm is limited to observing input-output behavior. To address the challenges of scaling failure sampling to complex, high-dimensional black-box problems, we propose an algorithm that adaptively trains a diffusion model to match the distribution over disturbance trajectories. First, we provide some background on diffusion models, and describe our adaptive training algorithm. Next, we demonstrate the approach that we call Diffusion-based Failure Sampling (DiFS) on a set of validation problems, comparing it to two baselines. We show that DiFS outperforms baselines in terms of sample efficiency, failure diversity, and mode coverage. Our results suggest that diffusion models can be an effective tool for sampling failure trajectories in high-dimensional, black-box systems.

5.1 Motivation

In real-world development of autonomous systems, the system or simulator often contain many diverse components, and may rely on complex machine learning models. Traditional validation methods often assume access to a complete mathematical model of the system under test [15]. These methods scale poorly to large, complex systems. It may be extremely difficult to obtain any internal knowledge about a simulator or system under test like gradients that would facilitate validation. Instead, we only have access to input-output behavior of the system. This is known as a black-box setting. In this work, we focus on the problem of validating safety-critical

autonomous systems in a black-box setting. Black-box approaches tend to scale well, since they only require the ability to simulate the system under test.

Sampling failures in black-box safety-critical autonomous systems is challenging for several reasons. First, the search space is high-dimensional due to the large state spaces and long trajectories over which autonomous systems operate. Second, failures tend to be rare because systems are typically designed to be relatively safe. Direct Monte Carlo sampling may require an enormous number of samples to discover rare failures. Third, autonomous systems can exhibit multiple potential failure modes that may be difficult to uncover.

Previous work in black-box validation generally frames the problem as optimization, where the objective is to find an input disturbance to the system that leads to system failure. Black-box approaches rely on classical gradient-free optimization [14], [53] and path planning [54]. An approach called adaptive stress testing (AST) [71] outputs disturbances that are likely to lead to a system failure. The main drawback of these algorithms is that they tend to converge to a single failure mode. See the survey by Corso et al. [13] for more on black-box safety validation algorithms.

There are parametric and non-parametric approaches to sampling the failure distribution. Parametric methods such as the cross-entropy method (CEM) [23], [30] iteratively update a parametric family towards the failure distribution. However, the choice of parametric family may limit performance in complex, high-dimensional distributions. Non-parametric methods like adaptive multilevel splitting (AMS) [32], [33] use Markov chain Monte Carlo (MCMC) to iteratively bring a set of particles towards the failure distribution. However, black-box MCMC-based methods struggle to scale to high-dimensional validation problems as demonstrated in Chapter 4. Our aim is to address these limitations by training a diffusion model to sample the failure distribution.

Recent validation methods have found success by incorporating generative models in various safety validation algorithms. For example, normalizing flows have been used to perform more efficient gradient-based sampling of the failure distribution [25]. A different approach fits a normalizing flow to a smooth approximation of the distribution over failures and draws samples from the trained flow [72]. The

drawback of these methods is that both require a differentiable simulation, which may not be available. Generative adversarial networks have also been used to falsify signal-temporal logic expressions [73]. This technique targets single failure modes, and does not account for the distribution over failures.

Diffusion models are a class of generative model that generate samples by learning to reverse an iterative data noising process [74], [75]. These models have demonstrated success in modeling high-dimensional, multimodal distributions in applications such as image generation and robot motion planning [76], [77], [78], [79]. Based on these results, we explore using diffusion models to learn a distribution over disturbances that lead to failure in safety-critical systems.

This work aims to enable efficient black-box sampling of the distribution over failure trajectories in autonomous systems. Our method generates system disturbances using a denoising diffusion model conditioned on a desired system robustness. We propose an adaptive training algorithm that updates the proposal distribution based on a lower-quantile of the sampled data, meaning that the proposal progressively moves samples towards the failure region. We evaluate the proposed approach on five validation problems ranging from a simple two-dimensional example to a complex 1200-dimensional aircraft ground collision avoidance autopilot. We compare our method with two baselines using metrics that assess the fidelity and diversity of failure samples. Our results show that our method outperforms baseline methods in modeling high-dimensional, multimodal failure distributions. Figure 5.1 illustrates the performance of the method compared to baselines on a toy problem. Our code is available on GitHub.[1] Our contributions are as follows:

- We model the distribution over failure trajectories in autonomous systems using a robustness-conditioned diffusion model.
- We propose Diffusion-based Failure Sampling (DiFS), a training algorithm that gradually trains a diffusion to sample the failure distribution.
- We evaluate DiFS on four challenging validation problems, demonstrating superior failure fidelity and diversity with respect to Monte Carlo samples.

[1] https://github.com/sisl/DiFS

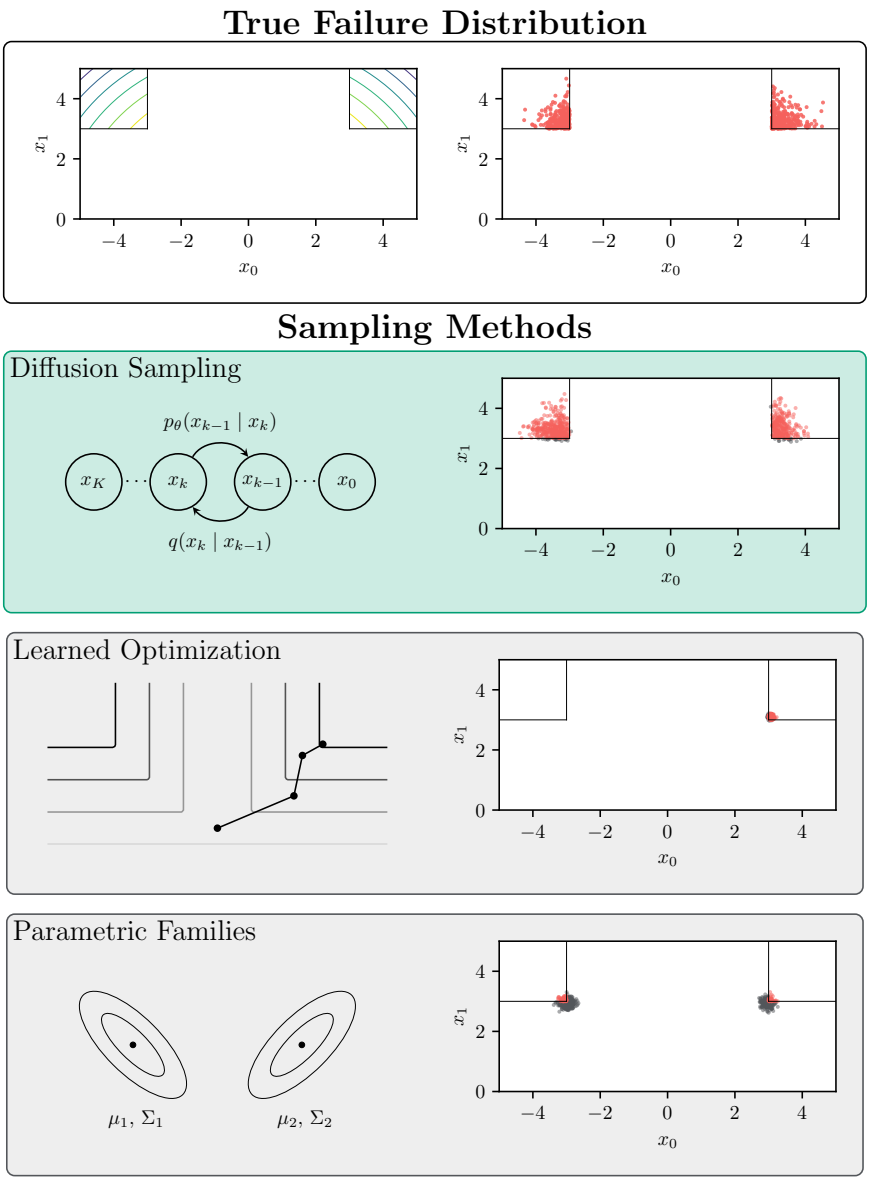


Figure 5.1: Illustration of the proposed method compared to baselines on a toy problem. The diffusion model learns to generate failure samples that are more similar to the true failure distribution compared to baselines.

5.2 *Diffusion Models*

In this section, we present some background information on diffusion models necessary for understanding our contribution. Denoising diffusion probabilistic models are a class of generative model that have shown remarkable success in modeling complex, high-dimensional distributions across various domains including image generation and robotic planning. At a high level, diffusion models learn to produce data samples that look very similar to those from a data distribution. The key idea behind diffusion models is to gradually add noise to data according to a fixed schedule and then learn to reverse this process to generate new samples. This approach provides a principled way to model complicated distributions while maintaining stable training dynamics.

The training of diffusion models involves learning a parameterized function that can reverse the forward diffusion process. In the forward process, a data sample is incrementally perturbed with Gaussian noise. Given a dataset of real samples $\mathbf{x}_0$, the forward process produces noisy samples according to

$$q(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}\Big(\mathbf{x}_k \mid \sqrt{1-\beta_k}\,\mathbf{x}_{k-1},\, \beta_k \mathbf{I}\Big) \tag{5.1}$$

where β_k controls the noise magnitude at each step. The variance schedule is designed such that $0 < \beta_1 < \beta_2 < \ldots < \beta_K$. As k increases, the sample becomes nearly indistinguishable from noise.

The reverse process aims to recover the original data distribution by iteratively denoising. This reverse process is modeled as:

$$p_\theta(\mathbf{x}_{k-1} \mid \mathbf{x}_k) = \mathcal{N}\Big(\mathbf{x}_{k-1} \mid \mu_\theta(\mathbf{x}_k, k),\, \beta_k \mathbf{I}\Big) \tag{5.2}$$

with θ representing the learned parameters. Starting from a sample drawn from a standard Gaussian distribution, the model gradually refines it to generate data that resembles the target distribution.

Training involves matching the learned reverse process to the true reverse dynamics, typically by minimizing the error between the predicted and the actual added noise. A common approach is to reparameterize the mean of the reverse process in terms of the noise removed at each timestep This reparameterization takes the form

$$\mu_\theta(\mathbf{x}_k, k) = \frac{1}{\alpha_k}\left(x_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\right) \tag{5.3}$$

where ϵ_θ is the network predicting the noise removed at each step and $\alpha_k = 1 - \beta_k$. The term $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$ is the cumulative noise scale. Under this reparameterization, the diffusion model can be trained using a regression-style loss that penalizes the difference between the predicted noise and the actual noise:

$$\mathcal{L}(\theta) = \mathbb{E}_{k,\mathbf{x}_0,\epsilon}\left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_k}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, k)||^2\right] \tag{5.4}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. After training, the model can be used to generate samples by starting from a standard Gaussian noise sample and running the reverse process:

$$\mathbf{x}_K \sim \mathcal{N}(0, \mathbf{I}), \quad \mathbf{x}_{k-1} = \mu_\theta(\mathbf{x}_k, k) + \sqrt{\beta_k}\epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \mathbf{I}) \tag{5.5}$$

For more details on training and sampling of diffusion models, we refer the reader to Ho et al. [74], Song et al. [75], and Song et al. [80].

5.3 *Methods*

In this section, we present our method to approximately sample from the distribution over failures in safety-critical systems, Diffusion-based Failure Sampling (DiFS). First, we introduce necessary notation and formulate the distribution over failures. Next, we describe how diffusion models may be used for validation given sufficient failure data. Finally, we describe DiFS, which collects failure samples during training through an adaptive training algorithm.

5.3.1 Problem Formulation

Consider a system under test that takes actions in an environment after receiving observations. We assume that we can evaluate the robustness of a system trajectory $\rho(\tau)$. We define a system failure to occur when the robustness is below a given threshold, $\rho(\tau) \leq r_{\text{fail}}$.

We perturb the environment with disturbance trajectories $\mathbf{x}$ to induce the system under test to violate the robustness constraint. For example, sensor noise and stochastic dynamics are potential disturbances that could lead to failure. We denote the simulation of the system under test with environment disturbances by a dynamics function f. Note that f represents the dynamics of both the system under test and the environment. We assume that disturbances determine all sources of stochasticity in the environment. Therefore, the resulting system trajectory under disturbances is written as $\tau = f(\mathbf{x})$.

Our goal is to sample disturbance trajectories that violate the robustness constraint, or sample from the conditional distribution

$$p^\star(\mathbf{x} \mid r_{\text{fail}}) \propto \mathbb{1}\{\rho(f(\mathbf{x})) \leq r_{\text{fail}}\} p(\mathbf{x}) \tag{5.6}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function.

Representing the distribution $p^\star(\mathbf{x} \mid r_{\text{fail}})$ analytically is typically intractable even for simple autonomous systems. Sampling from this distribution is often difficult due to the high dimensionality, rareness, and multimodality of failure trajectories in autonomous systems. We propose to sample from this distribution by training a denoising diffusion model to generate disturbance trajectories from the failure distribution. Diffusion models are particularly appealing for this application due to their success in modelling complex high-dimensional data in domains like robotic planning, their stability during training, and their potential to compute likelihoods. In the following subsection, we present a brief discussion of diffusion models in the context of our failure sampling problem.

5.3.2 *Diffusion for Validation*

We use a denoising diffusion probabilistic model to represent the failure distribution [74], [81]. Let $\mathbf{x}_k$ be the disturbance trajectory at the k-th diffusion step, with $k = 0$ corresponding to a disturbance trajectory from the target data distribution. The forward diffusion process, starting from $\mathbf{x}_0$, is

$$q\left(\mathbf{x}_{1:K} \mid \mathbf{x}_0\right) = \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{x}_k \mid \sqrt{1-\beta_k}\mathbf{x}_{k-1}, \beta_k \mathbf{I}\right) \tag{5.7}$$

where $\beta_1, \ldots, \beta_K$ is a predetermined variance schedule controlling the noise magnitude at each diffusion step. As the noise accumulates, the data approximately transforms into a unit Gaussian at the final diffusion step.

To generate disturbances with a desired level of robustness, we train a conditional diffusion model. This model is trained on a dataset $\mathcal{D} = \{(\mathbf{x}_m, r_m)\}_{m=1}^{M}$ containing disturbance trajectories paired with corresponding robustness level. Starting from sampled noise, the diffusion model gradually denoises the sample at each step. This reverse process is

$$p_\theta\left(\mathbf{x}_{0:K} \mid r\right) = p\left(\mathbf{x}_K\right) \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{x}_{k-1} \mid \mu_\theta\left(\mathbf{x}_k, k, r\right), \beta_k \mathbf{I}\right) \tag{5.8}$$

where $p\left(\mathbf{x}_K\right) = \mathcal{N}(\mathbf{x}_K \mid \mathbf{0}, \mathbf{I})$ and θ represents the parameters of the diffusion model. In the appendix, we perform an ablation that shows that conditioning the model significantly improves generalization compared to an unconditional model.

5.3.3 *Adaptive Training*

Inspired by adaptive methods such as CEM and AMS, we propose an iterative algorithm that adaptively selects the robustness threshold r_i during training. The algorithm has two key features. First, the algorithm collects information about the conditional distribution by sampling trajectories with robustness levels from the current r_i to the target. Next, we assess how good the model is by evaluating

the true robustness associated with each sample, and adaptively update the target robustness threshold. These features balance between collecting information about the target distribution and improving model performance.

Initially, the diffusion model is trained on samples from $p(\mathbf{x})$ so that the distribution is close to the disturbance model. At each iteration, the algorithm performs four main steps consisting of sampling disturbances, evaluating their robustness level, updating the robustness threshold, and training the diffusion model.

To sample disturbances from the model, we need to choose an input desired robustness level. One option would be to use the failure robustness level r_{fail}. However, this may lead to poor performance if there is limited data near r_{fail}. Instead, we sample conditioning robustness values uniformly between the current robustness threshold and the target, $c_n \sim \mathcal{U}[r_{\text{fail}}, r_i]$. By uniformly sampling the input conditions, we aim to balance generating samples close to failure when the model can accurately represent the target distribution with collecting new disturbance data in unexplored regions. Disturbances are sampled from the diffusion model according to $\mathbf{x}_n \sim p_{\theta_i}(\mathbf{x} \mid c_n)$, and we evaluate the robustness r_n of each sampled $\mathbf{x}_n$. We compute the updated robustness threshold r_{i+1} according to the bottom α quantile of sampled robustness values. This ensures that a minimum fraction α of the samples have at most a robustness level r_{i+1}.

Since our objective is to model a distribution conditioned on the desired robustness level, we maintain a running dataset of collected samples rather than only training on the newly sampled data as in CEM. After updating the robustness threshold, we append all samples to the dataset. For training the next iteration of the diffusion model, we only use samples with a robustness of $r \leq r_i$. We do this to focus the training effort towards the areas of decreasing robustness that we ultimately wish to sample from. Finally, we train the diffusion model on this updated dataset. We call this algorithm Diffusion-based Failure Sampling, (DiFS). DiFS as described here is outlined in algorithm 5.1.

Algorithm 5.1 Diffusion-based Failure Sampling (DiFS)

Require: simulator $f(\cdot)$, robustness $\rho(\cdot)$, failure robustness r_{fail}
Require: initial denoising diffusion model $p_{\theta_0}(\mathbf{x} \mid r)$
Require: prior disturbance model $p(\mathbf{x})$, quantile threshold α

function DiFS($f, \rho, r_{\text{fail}}, \alpha, p_\theta(\mathbf{x} \mid r), p(\mathbf{x})$)
  $i \leftarrow 0$
  sample $\{\mathbf{x}_n \sim p(\mathbf{x})\}_{n=1}^N$
  evaluate robustness $\{r \leftarrow \rho(f(\mathbf{x}_n))\}_{n=1}^N$
  compute $r_\alpha \leftarrow \text{quantile}(\alpha, \{r_n\}_{n=1}^N)$
  compute $r_i \leftarrow \text{maximum}(r_{\text{fail}}, r_\alpha)$
  $\mathcal{D} \leftarrow \{(\mathbf{x}_n, r_n)\}_{n=1}^N$
  $\theta_i \leftarrow \textsc{Train}(\theta_0, \mathcal{D})$
  while $r_i > r_{\text{fail}}$
    sample robustness conditions $\{c_n \sim \mathcal{U}[r_{\text{fail}}, r_i])\}_{j=1}^N$
    sample disturbances $\{\mathbf{x}_n \sim p_{\theta_i}(\mathbf{x} \mid c_n)\}_{n=1}^N$
    evaluate robustness $\{r \leftarrow \rho(f(\mathbf{x}_n))\}_{n=1}^N$
    compute $r_\alpha \leftarrow \text{quantile}(\alpha, \{r_n\}_{n=1}^N)$
    compute $r_{i+1} \leftarrow \text{maximum}(r_{\text{fail}}, r_\alpha)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_n, r_n)\}_{n=1}^N$
    $\theta_{i+1} \leftarrow \textsc{Train}(\theta_i, \{(\mathbf{x}, r) \in \mathcal{D} \mid r \leq r_i\})$
    $i \leftarrow i + 1$
  return θ_i

5.3.4 *Implementation*

For this work, we use a diffusion model based on Nichol et al. [81]. We use a simple Unet neural network for ϵ_θ with the 2-D convolutional layers replaced with 1-D convolutional layers for time signals. We condition the network by concatenating the desired robustness to the network input. We use a cosine noise schedule for β_k with $K = 1000$ steps. We train the model using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 64. We train the model for 100 epochs.

5.4 Experiments

This section covers the experimental setup used to evaluate the proposed approach. First, we discuss validation problems. Next, we discuss baselines, metrics, and experimental setup.

5.4.1 Validation Problems

We demonstrate the feasibility of our diffusion-model based validation framework using a two-dimensional toy problem and four robotics validation problems.

2D toy. The toy problem is adapted from Sinha et al. [25]. Disturbances are sampled from a unit 2D Gaussian, and robustness is defined as $\rho(\mathbf{x}) = 3.0 - \text{minimum}(|x_0|, x_1)$. Failure occurs when $\rho(\mathbf{x}) \leq 0$. This problem has two failure modes, corresponding to the regions where $(x_0 \leq -3, x_1 \geq 3)$ and where $(x_0 \geq 3, x_1 \geq 3)$.

Pendulum. We consider an underactuated, inverted pendulum environment from Gymnasium[1] that is stabilized by a PD-controller subject to input torque disturbances. We assume that the torque perturbations at each time step are distributed according to $\mathcal{N}(\mathbf{0}, 0.5\mathbf{I})$, and limit the trajectories to 100 time steps. Trajectory robustness is defined as the maximum absolute deflection of the pendulum from vertical, and we define failures to occur when the absolute deflection exceeds 30 deg from vertical. The system has two failure modes where the pendulum can fall left or right.

Lunar lander We validate a heuristic control policy for the continuous LunarLander environment from Gymnasium. In this problem, the agent controls a lunar lander on final descent to gently land at a goal location. The state of the environment is represented by an eight-tuple $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}, \text{leg}_1\text{leg}_2)$ where x is the horizontal position, y is the vertical position and θ is the orientation of the lander. Additionally,

[1] https://github.com/Farama-Foundation/Gymnasium

leg_1 and leg_2 represent are binary indicators of whether the first and second landing legs, respectively, are in contact with the ground. The control policy can fire a main thruster that applies longitudinal force smaller engines that control the angular rate. The control policy is shown in algorithm 5.2. The heuristic policy first computes a target angle and altitude based based on the current state, and applies control actions to minimize the error between the target and current state.

We add noise to the lander's observations of its horizontal position and orientation over 200 time steps, giving this problem 400 dimensions. Disturbances for position and orientation are zero-mean Gaussians with a variance of 0.31. We define failure to occur when the lander touches down outside of the landing pad. The landing pad is 0.25 units wide and centered at $x = 0$, $y = 0$. The trajectory robustness corresponds to the distance between the center of the lander to the edge of the landing pad at the end of the horizon.

Algorithm 5.2 Heuristic Lunar Lander Policy

procedure LANDERHEURISTICPOLICY($x, y, \dot{x}, \dot{y}, \theta, \dot{\theta}, leg_1, leg_2$)
  $\theta_{\text{targ}} \leftarrow 0.5x + \dot{x}$ ▷ Compute target angle
  $\theta_{\text{targ}} \leftarrow \text{clamp}(\theta_{\text{targ}}, -0.4, 0.4)$ ▷ Clamp between -0.4 and 0.4 radians
  $h_{\text{targ}} \leftarrow 0.55|x|$ ▷ Compute target altitude
  $\theta_{\text{err}} \leftarrow 0.5(\theta_{\text{targ}} - \theta) - \dot{\theta}$ ▷ Compute error
  $h_{\text{err}} \leftarrow 0.5(h_{\text{targ}} - y) - 0.5\dot{y}$
  if leg_1 **or** leg_2 ▷ If any leg has contact
    $\theta_{\text{err}} \leftarrow 0$
    $h_{\text{err}} \leftarrow -0.5\dot{y}$
  $a \leftarrow [20h_{\text{err}} - 1, -20\theta_{\text{err}}]$ ▷ Compute longitudinal and lateral actions
  $a \leftarrow \text{clamp}(a, -1, 1)$
  return a

F-16 Ground Collision Avoidance. Finally, we validate an automated ground collision avoidance system (GCAS) for the F-16 aircraft. We add disturbances to observations of orientation and angular velocity. The aircraft starts at an altitude of 600 ft, pitched down 30 deg, and is rolled 20 deg. We noise the 6 observations for 200 time steps, giving this problem 1200 dimensions.

5.4.2 Baselines

We compare DiFS against two baseline validation techniques. First, we consider the cross-entropy method with a two-component Gaussian mixture model proposal (CEM-2). CEM-2 iteratively updates the parametric proposal by minimizing the KL divergence to the failure distribution [37]. Finally, we also consider adaptive stress testing (AST), which is a validation algorithm for sequential systems that frames validation as a Markov decision process (MDP) [26]. At each time step, the agent observes the system state and chooses an input disturbance. We solve the MDP using the proximal policy optimization reinforcement learning algorithm implemented in the stable-baselines library.[2]

5.4.3 Metrics

We quantitatively evaluate the samples of state trajectories generated by DiFS and the baselines after training using failure sample density, coverage, and failure rate metrics. The density and coverage metrics of Naeem et al. [82] evaluate failure sample similarity to the true failure distribution, while the failure rate evaluates sample efficiency.

Density. The sample density metric, denoted D_τ, measures the fraction of ground-truth sample neighborhoods that include the generated samples. Neighborhoods are defined by the k-nearest neighbors. Density is unbounded and may be greater than 1 if the true samples are densely clustered around the generated samples. Given M generated samples $\{y_j\}_{j=1}^{M}$ and N ground-truth samples $\{x_i\}_{i=1}^{N}$, the density is computed as

$$D_\tau = \frac{1}{kM} \sum_{j=1}^{M} \sum_{i=1}^{N} \mathbb{1}\{y_j \in B\left(x_i, \mathrm{NND}_k\left(x_i\right)\right)\} \tag{5.9}$$

where k is the number of nearest neighbors for the nearest neighbor distance (NND) computation, and $B(x, r)$ is the ball of radius r centered at x. Intuitively, density

[2] https://github.com/DLR-RM/stable-baselines3

measures how well the spatial density of generated system trajectories matches those from the ground-truth samples collected using Monte Carlo sampling. Larger density values closer to 1 indicate better performance.

Coverage. Coverage, denoted by C_τ, measures the fraction of true samples whose neighborhoods contain at least one generated sample. Coverage is bounded between 0 and 1, with 1 being perfect coverage. The density is computed as:

$$C_\tau = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\exists j \text{ s.t. } y_j \in B\left(x_i, \text{NND}_k\left(x_i\right)\right)\} \tag{5.10}$$

Intuitively, coverage measures how well the generated samples cover the different modes of the true failure distribution. Larger coverage values closer to 1 indicate better mode coverage.

Failure rate. The failure rate R_{fail} is the proportion of samples drawn after result in failure after training. While density and coverage measure how well the sampled failures match the true distribution, it is also important to avoid wasting time evaluating safe samples. A failure rate of 1 means that all generated samples result in failure, indicating high efficiency.

5.4.4 Experimental Procedure

All experiments were performed on a machine with a single GPU with 11 GB of GPU memory. Training hyperparameters used for DiFS are shown in Table 5.1. Due to computational constraints, a full hyperparameter search is infeasible. However, empirically we find that experiments with the more challenging environments (i.e., lander and F-16) benefit from from a larger sample budget and smaller α. All methods use the same number of simulations across 5 random seeds.

Density and coverage metrics require ground-truth samples. For each problem, we collect 1000 ground-truth samples from the failure distribution using Monte Carlo sampling. We evaluate coverage and density using 1000 failure samples from

Table 5.1: Training hyperparameters for DiFS.

Hyperparameter	Toy	Pendulum	Lander	F-16
Sample budget	50,000	50,000	100,000	100,000
Samples / iter.	10,000	10,000	10,000	10,000
Train steps / iter.	10,000	10,000	20,000	20,000
α	0.5	0.5	0.5	0.3

Table 5.2: Results of Monte Carlo sampling for each problem.

	Samples	Failure Probability Estimate	Sampling Time (days)
Toy	1×10^7	3.5×10^{-5}	0.0
Pendulum	2×10^7	2.1×10^{-4}	3.5
Lander	2×10^9	5.6×10^{-7}	520
F-16	1×10^7	1.1×10^{-4}	4.7

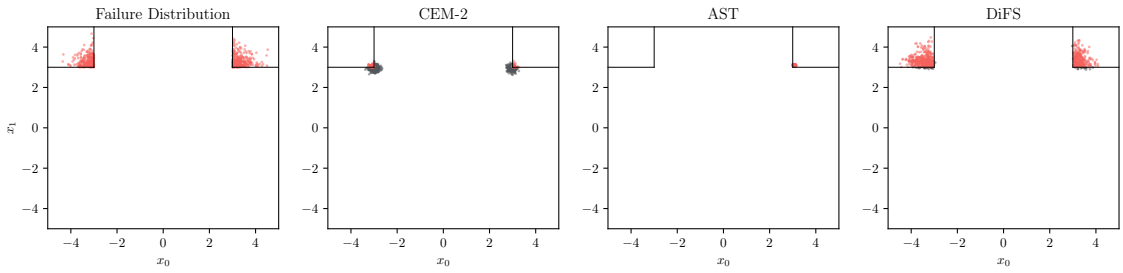


Figure 5.2: Samples from the true failure distribution and each method for the toy 2D corners problem. The failure region is outlined in black.

each method. Naeem et al. suggest a method for selecting the number of nearest neighbors using two groups of ground-truth samples. Using this method, we select $k = 5$. The failure rate R_{fail} is evaluated using 1000 samples.

5.5 *Results*

Quantitative results for all methods are shown in Table 5.3. DiFS generally outperforms baselines in failure rate, and also achieves a lower variance. The failure rate of DiFS is greater than 0.66 even on high-dimensional problems with low failure

Table 5.3: Results comparing DiFS, CEM-2, and AST across various problems. Higher is better for all metrics except training time.

	Method	$D_\tau \uparrow$	$C_\tau \uparrow$	$R_{\text{fail}} \uparrow$	Time (hr)
Toy	**DiFS**	$\mathbf{0.998 \pm 0.012}$	$\mathbf{0.980 \pm 0.020}$	$\mathbf{0.896 \pm 0.047}$	0.61
	CEM-2	0.636 ± 0.371	0.257 ± 0.150	0.182 ± 0.251	0.01
	AST	0.501 ± 0.462	0.108 ± 0.067	0.868 ± 0.107	0.13
Pendulum	**DiFS**	$\mathbf{0.872 \pm 0.068}$	$\mathbf{0.901 \pm 0.042}$	$\mathbf{0.795 \pm 0.091}$	2.93
	CEM-2	0.453 ± 0.134	0.473 ± 0.004	0.599 ± 0.036	0.22
	AST	0.578 ± 0.001	0.116 ± 0.021	0.337 ± 0.078	2.58
Lander	**DiFS**	$\mathbf{0.653 \pm 0.065}$	$\mathbf{0.876 \pm 0.043}$	0.667 ± 0.071	7.24
	CEM-2	—	—	0.0	0.74
	AST	0.113 ± 0.082	0.031 ± 0.016	$\mathbf{0.912 \pm 0.065}$	9.22
F-16	**DiFS**	$\mathbf{0.886 \pm 0.031}$	$\mathbf{0.981 \pm 0.005}$	$\mathbf{0.687 \pm 0.035}$	7.59
	CEM-2	0.742 ± 0.361	0.341 ± 0.359	0.424 ± 0.411	0.66
	AST	0.156 ± 0.302	0.002 ± 0.003	0.410 ± 0.489	12.85

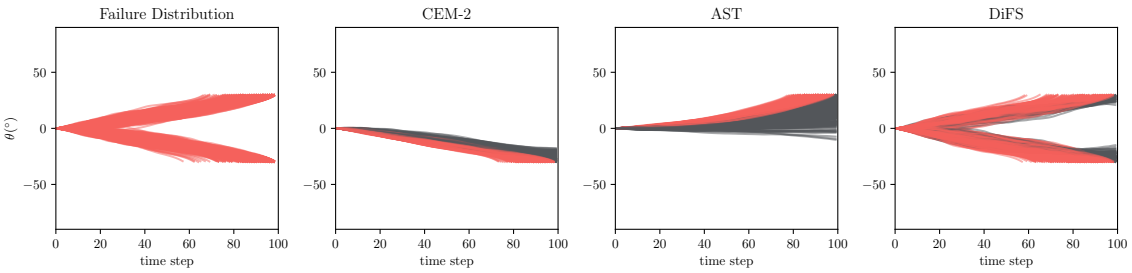


Figure 5.3: Inverted pendulum deflection angle θ trajectories sampled from the true failure distribution and sampled from each method after training. Failure occurs when the deflection angle exceeds 30 deg.

probability such as the lander and F-16. This high sample efficiency is due to the expressivity of the diffusion model, especially compared to the simple Gaussian proposal of CEM. AST also achieves a high failure rate for problems like the lunar lander because it optimizes a policy to sample a single most likely failure event. However, this generally leads to poor approximation of the failure distribution, which can be seen in lower density and coverage metric across all experiments.

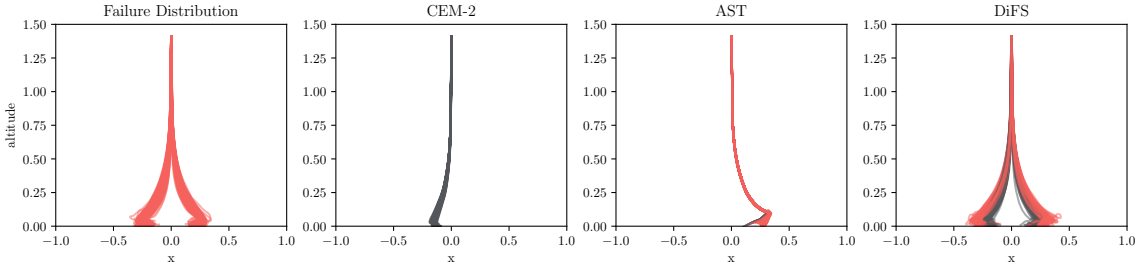


Figure 5.4: Lunar lander trajectories of altitude and horizontal position sampled from the true failure distribution and sampled from each method after training. Failure occurs when the absolute value of the horizontal position at landing exceeds 0.25, leading to two failure modes.

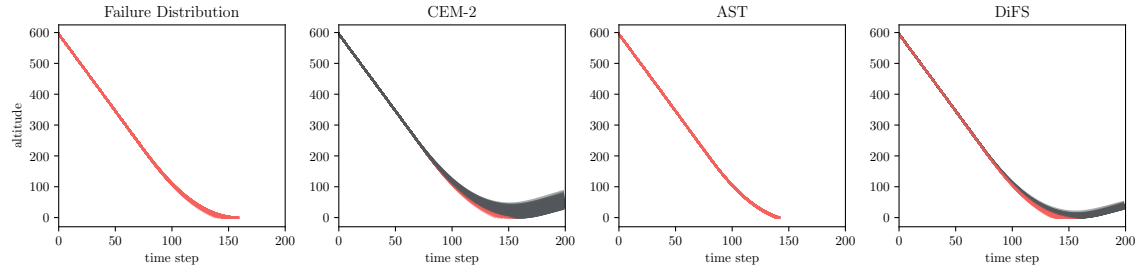


Figure 5.5: F-16 aircraft altitude and pitch angle sampled from the true failure distribution and sampled from each method after training. Failure occurs when the aircraft collides with the ground.

DiFS outperforms baselines in terms of density and coverage metrics on all problems, indicating that DiFS better captures the true failure distribution. The high density values indicate that the DiFS samples have a higher fidelity than the baselines. The coverage achieved by DiFS is consistently higher than the baseline methods, indicating that DiFS reliably discovers multimodal failure distributions. CEM and AST have coverage below 0.5 on multimodal problems like the toy, pendulum, and lander, indicating that these baselines are prone to mode collapse.

We also report the mean wall-clock time for each method in Table 5.3. CEM exhibits the fastest performance due to its efficient training process, which primarily involves fitting a GMM. In contrast, both DiFS and AST typically require hours of computation time since they involve training machine learning models. However,

the increased runtime of DiFS comes with a large improvement in failure sampling, especially when the problem is high-dimensional.

2D toy. Figure 5.2 shows the samples from each method on the 2D toy problem. Failure samples are shown in red, while safe samples are shown in gray. The true failure distribution is multimodal, with two modes corresponding to the two corners of the 2D space. DiFS captures both modes, while CEM and AST only capture one mode. The samples generated by DiFS visually match the ground truth samples collected by MC. The multimodal nature of the failure distribution makes it difficult for CEM and AST to sample failures, leading to lower density and coverage metrics. We also include a video illustrating DiFS over several iterations on the 2D toy problem.[3]

Inverted pendulum. Figure 5.3 compares system trajectories sampled using each method on the inverted pendulum problem. The true failure distribution is bimodal, with two modes corresponding to the two directions in which the pendulum can fall over. Again, DiFS captures both of these modes, while CEM and AST methods only capture one mode of the failure distribution. Visually, the samples generated by DiFS match the ground truth samples well, which is reflected in the higher density and coverage metrics.

Lunar lander. Figure 5.4 shows the results on the lunar lander problem. The trajectories show the lander's altitude versus horizontal position. The failure trajectories in red are those that land outside of the landing pad. The true failure distribution is multimodal, with two modes corresponding to the two sides of the landing pad. This environment is relatively expensive to simulate. MC sampling took 520 CPU days to collect 1000 failure samples. In contrast, DiFS learns to generate samples from both failure modes in around 9 hours.

[3] `https://youtu.be/3gQaHAYIkXA`

F-16 GCAS. Finally, sampled trajectories for the F-16 GCAS are shown in Figure 5.5. In this case the failure distribution appears to be unimodal. However, DiFS still outperforms the baselines in terms of density and coverage. The F-16 environment is also expensive to simulate, taking 4.7 CPU days to collect 1000 failure samples compared to 7.5 hours for DiFS.

While DiFS outperforms the baselines on all problems, the greatest differences can be seen for high-dimensional problems with low failure probabilities. Training the diffusion model for DiFS greatly benefits from GPU availability, while evaluating costly environments in parallel requires multiple CPUs. For the more computationally expensive problems like the lander and F-16, both training stages require similar wall-times. DiFS is therefore especially useful on systems with GPU and CPU resources.

5.6 Discussion

In this chapter, we proposed a method for failure sampling in high-dimensional black-box autonomous systems using diffusion models. Our algorithm adaptively trains a conditional diffusion model to approximate the failure distribution. We condition the diffusion model on a desired robustness level, and train the model using pairs of sampled disturbances and their simulated robustness. During training, we uniformly sample robustness levels between the current and target robustness levels to generate new disturbance trajectories from the model. We show that this training with a conditional model efficiently learns to sample from the distribution over failures even in very high-dimensional and multimodal problems. We demonstrate the effectiveness of our approach on a variety of validation problems, including a 2D toy problem, an inverted pendulum, a lunar lander, and an F-16 aircraft. Our results show that our method outperforms existing baselines in terms of failure rate, density, and coverage metrics.

One of the key takeaways from this work is that conditional models like diffusion models can effectively learn global information about the relationship between the

disturbance trajectories and robustness. Crucially, they can learn the inverse relationship, which is the distribution over disturbances conditioned on robustness. This is a very difficult problem to solve using traditional methods like CEM, which only learn local information about the failure distribution. The ability to learn this inverse relationship creates the potential for efficient exploration of the failure distribution, which is critical for high-dimensional problems with low failure probabilities.

In the next chapter, we transition from failure sampling to failure probability estimation. Similar to this work, we will take a black-box approach, but assume that we can access per-timestep state information in the simulator. This enables us to learn a proposal distribution that builds trajectories in a sequential manner. These state-dependent proposals prove to be very sample-efficient for estimating the probability of failure.

6 State-dependent Importance Sampling

In this chapter, we propose a novel method for estimating the probability of failure of autonomous systems. To address the challenges of constructing sample efficient proposal distributions for importance sampling in long time horizons, we propose a method that builds trajectories sequentially. First, we provide a brief background on importance sampling for sequential systems. Next, we describe our approach, which optimizes a state-dependent proposal distribution by minimizing the forward KL divergence to an approximate distribution over failure trajectories. We estimate gradients of the objective using a Markov score ascent method, which approximates samples from the target distribution using MCMC. We demonstrate the efficacy of our method on four validation problems including examples from autonomous driving and aviation. Our method consistently achieves lower empirical bias and variance in the estimated failure probability compared to baselines.

6.1 Motivation

Estimating the probability of failure for autonomous systems is a critical task in safety validation. Estimates of the probability of failure can provide a measure of the risk of deploying the system in a certain context. This information may be especially useful to system designers and regulators in safety-critical domains while making decisions about system deployment. Additionally, estimating the probability of failure also encompasses other important safety validation tasks like falsification and failure sampling, which can help identify weaknesses in the system and uncover potentially dangerous scenarios.

There are three key challenges associated with estimating the probability of failure in sequential autonomous systems. First, failures tend to be rare when the system is designed to be safe. Estimating the probability of failure using simple methods like Monte Carlo sampling may require an enormous number of expensive simulations. Second, the search space over failure events is very high-dimensional because autonomous systems operate over large state spaces and long time horizons. Third, autonomous systems can exhibit multimodal failures, requiring validation techniques that can adequately capture diverse behaviors.

Three common techniques for failure probability estimation are Sequential Monte Carlo (SMC) [83], multilevel splitting [32], and importance sampling [84]. SMC and splitting methods typically rely on Markov chain Monte Carlo (MCMC) methods to directly sample many failure trajectories which are then used to estimate the failure probability. These approaches may require domain knowledge to perform well [33], [85] or may require the system under test to be differentiable to scale to larger problems [25]. In contrast, our method uses MCMC only to help optimize our proposal distribution through Markov score ascent, rather than for generating samples for estimation. This key distinction allows our approach to avoid the scalability limitations of direct MCMC sampling.

Adaptive Importance Sampling (AIS) methods iteratively improve a proposal distribution, often by minimizing the KL divergence between the proposal and the true failure distribution [86], [87]. Existing AIS methods face challenges with long time horizons, because they typically model only a small set of input parameters using simple exponential family proposals [88], making them intractable for sequential problems. Our approach addresses these challenges by decomposing the high-dimensional sampling problem using a sequential state-dependent proposal distribution. Importance sampling algorithms have been extended to sequential problems by incorporating dynamic programming, but this approach is limited to small discrete state spaces [43]. Recent work also builds on policy gradient methods for AIS, but face challenges in scaling due to high variance gradient estimates. [89]. In this work, we propose an AIS algorithm for sequential problems with continuous state spaces and long horizons that overcomes these limitations by optimizing a

flexible state-dependent proposal distribution using Markov score ascent to estimate lower-variance objective gradients.

In this work, we aim to enable efficient failure probability estimation for black-box sequential systems through importance sampling. Inspired by previous work, we decompose the high-dimensional sampling problem over system trajectories using a sequential state-dependent proposal distribution. We optimize the proposal by minimizing the forward Kullback–Leibler (KL) divergence between the sequential proposal and an approximate distribution over failure trajectories. We estimate gradients of the objective using a Markov score ascent method [90], [91], which approximates samples from the target distribution using MCMC. In addition to optimizing the proposal, the algorithm uses these samples to compute an importance sampling estimate of the probability of failure. The impact of our approach is illustrated in Figure 6.1. In summary, our contributions are as follows:

- We propose a sequential state-based importance sampling proposal for failure probability estimation of autonomous systems.
- We optimize the proposal by minimizing the forward KL divergence to a tractable approximation of the optimal importance sampling distribution.
- We demonstrate that the proposed approach achieves more accurate estimates of the probability of failure compared to baselines on four sample problems.

6.2 Preliminaries

In this section, we provide necessary background on safety validation for sequential autonomous systems, failure probability estimation, and importance sampling.

6.2.1 Safety Validation for Sequential Systems

In safety validation, we search for failure events by controlling sources of randomness in the state transitions, observations, and system under test. We use disturbances, denoted by x, to control sources of randomness. We assume disturbances are

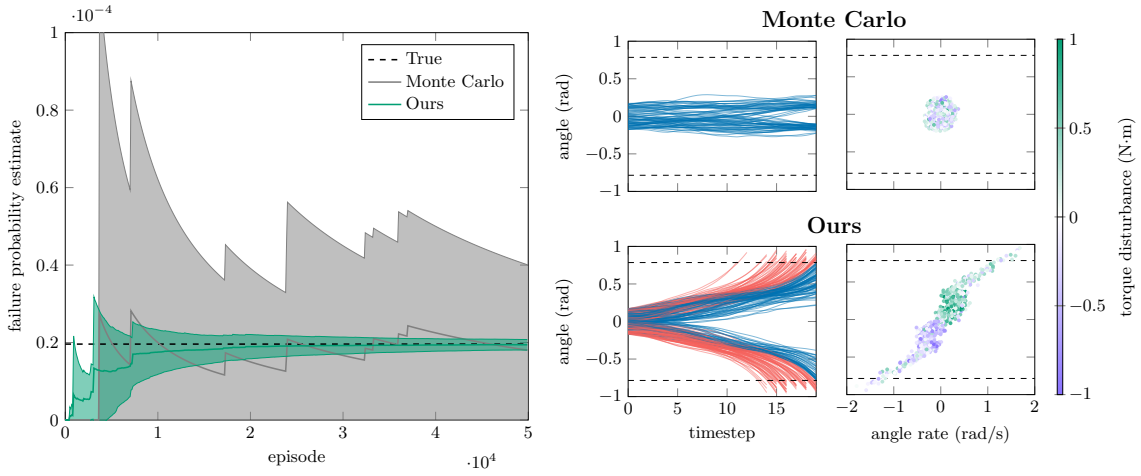


Figure 6.1: Comparison of the proposed approach and Monte Carlo (MC) sampling for estimating failure probabilities in an inverted pendulum with a rule-based controller under torque disturbances. Failures occur when the magnitude of the pendulum's angle from the vertical exceeds a threshold. The left figure shows the estimated failure probability using our method during training compared to MC. Our method is more accurate with fewer samples than MC. The middle two figures compare 200 pendulum angle trajectories over time using samples from MC and from ours after training. Our method discovers failure trajectories (red) across modes, while all MC samples are safe (blue). The two right figures show torque disturbances applied at each state in the sampled trajectories. Our method learns to add positive torque disturbances for positive angles (left from vertical) and vice versa, enabling efficient sampling for failure probability estimation.

continuous values in the space $\mathcal{X} \subseteq \mathbb{R}^{D_x}$ where $D_x \in \mathbb{N}$ is the dimension of the disturbance space. Given a state s_t and disturbance x_t at time t, we can simulate the next timestep $t+1$ according to

$$o_t \sim O(\cdot \mid s_t, x_t), \quad a_t \sim \pi(\cdot \mid o_{1:t}, x_t), \quad s_{t+1} \sim T(\cdot \mid s_t, a_t, x_t) \tag{6.1}$$

where different components of the disturbance x_t may be applied to the observation, system under test, and transition. Given a set of disturbances up to horizon d, we can simulate the system under test with disturbances to compute a system trajectory $\tau = (s_1, a_1, o_1, x_1, \ldots, s_d, a_d, o_d, x_d)$.

We assume that disturbances are sampled according to a distribution that may depend on the current system state $D(x \mid s)$. The disturbance distribution reflects how likely different disturbances are to occur in the real operating environment, and is typically designed using real-world data or expert knowledge. This formulation could also consider disturbances that depend on observations and actions, but we focus only on state-dependent disturbances for simplicity. Using the disturbance distribution, we can define a distribution over system trajectories

$$p(\tau) = p(s_1) \prod_{t=1}^{d} T(s_{t+1} \mid s_t, a_t, x_t) O(o_t \mid s_t, x_t) \pi(a_t \mid o_{1:t}, x_t) D(x_t \mid s_t) \tag{6.2}$$

where $p(s_1)$ is the initial state distribution. We denote this the nominal trajectory distribution since it represents the anticipated trajectory distribution in operation. Note that under this formulation, we do not necessarily assume that disturbances directly control all sources of randomness in the simulation. Instead, we allow some sources of randomness to be controlled by the system itself, such as the policy π or initial state distribution.

We evaluate the safety of system trajectories using an evaluation function $\rho(\tau) \in \mathbb{R}$. When $\rho(\tau)$ is less than a failure threshold γ, we say that the trajectory is a failure event, with lower values indicating greater proximity to failure. We can express the

distribution over all failure trajectories as the conditional distribution

$$p(\tau \mid \rho(\tau) \leq \gamma) = \frac{p(\tau)\mathbb{1}\{\rho(\tau) \leq \gamma\}}{\int p(\tau')\mathbb{1}\{\rho(\tau') \leq \gamma\}d\tau'} \tag{6.3}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and the denominator integrates over all trajectories. The denominator is the total probability of failure. Due to the high dimensionality of trajectories and the complexity of autonomous systems, exact computation of this quantity is intractable. Instead, we rely on black-box evaluations of the system to estimate this probability.

6.2.2 *Failure Probability Estimation*

Failure probability estimation is the problem of estimating the denominator of Equation (6.3), or the expected value:

$$P_{\text{fail}} = \mathbb{E}_{p(\tau)}\left[\mathbb{1}\{\rho(\tau) \leq \gamma\}\right] \tag{6.4}$$

A simple approach to estimate P_{fail} is using Monte Carlo (MC) sampling, which uses N independent samples from $p(\tau)$ to compute the empirical estimate

$$\hat{P}_{\text{fail}}^{\text{MC}} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\{\rho(\tau_i) \leq \gamma\} \tag{6.5}$$

For very rare failures (e.g. $P_{\text{fail}} \approx 10^{-9}$), standard Monte Carlo may require billions of samples, making direct estimation infeasible for complex systems.

6.2.3 *Importance Sampling*

Importance sampling can reduce the variance of failure probability estimates by focusing samples on regions of interest. IS draws samples from a proposal distribution

$q(\tau)$, and computes an estimate

$$\hat{P}_{\text{fail}}^{\text{IS}} = \frac{1}{N} \sum_{i=1}^{N} w(\tau_i) \mathbb{1}\{\rho(\tau_i) \leq \gamma\} \tag{6.6}$$

where the samples are weighted according to their importance weight $w(\tau) = p(\tau)/q(\tau)$. The minimum variance importance sampling distribution is the failure distribution itself, meaning that samples are only drawn from the failure region with likelihood proportional to $p(\tau)$. The estimator is unbiased if $q(\tau) > 0$ whenever $\mathbb{1}\{\rho(\tau_i) \leq \gamma\} p(\tau) > 0$.

The key challenge of importance sampling is designing a good proposal distribution. The variance of the estimator increases roughly exponentially with the problem dimension, but it can be reduced by a well-designed proposal distribution. However, modeling the complex $D_x d$-dimensional trajectory distributions following previous IS approaches is extremely difficult.

6.3 Methods

Next, we introduce our method for failure probability estimation in sequential systems. We propose a sequential state-based IS proposal, discuss proposal optimization, and detail our algorithm.

6.3.1 Sequential Proposal Distribution

To address the challenges of fitting a complex, high-dimensional proposal distribution, we decompose the problem using a proposal applied at each timestep. Since the system behavior and disturbance distribution may depend on the system state, we condition the distribution on the current state. These proposals take the form $q_\theta(x \mid s)$ with parameters θ. We can draw samples from $q_\theta(\tau)$ by sampling an initial

state $s_1 \sim p(s_1)$ and simulating each timestep:

$$x_t \sim q_\theta(\cdot \mid s_t), \quad o_t \sim O(\cdot \mid s_t, x_t), \quad a_t \sim \pi(\cdot \mid o_{1:t}, x_t), \quad s_{t+1} \sim T(\cdot \mid s_t, a_t, x_t) \tag{6.7}$$

System trajectories sampled using the proposal $q_\theta(x \mid s)$ have the density:

$$q_\theta(\tau) = p(s_1) \prod_{t=1}^{d} T(s_{t+1} \mid s_t, a_t, x_t) O(o_t \mid s_t, x_t) \pi(a_t \mid o_{1:t}, x_t) q_\theta(x_t \mid s_t) \tag{6.8}$$

In computing the importance weight as the ratio of Equation (6.2) to Equation (6.8), all terms cancel except the disturbance distribution and the proposal. The importance weight reduces to:

$$w(\tau) = \prod_{t=1}^{d} \frac{D(x_t \mid s_t)}{q_\theta(x_t \mid s_t)} \tag{6.9}$$

This formulation reduces the problem of learning a proposal over full disturbance trajectories to learning a proposal over a single timestep given the current state. However, obtaining a reliable IS estimate of the failure probability still requires careful design of this proposal.

6.3.2 Proposal Optimization

The optimal importance sampling distribution is known to be proportional to the true distribution over failure trajectories. Therefore, we propose to learn the proposal parameters by minimizing the forward KL divergence between the proposal and the failure distribution:

$$\theta^* = \arg\min_\theta D_{KL}\left(p\left(\tau \mid \rho(\tau) \leq \gamma\right) \,||\, q_\theta\left(\tau\right)\right) \tag{6.10}$$

Minimizing the forward KL divergence is desirable for importance sampling because its mass covering properties help ensure the IS estimator remains unbiased [84]. This objective is equivalent up to a constant of proportionality to minimizing the

cross-entropy:

$$\min_{\theta} \mathcal{L}(\theta) = \min_{\theta} \mathbb{E}_{p(\tau|\rho(\tau)\leq\gamma)} \left[-\log q_\theta(\tau) \right] \tag{6.11}$$

However, this formulation creates a chicken-and-egg problem, since our goal is to approximate $p(\tau \mid \rho(\tau) \leq \gamma)$ with $q_\theta(\tau)$, but the cross-entropy objective involves an expectation over this same distribution. A common approach optimizes an IS estimate of the expectation. Unfortunately, the variance of the IS estimate increases rapidly with problem dimension, making optimization very difficult especially when the proposal is far from the target.

Building on previous work in variational Bayesian inference, we optimize the objective in Equation (6.11) using Markov score ascent methods [90], [91]. Markov score ascent methods can achieve much lower variance estimates of the objective. The key idea of Markov score ascent methods is to estimate the cross-entropy using samples from an MCMC kernel that is ergodic with respect to the target distribution. The kernel is chosen to directly use the approximation of the target distribution $q_\theta(\tau)$. With repeated application of the kernel, the samples will be approximately distributed according to the target. Given N trajectories approximately distributed according to the target, we may compute a Monte Carlo estimate of the objective:

$$\mathcal{L}(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} -\log q_\theta(\tau_n) \tag{6.12}$$

Expanding the objective and removing terms that do not depend on θ, the loss can be computed as

$$\mathcal{L}(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{d} -\log q_\theta(x_{t,n} \mid s_{t,n}) \tag{6.13}$$

where subscript t, n denotes timestep t in trajectory n. This estimated cross-entropy objective is optimized using stochastic gradient descent.

In this work, we construct an MCMC kernel by performing a single step of Independent Metropolis-Hastings (IMH) [92]. Given an initial trajectory τ, we can generate a new sample from IMH by first sampling a proposal trajectory $\tau' \sim q_\theta(\cdot)$. IMH either accepts the new trajectory τ' according to the Metropolis-Hastings

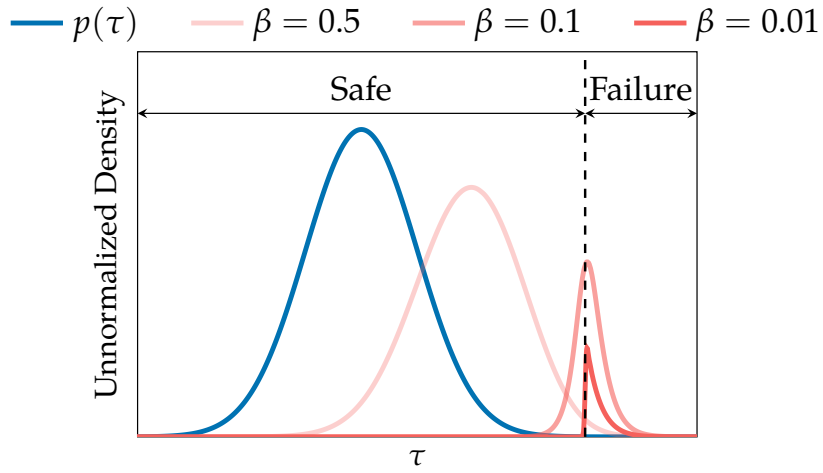


Figure 6.2: Illustration of $\tilde{p}(\tau)$.

acceptance probability a:

$$a(\tau, \tau') = \min\left(1, \frac{p(\tau' \mid \rho(\tau') \leq \gamma) q_\theta(\tau)}{p(\tau \mid \rho(\tau) \leq \gamma) q_\theta(\tau')}\right) = \min\left(1, \frac{w(\tau')}{w(\tau)}\right) \tag{6.14}$$

Otherwise, IMH rejects the proposal and τ remains the same.

One challenge in applying IMH to sample from the true failure distribution is that the failure distribution applies zero density to safe trajectories. Since failures are rare, many sampled trajectories may be safe. In turn, the acceptance probabilities will be zero, making it very difficult for the MCMC step to move trajectories closer to the failure distribution.

6.3.3 *Approximating the Failure Distribution*

To address the issue of low acceptance rates, we use a smooth approximation of the discontinuous failure distribution. Specifically, we approximate the indicator function in the target by $P_\beta(\gamma - \rho(\tau))$ where P_β is the cumulative distribution function of a logistic distribution with zero mean and scale β. This approximation replaces the discontinuous indicator with a smooth logistic curve from zero to one. The

approximate failure distribution $\tilde{p}(\tau)$ has the unnormalized density

$$\tilde{p}(\tau) \propto p(\tau) P_\beta(\gamma - \rho(\tau)) \tag{6.15}$$

which relaxes the failure distribution, enabling stable sampling during optimization. This approximation is visualized in Figure 6.2 for a few values of β.

Using the approximate posterior, our objective becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tilde{p}(\tau)} \left[-\log q_\theta(\tau) \right] \tag{6.16}$$

We can now apply IMH to generate proposal samples for Markov score ascent by computing the acceptance probability using the unnormalized importance weights under the approximate failure distribution $\tilde{w}(\tau) = \tilde{p}(\tau)/q_\theta(\tau)$. Note that we can still compute the acceptance probability using $\tilde{w}(\tau)$ without knowing the normalizing constant of the approximate failure distribution, because the constant values in the numerator and denominator in the ratio of Equation (6.14) cancel.

6.3.4 Algorithm Details

The complete algorithm, which we call State-dependent Proposal Adaptive Importance Sampling (SPAIS), is shown in algorithm 6.1. The algorithm maintains a set of N trajectories that will be updated using the IMH kernel and a buffer to store evaluations $f(\tau)$ and the corresponding IS weight of samples from $q_\theta(\tau)$. At each iteration, the algorithm samples N new proposal trajectories τ'. These samples are first added to the IS buffer. Next, they are used to update the trajectory set. Each proposed trajectory is accepted or rejected according to the MH acceptance probability. At each iteration, our algorithm performs N simulations of the system under test and requires $2N$ evaluations of the proposal. However, the cost of evaluating the proposal is typically insignificant compared to that of simulation. At the end of each iteration, we update the proposal by fitting the proposal to the updated set of trajectories using stochastic gradient descent. After a fixed number of iterations, the algorithm returns an IS estimate of the failure probability using Equation (6.6).

Algorithm 6.1 State-dependent Proposal Adaptive Importance Sampling.

function SPAIS($p(\tau)$, $\rho(\tau)$, $q_{\theta_1}(x \mid s)$, γ, N, N_{iter})
  $\tau_n \sim q_{\theta_1}(\tau)$ for $n \in 1:N$ ▷ Sample initial trajectories using Equation (6.7)
  $\mathcal{D} \leftarrow \{\rho(\tau_n),\ p(\tau_n)/q_{\theta_1}(\tau_n)\}_{n=1}^{n}$ ▷ Initialize IS buffer
  $\tilde{w}_n \leftarrow p(\tau_n)\, P_\beta(\gamma - \rho(\tau_n))/q_{\theta_1}(\tau_n)$ for $n \in 1:N$ ▷ Compute MH weights
  for $k = 1$ **to** N_{iter}
    $\tau'_n \sim q_{\theta_k}(\tau)$ for $n \in 1:N$ ▷ Sample proposal trajectories
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\rho(\tau'_n),\ p(\tau'_n)/q_{\theta_k}(\tau'_n) \mid n = 1, \ldots, N\}$ ▷ Update IS buffer
    $\tilde{w}'_n \leftarrow p(\tau'_n)\, P_\beta(\gamma - \rho(\tau'_n))/q_{\theta_k}(\tau'_n)$ for $n \in 1:N$
    $\tilde{w}_n \leftarrow p(\tau_n) P_\beta(\gamma - \rho(\tau_n))/q_{\theta_k}(\tau_n)$ for $n \in 1:N$
    $\tau_n \leftarrow \text{MHAccept}(\tilde{w}'_n, \tilde{w}_n)$ for $n \in 1:N$ ▷ Accept or reject
    $\mathcal{L}(\theta) \leftarrow \sum_{n=1}^{N}\sum_{t=1}^{d} -\log q_\theta(x_{t,n} \mid s_{t,n})$ ▷ Compute loss Equation (6.13)
    $\theta_{k+1} \leftarrow \text{SGD}(\mathcal{L}(\theta), \theta_k)$ ▷ Update proposal using gradient descent
  return $\hat{P}_{\text{fail}} \leftarrow \text{ImportanceSamplingEstimate}(\mathcal{D}, \gamma)$ ▷ Equation (6.6)

In practice, we parameterize $q_\theta(x \mid s)$ as a multivariate Gaussian

$$q_\theta(x \mid s) = \mathcal{N}(x \mid \mu_\theta(s), \Sigma_\theta(s)) \tag{6.17}$$

where $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$ are both neural networks with two hidden layers of size 64 and 32 respectively. We find that using any small value of β in the range $[10^{-4}, 10^{-2}]$ gives good performance. We use $\beta = 10^{-2}$ for all evaluations. For each problem, we normalize the state space features and pretrain $q_\theta(x \mid s)$ to match $D(x \mid s)$.

6.4 Experiments

In this section, we describe our experiments to evaluate the proposed approach. We describe baseline estimation methods, evaluation metrics, and introduce four example safety validation problems.

Problem	P_{fail}	D_s	D_x	d
Pendulum	1.96×10^{-5}	2	1	20
Crosswalk	5.90×10^{-5}	8	5	100
Collision Avoidance	2.23×10^{-5}	12	1	40
F-16 GCAS	6.59×10^{-5}	13	6	200

Table 6.1: Failure probability P_{fail}, state size D_s, disturbance size D_x, and horizon d for each problem.

6.4.1 Evaluation Metrics and Baselines

We evaluate the bias and variance of the estimated probability of failure using the relative error ϵ_{rel} and absolute relative error ϵ_{abs}, respectively:

$$\epsilon_{\text{rel}} = (\hat{P}_{\text{fail}} - P_{\text{fail}})/P_{\text{fail}}, \quad \epsilon_{\text{abs}} = |\hat{P}_{\text{fail}} - P_{\text{fail}}|/P_{\text{fail}} \tag{6.18}$$

We obtain an estimate of the ground truth failure probability μ using 10^7 Monte Carlo samples. Each metric is averaged over 10 trials with 50000 samples per trial to arrive at estimates of the empirical bias and variance for each method.

We compare the method against three baseline methods. The simplest baseline is Monte Carlo estimation. We also compare against a variant of the cross-entropy method (CEM) that optimizes a state-independent proposal distribution applied at each timestep. Finally, we evaluate against PG-AIS, which learns a reinforcement learning policy for importance sampling [89].

6.4.2 Validation Problems

We demonstrate SPAIS on the inverted pendulum, AV crosswalk, aircraft collision avoidance, and F-16 GCAS problems. A summary of each problem is reported in Table 6.1.

Inverted pendulum. We evaluate an underactuated inverted pendulum system with a nonlinear energy-based controller based on Åström et al. [93]. The controller

computes the control input as

$$a = -k_1\dot{\theta} - k_2(\dot{\theta}_{\text{target}} - \dot{\theta}) \tag{6.19}$$

where $k_1 = 2, k_2 = 1$ and $\dot{\theta}_{\text{target}} = \text{sign}(\theta)\sqrt{60(1-\cos\theta)}$. The system is subjected to additive zero-mean Gaussian torque disturbances with standard deviation of 0.2 for 20 timesteps. The initial angle of the pendulum is sampled in the range $[-\pi/8, \pi/8]$ with initial angular velocity sampled from $[-0.1, 0.1]$. We define failure to occur when the magnitude of the pendulum's angle from the vertical exceeds $\pi/4$ rad. This problem has two distinct failure modes: leftward and rightward falls.

Crosswalk. In the second experiment we evaluate an autonomous vehicle approaching a pedestrian at a crosswalk inspired by Koren et al. [67]. The AV uses the Intelligent Driver Model [47] for longitudinal control, maintaining safe distances from obstacles. We adversarially control the pedestrian's 2D acceleration, perceived position and speed as detected by the AV over 100 time steps. All disturbances are distributed according to a zero-mean unit Gaussian. Failures occur when the AV collides with the pedestrian.

Aircraft collision avoidance. The third experiment evaluates an aircraft collision avoidance system based on the AVOIDDS dataset and simulator [94]. The system detects nearby intruding aircraft and recommends pilot maneuvers (e.g. climb or descend) to avoid midair collisions with the ownship. We evaluate the collision avoidance policy in scenarios with perfect perception of a single intruding aircraft. The intruder's initial state is randomly sampled according to the AVOIDDS encounter model. We adversarially control the intruder's vertical rate to cause midair collisions with the ownship.

F-16 ground collision avoidance. Finally, we consider a model of the ground collision avoidance system (GCAS) for the F-16 aircraft adapted from Heidlauf et al. [48]. We add disturbances to observations of orientation and angular velocity over 200 timesteps. The noise on the observed angles is distributed according to a zero-mean

gaussian with standard deviation 0.01. The disturbances on the observed angle-rates are distribution according to zero-mean Gaussians with standard deviation 0.005. The aircraft starts at an altitude of 578 ft, pitched down 27 deg, and is rolled 22 deg.

6.5 Results

The relative and absolute error for each method across all problems are reported in Table 6.2. The proposed SPAIS algorithm achieves lower relative and absolute error on all four validation problems. The MC estimator generally has low bias, but high variance. The PG-AIS baseline method tends to have large variance, indicating that it struggles to represent the failure distribution. CEM exhibits a bias, which may be due to mode collapse where the proposal does not cover all failure modes or due to the proposal being spread very wide leading to large importance weights.

Inverted pendulum. In Figure 6.3, we plot the pendulum angle θ against angular rate $\dot{\theta}$, and color the points according to the torque disturbance. SPAIS generally adds positive torque disturbances for positive angles, and vice versa. In contrast, the CEM proposal only samples negative disturbances and thus only samples from one of the two failure modes.

Crosswalk. CEM and SPAIS proposals for the crosswalk problem are shown in Figure 6.4. This figure shows the 2D position of the pedestrian with points colored according to the controlled pedestrian x-axis acceleration. Both proposals steer the pedestrian into the street and towards the oncoming AV. The SPAIS proposal is more structured around an important failure mode. The trajectories are more consistent, and use some deceleration to keep the pedestrian in front of the AV for longer. In contrast, the CEM proposal is more spread out, contributing to its larger bias.

Aircraft collision avoidance. Figure 6.5 shows the proposals for the aircraft collision avoidance problem, plotting normalized vertical separation Δz against horizontal distance Δx of the ownship and intruder. The CEM proposal exhibits mode collapse,

Method	Pendulum		Crosswalk		Collision Avoidance		F-16 GCAS	
	ϵ_{rel}	ϵ_{abs}	ϵ_{rel}	ϵ_{abs}	ϵ_{rel}	ϵ_{abs}	ϵ_{rel}	ϵ_{abs}
MC	-0.39 ± 0.99	0.81 ± 0.65	0.15 ± 0.36	0.28 ± 0.27	0.30 ± 0.55	0.54 ± 0.26	0.39 ± 0.88	0.74 ± 0.70
CEM	-0.54 ± 0.05	0.53 ± 0.05	-0.22 ± 0.11	0.22 ± 0.11	-0.18 ± 0.50	0.26 ± 0.43	0.63 ± 1.75	1.13 ± 1.09
PG-AIS	-0.41 ± 0.28	0.47 ± 0.16	-0.26 ± 0.55	0.41 ± 0.44	-0.91 ± 0.10	0.91 ± 0.10	-0.91 ± 0.45	0.94 ± 0.58
SPAIS	$\mathbf{-0.04 \pm 0.06}$	$\mathbf{0.06 \pm 0.03}$	$\mathbf{-0.02 \pm 0.13}$	$\mathbf{0.09 \pm 0.09}$	$\mathbf{-0.05 \pm 0.27}$	$\mathbf{0.18 \pm 0.21}$	$\mathbf{0.06 \pm 0.10}$	$\mathbf{0.09 \pm 0.07}$

Table 6.2: Relative and absolute error for each method across all four validation problems.

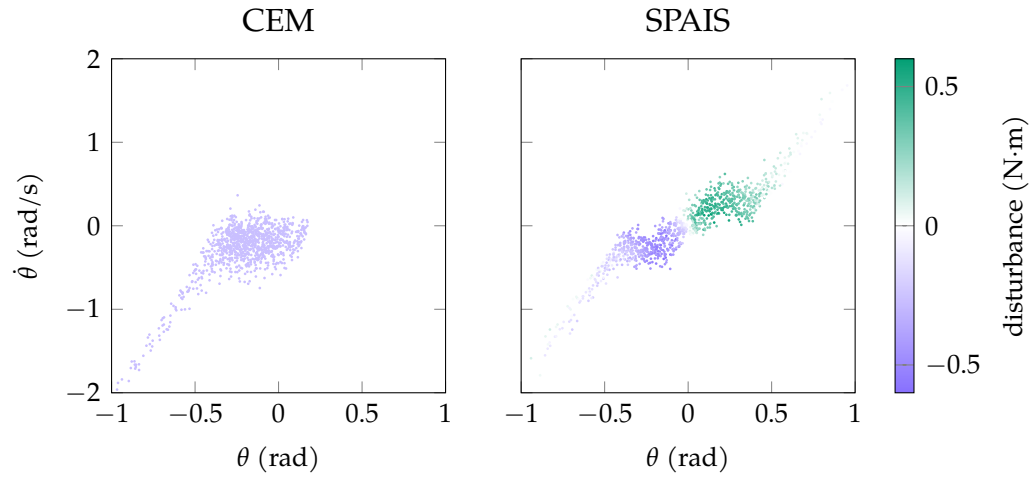


Figure 6.3: Visited states and applied disturbances by CEM (left) and SPAIS (right) on the inverted pendulum after training. SPAIS learns to apply positive disturbances for positive angles and vice versa, while CEM applies disturbances uniformly.

favoring intruder trajectories below the ownship. The SPAIS proposal symmetrically samples trajectories above and below the ownship, contributing to reduced estimation bias.

F-16 GCAS. Figure 6.6 shows F-16 GCAS trajectories of altitude h against roll angle ϕ with points colored by the observed roll angle disturbance. SPAIS increases the error in the observed roll angle just before the controller begins to pull up, causing a delay in the maneuver and more ground collisions. The mean CEM disturbance is small, suggesting that the proposal must be wide to capture the failure trajectories discovered by SPAIS.

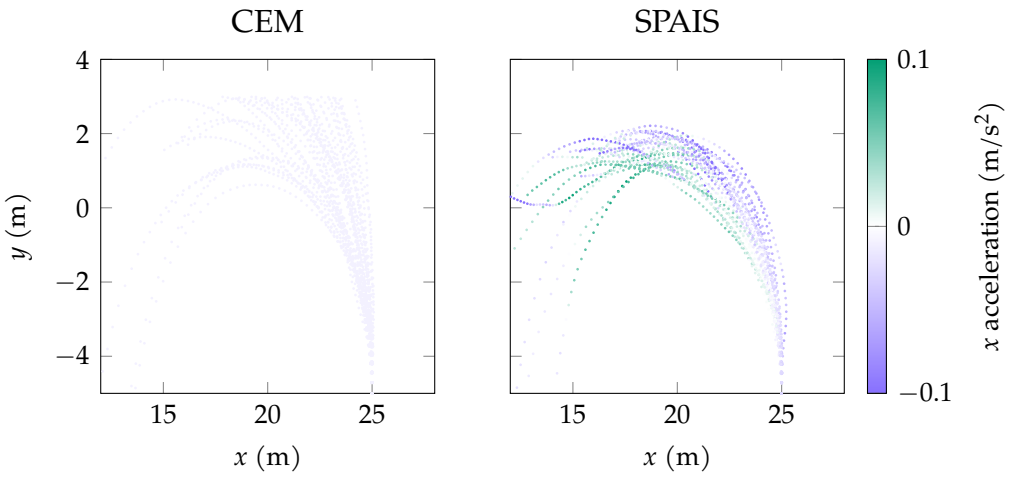


Figure 6.4: Visited positions and pedestrian x-acceleration disturbances by CEM (left) and SPAIS (right) on the crosswalk problem. The SPAIS policy generally accelerates towards the oncoming AV and then decelerates once in the street.

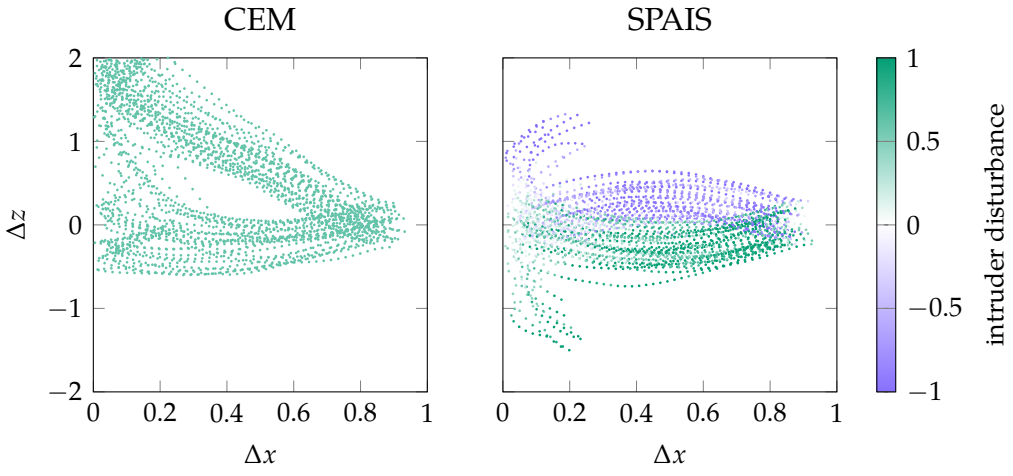


Figure 6.5: Visited states and applied vertical rate disturbances by CEM (left) and SPAIS (right) on the CAS problem. SPAIS applies either positive or negative disturbances to the intruder's vertical rate depending on the relative altitude.

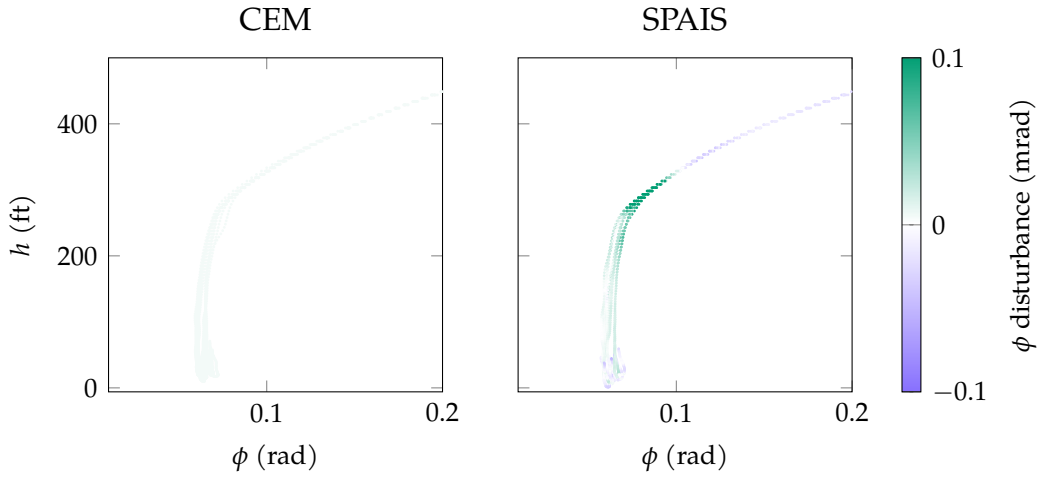


Figure 6.6: Visited roll angle vs altitude states and roll observation disturbances by CEM (left) and SPAIS (right) on the F-16 problem.

6.6 Discussion

In this chapter, we presented a method for estimating the failure probability of autonomous systems. Our method decomposes the problem of learning an importance sampling proposal over trajectories into learning a proposal over a single timestep conditioned on the current state. This state-dependent proposal distribution builds trajectories sequentially, mirroring the system's operation. Our method learns a proposal by minimizing the forward KL divergence between the proposal and the failure distribution. Markov score ascent methods provide a low-variance estimate of the KL divergence, enabling effective proposal optimization. We approximate the failure distribution with a smooth function to stabilize sampling during optimization. Using a relatively small number of samples, our results show that SPAIS achieves under 10% empirical bias with at least 2 times lower empirical variance than baselines.

A key takeaway from this work is that validation algorithms can leverage the sequential nature of autonomous systems to improve sample efficiency. When the simulation provides per-timestep state information, we can construct proposal distributions that are simple at each timestep but lead to complex trajectories. This

approach is particularly useful for high-dimensional systems where learning a single proposal over the full trajectory is difficult. This work is related to previous safety validation methods that use a dynamic programming approach to construct a failure distribution over trajectories [42], [43], [89]. These methods construct value functions over trajectories and are generally limited to discrete problems. In contrast, our method takes an approach more similar to variational Bayesian inference by learning a proposal by minimizing the KL divergence to the failure distribution.

In the final chapter of this thesis, we summarize the three core contributions of this work and discuss promising avenues for future work.

7 Summary and Future Work

This chapter summarizes the sampling and estimation methods for safety validation developed in this thesis, highlights the key contributions, and presents possible directions for future work.

7.1 Summary

While autonomous systems hold significant promise for improving safety and efficiency in safety-critical applications, these systems must be rigorously validated before deployment. However, validating these autonomous systems remains a challenge due to the complexity of the systems and the diverse ways that they may fail. Formal verification methods typically cannot scale to complex systems or stochastic environments. Real-world testing is typically expensive and may be dangerous. In this thesis, we focus on simulation-based validation methods that control stochastic elements of a simulated system to discover failure events. Simulation-based validation methods are often more scalable than formal verification methods and can be less expensive than real-world testing.

In the scope of simulation-based validation, this thesis addresses two key tasks: failure sampling and failure probability estimation. Failure sampling involves sampling from the distribution over all possible failure trajectories. Failure probability estimation involves estimating the probability of a failure event occurring. Together, these tasks aim to provide a comprehensive understanding of the system's failure modes and their likelihoods. There are three fundamental challenges that any algorithm must address when performing these tasks: high dimensionality,

multimodality, and rare events. Current methods for sampling and estimation often rely on MCMC, importance sampling, or dynamic programming to address these challenges. However, these methods still suffer from drawbacks in terms of generalizability, scalability, and sample efficiency.

Many methods for failure sampling rely on ad-hoc modifications to existing sampling algorithms based on the specific application or system under test. This tightly couples the problem of failure sampling with the solution method, making it difficult to generalize to new systems. In this thesis, we propose a model-based validation framework that decouples the problem of failure sampling from the solution method. We frame failure sampling as probabilistic inference and construct a generic probabilistic program for failure sampling. This probabilistic program can be used together with a wide variety of generic inference algorithms to perform failure sampling.

Many real-world systems or simulations may be limited to black-box access, where a validation algorithm can only input disturbance trajectories and observe the simulation result. Common approaches in this setting use MCMC or adapt simple parametric distributions for failure sampling. However, these methods often struggle to scale to very high-dimensional systems. To address this, we propose an algorithm that adaptively learns a generative model for failure trajectories using diffusion models. We show that diffusion models can scale to sample failure trajectories in problems with thousands of dimensions and multiple failure modes.

The traditional approaches to failure probability estimation rely on importance sampling and MCMC methods, which target the optimal distribution over failure trajectories. However, learning a distribution over high-dimensional trajectories can require a large number of samples. If state information is available, dynamic programming methods may also be applied but are generally limited to small discrete state spaces. To mitigate this problem, we propose a state-dependent importance sampling method that maps environment states to disturbances that cause the system to fail. The state-dependent distribution is learned using Markov score ascent methods and is a sample-efficient way to estimate failure probabilities.

7.2 Contributions

This work attempts to address several of the current challenges for sampling and estimation in simulation-based validation including generalizability, scalability, and sample efficiency. The key contributions of this thesis are as follows:

A framework for validation as probabilistic inference. In Chapter 4, we propose a model-based validation framework that decouples the problem of failure sampling from the solution method. We formulate failure sampling as a probabilistic inference problem and construct a generic probabilistic program for failure sampling. This probabilistic program can be used together with a wide variety of generic inference algorithms to perform failure sampling.

A technique for high-dimensional failure sampling using generative models. In Chapter 5, we propose a method for failure sampling that adaptively learns a generative model of failure trajectories using a diffusion model. Diffusion models are a type of deep generative model that have shown promise in modeling high-dimensional and multimodal data like images. The proposed method can scale to sample failure trajectories in problems with thousands of dimensions and multiple failure modes.

A state-dependent importance sampling method for failure probability estimation. In Chapter 6, we propose a state-dependent importance sampling method that scales to continuous state spaces. The state-dependent proposal distribution enables better sample efficiency by decomposing the sampling of long failure trajectories over time.

7.3 Future Work

The contributions of this thesis take a step toward addressing the key challenges of sampling and estimation for safety validation. However, further research is needed to expand the capabilities and utility of sampling and estimation algorithms. This section outlines some possible directions for future work:

Failure sampling with perceptual inputs. This work primarily focuses on applying high-level disturbances to the environment, agent, or sensor. For example, in the autonomous driving setting, we perturb the perceived heading of a pedestrian. However, many real world systems use perceptual sensors like camera images or lidar point clouds. Previous work has explored falsification in the context of pixel-wise disturbances in images [95] and in the context of lidar point clouds in adverse weather conditions [96]. Key challenges to failure sampling for perceptual inputs include the high dimensionality of the disturbance space and building a good model of disturbances to the perceptual inputs. Future work could explore generative models for high-dimensional image, video, or point cloud inputs, and how to effectively sample from these models [97], [98].

Improving efficiency of state-dependent importance sampling. There are theoretical parallels that can be drawn between state-dependent importance sampling and reinforcement learning for safety validation [89]. Reinforcement learning methods have shown promise in sample efficiency through a variety of techniques including off-policy learning [99], hindsight experience replay [100], and model-based approaches [101]. Future work could explore how these techniques could be applied to state-dependent importance sampling to further improve sampling efficiency for learning importance sampling distributions.

Large language models for interpretability. One key challenge in integrating safety validation methods with industry applications is the limited interpretability of the high-dimensional failure examples. Engineers may wish to gain deeper insight about why certain failure modes occur to improve the system, or share safety validation results with stakeholders. Large language models (LLMs) offer promising opportunities to enhance the interpretability of safety validation results. After sampling failure trajectories, LLMs could be used to generate natural language descriptions that explain failure modes in terms humans can easily understand, similar to how they've been used to explain other technical concepts [102]. Previous work on interpretable safety validation optimized a signal temporal logic (STL) formula

to describe disturbance trajectories [103]. LLMs could facilitate this process by generating STL formulas based on natural language descriptions of the safety validation problem. Using LLMs to provide common-sense priors for generating realistic disturbance trajectories during validation may accelerate the validation process and make validation results more accessible to system designers and stakeholders.

Bibliography

[1] Federal Aviation Administration (FAA), ''Introduction to TCAS II: Version 7.1,'' Tech. Rep., 2011.

[2] T Arino, K Carpenter, S Chabert, H Hutchinson, T Miquel, B Raynaud, K Rigotti, and E Vallauri, ''Studies on the safety of ACAS II in europe,'' *Eurocontrol, Technical Rep. ACASA/WP-1.8 D*, vol. 210, 2002.

[3] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, ''Next-generation airborne collision avoidance system,'' *Lincoln Laboratory Journal*, vol. 19, no. 1, pp. 17–33, 2012.

[4] National Highway Traffic Safety Administration (NHTSA), ''Traffic safety facts 2022: A compilation of motor vehicle traffic crash data,'' Tech. Rep., 2022.

[5] K. D. Kusano, J. M. Scanlon, Y.-H. Chen, T. L. McMurry, R. Chen, T. Gode, and T. Victor, ''Comparison of Waymo rider-only crash data to human benchmarks at 7.1 million miles,'' *Traffic Injury Prevention*, vol. 25, S66–S77, 2024.

[6] National Transportation Safety Board (NTSB), ''Collision between vehicle controlled by developmental automated driving system and pedestrian,'' Tech. Rep., 2018. [Online]. Available: `https://www.ntsb.gov/investigations/accidentreports/reports/har1903.pdf`.

[7] California Department of Motor Vehicles. ''Dmv statement on Cruise LLC suspension,'' Accessed: Mar. 5, 2025. [Online]. Available: `https://www.dmv.ca.gov/portal/news-and-media/dmv-statement-on-cruise-llc-suspension/`.

[8] H. Yang and L. Carlone, ''Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization,'' *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 3, pp. 2816–2834, 2022.

[9] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, ''Control barrier functions: Theory and applications,'' in *2019 18th European Control Conference (ECC)*, 2019.

[10] S. Kolathaya and A. D. Ames, ''Input-to-state safety with control barrier functions,'' *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 108–113, 2019.

[11] X. Guo and Y. Zhang, ''Maturity in automated driving on public roads: A review of the six-year autonomous vehicle tester program,'' *Transportation Research Record*, vol. 2676, no. 11, pp. 352–362, 2022.

[12] R. C. Rorie and C. L. Smith, ''Detect and avoid and collision avoidance flight test results with acas xr,'' in *Digital Avionics Systems Conference (DASC)*, 2024.

[13] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer, ''A survey of algorithms for black-box safety validation of cyber-physical systems,'' *Journal of Artificial Intelligence Research*, vol. 72, no. 2005.02979, pp. 377–428, 2021.

[14] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, ''Efficient guiding strategies for testing of temporal properties of hybrid systems,'' in *NASA Formal Methods Symposium (NFM)*, 2015.

[15] E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem, et al., *Handbook of model checking*. Springer, 2018, vol. 10.

[16] M. Althoff and J. M. Dolan, ''Online verification of automated road vehicles using reachability analysis,'' *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[17] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, ''Verification of image-based neural network controllers using generative models,'' *Journal of Aerospace Information Systems*, vol. 19, no. 9, pp. 574–584, 2022.

[18] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M. J. Kochenderfer, et al., ''Algorithms for verifying deep neural networks,'' *Foundations and Trends in Optimization*, vol. 4, no. 3-4, pp. 244–404, 2021.

[19] H. Araujo, M. R. Mousavi, and M. Varshosaz, ''Testing, validation, and verification of robotic and autonomous systems: A systematic review,'' *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 2, pp. 1–61, 2023.

[20] S. A. Seshia, D. Sadigh, and S. S. Sastry, ''Toward verified artificial intelligence,'' *Communications of the ACM*, vol. 65, no. 7, pp. 46–55, 2022.

[21] V. Chandola, A. Banerjee, and V. Kumar, ''Anomaly detection: A survey,'' *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[22] J. Yang, K. Zhou, Y. Li, and Z. Liu, ''Generalized out-of-distribution detection: A survey,'' *International Journal of Computer Vision*, vol. 132, no. 12, pp. 5635–5662, 2024.

[23] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, ''Scalable end-to-end autonomous vehicle testing via rare-event simulation,'' *Advances in Neural Information Processing Systems (NIPS)*, vol. 31, 2018.

[24] D. Yeh, ''Autonomous systems and the challenges in verification, validation, and test,'' *IEEE Design & Test*, vol. 35, no. 3, pp. 89–97, 2018.

[25] A. Sinha, M. O'Kelly, R. Tedrake, and J. C. Duchi, ''Neural bridge sampling for evaluating safety-critical autonomous systems,'' in *Advances in Neural Information Processing Systems (NIPS)*, vol. 33, 2020.

[26] R. Lee, O. J. Mengshoel, A. Saksena, R. W. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer, ''Adaptive stress testing: Finding likely failure events with reinforcement learning,'' *Journal of Artificial Intelligence Research*, vol. 69, pp. 1165–1201, 2020.

[27] T. L. Arel, ''Safety management system manual,'' Air Traffic Organization, Federal Aviation Administration, Manual, 2022, Version 5.1.

[28] A. Donzé and O. Maler, ''Robust satisfaction of temporal logic over real-valued signals,'' in *International Conference on Formal Modeling and Analysis of Timed Systems*, 2010.

[29] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 1999, vol. 2.

[30] Y. Kim and M. J. Kochenderfer, ''Improving aircraft collision risk estimation using the cross-entropy method,'' *Journal of Air Transportation*, vol. 24, no. 2, pp. 55–62, 2016.

[31] H. Kahn and T. E. Harris, ''Estimation of particle transmission by random sampling,'' *National Bureau of Standards applied mathematics series*, vol. 12, pp. 27–30, 1951.

[32] F. Cérou and A. Guyader, ''Adaptive multilevel splitting for rare event analysis,'' *Stochastic Analysis and Applications*, vol. 25, no. 2, pp. 417–443, 2007.

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

[34] C. Innes and S. Ramamoorthy, ''Adaptive splitting of reusable temporal monitors for rare traffic violations,'' in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[35] D. Jacquemart and J. Morio, ''Conflict probability estimation between aircraft with dynamic importance splitting,'' *Safety science*, vol. 51, no. 1, pp. 94–100, 2013.

[36] S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar, ''A statistical approach to assessing neural network robustness,'' *arXiv preprint arXiv:1811.07209*, 2018.

[37] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte Carlo simulation, and machine learning*. Springer, 2004, vol. 133.

[38] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, ''A tutorial on the cross-entropy method,'' *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[39] D. Zhao, H. Peng, H. Lam, S. Bao, K. Nobukawa, D. J. LeBlanc, and C. S. Pan, ''Accelerated evaluation of automated vehicles in lane change scenarios,'' in *Dynamic Systems and Control Conference*, vol. 57243, 2015.

[40] C. J. Watkins and P. Dayan, ''Q-learning,'' *Machine learning*, vol. 8, pp. 279–292, 1992.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, ''Human-level control through deep reinforcement learning,'' *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[42] A. Corso, R. Lee, and M. J. Kochenderfer, ''Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition,'' in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[43] J. P. Chryssanthacopoulos, M. J. Kochenderfer, and R. E. Williams, ''Improved Monte Carlo sampling for conflict probability estimation,'' in *AIAA Non-Deterministic Approaches Conference*, 2010.

[44] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz, et al., ''Self-driving cars: A survey,'' *Expert Systems with Applications*, vol. 165, p. 113 816, 2021.

[45] J. Wang, L. Zhang, Y. Huang, and J. Zhao, ''Safety of autonomous vehicles,'' *Journal of advanced transportation*, vol. 2020, no. 1, p. 8 867 757, 2020.

[46] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi, and W. Cho, ''A review on autonomous vehicles: Progress, methods and challenges,'' *Electronics*, vol. 11, no. 14, p. 2162, 2022.

[47] M. Treiber, A. Hennecke, and D. Helbing, ''Congested traffic states in empirical observations and microscopic simulations,'' *Physical Review E*, vol. 62, p. 1805, 2000.

[48] P. Heidlauf, A. Collins, M. Bolender, and S. Bak, ''Verification challenges in F-16 ground collision avoidance and other automated maneuvers,'' in *ARCH@ ADHS*, 2018.

[49] B. Stevens, F. Lewis, and E. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. Wiley, 2015, ISBN: 9781118870983. [Online]. Available: https://books.google.com/books?id=lvhcCgAAQBAJ.

[50] D. Seto, E. Ferreira, and T. F. Marz, ''Case study: Development of a baseline controller for automatic landing of an f-16 aircraft using linear matrix inequalities (LMIs),'' *Software Engineering Institute, Carnegie Mellon University*, 2000.

[51] R. Neal, ''MCMC using Hamiltonian dynamics,'' in *Handbook of Markov Chain Monte Carlo*, S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, Eds., Chapman & Hall/CRC, 2011, ch. 5, pp. 113–117.

[52] M. D. Hoffman, A. Gelman, et al., ''The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo.,'' *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1593–1623, 2014.

[53] T. Akazaki, S. Liu, Y. Yamagata, Y. Duan, and J. Hao, ''Falsification of cyber-physical systems using deep reinforcement learning,'' in *International Symposium on Formal Methods (FM)*, 2018.

[54] C. E. Tuncali and G. Fainekos, ''Rapidly-exploring random trees for testing automated vehicles,'' in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2019.

[55] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc, ''Accelerated evaluation of automated vehicles in car-following maneuvers,'' *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 733–744, 2017.

[56] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, ''Stan: A probabilistic programming language,'' *Journal of statistical software*, vol. 76, pp. 1–32, 2017.

[57] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, ''Pyro: Deep universal probabilistic programming,'' *Journal of machine learning research*, vol. 20, no. 28, pp. 1–6, 2019.

[58] H. Ge, K. Xu, and Z. Ghahramani, ''Turing: A language for flexible probabilistic inference,'' in *International Conference on Artificial Intelligence and Statistics*, 2018.

[59] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani, ''Probabilistic programming,'' in *International Conference on Software Engineering (ICSE)*, 2014.

[60] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, ''Hybrid Monte Carlo,'' *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[61] M. Girolami and B. Calderhead, ''Riemann manifold Langevin and Hamiltonian Monte Carlo methods,'' *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, pp. 123–214, 2011.

[62] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt, ''A differentiable programming system to bridge machine learning and scientific computing,'' *arXiv preprint arXiv:1907.07587*, 2019.

[63] D. Rudoy and P. J. Wolfe, ''Monte Carlo methods for multi-modal distributions,'' in *Asilomar Conference on Signals, Systems and Computers*, 2006.

[64] J. M. Esposito, J. Kim, and V. Kumar, ''Adaptive RRTs for validating hybrid robotic control systems,'' in *Algorithmic Foundations of Robotics VI*, Springer, 2004, pp. 107–121.

[65] C. Andrieu, A. Doucet, and R. Holenstein, ''Particle Markov chain Monte Carlo methods,'' *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[66] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, ''Proximal policy optimization algorithms,'' *arXiv preprint arXiv:1707.06347*, 2017.

[67] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, ''Adaptive stress testing for autonomous vehicles,'' in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[68] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, ''Adaptive stress testing with reward augmentation for autonomous vehicle validation,'' in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2019.

[69] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, ''Variational inference: A review for statisticians,'' *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[70] Q. Liu and D. Wang, ''Stein variational gradient descent: A general purpose bayesian inference algorithm,'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.

[71] R. Lee, O. J. Mengshoel, A. K. Agogino, D. Giannakopoulou, and M. J. Kochenderfer, ''Adaptive stress testing of trajectory planning systems,'' 2019.

[72] L. Gibson, M. Hoerger, and D. Kroese, ''A flow-based generative model for rare-event simulation,'' *arXiv preprint arXiv:2305.07863*, 2023.

[73] J. Peltomäki and I. Porres, ''Requirement falsification for cyber-physical systems using generative models,'' *arXiv preprint arXiv:2310.20493*, 2023.

[74] J. Ho, A. Jain, and P. Abbeel, ''Denoising diffusion probabilistic models,'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[75] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, ''Score-based generative modeling through stochastic differential equations,'' in *International Conference on Learning Representations*, 2021.

[76] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, ''Planning with diffusion for flexible behavior synthesis,'' in *International Conference on Machine Learning (ICML)*, 2022.

[77] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, ''Motion planning diffusion: Learning and planning of robot motions with diffusion models,'' in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[78] Z. Zhang, Z. Zhao, J. Yu, and Q. Tian, ''Shiftddpms: Exploring conditional diffusion models by shifting diffusion trajectories,'' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023.

[79] B. Wang, G. Wu, T. Pang, Y. Zhang, and Y. Yin, ''Diffail: Diffusion adversarial imitation learning,'' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024.

[80] J. Song, C. Meng, and S. Ermon, ''Denoising diffusion implicit models,'' *arXiv preprint arXiv:2010.02502*, 2020.

[81] A. Q. Nichol and P. Dhariwal, ''Improved denoising diffusion probabilistic models,'' in *International Conference on Machine Learning (ICML)*, 2021.

[82] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, ''Reliable fidelity and diversity metrics for generative models,'' in *International Conference on Machine Learning (ICML)*, 2020.

[83] F. Cérou, P. Del Moral, T. Furon, and A. Guyader, ''Sequential Monte Carlo for rare event estimation,'' *Statistics and Computing*, vol. 22, no. 3, pp. 795–808, 2012.

[84] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer, 2004.

[85] 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 (NeurIPS)*, 2018.

[86] M. F. Bugallo, V. Elvira, L. Martino, D. Luengo, J. Miguez, and P. M. Djuric, ''Adaptive importance sampling: The past, the present, and the future,'' *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 60–79, 2017.

[87] O. Cappé, A. Guillin, J.-M. Marin, and C. P. Robert, ''Population Monte Carlo,'' *Journal of Computational and Graphical Statistics*, vol. 13, no. 4, pp. 907–929, 2004.

[88] J. Uesato, A. Kumar, C. Szepesvári, T. Erez, A. Ruderman, K. Anderson, K. Dvijotham, N. Heess, and P. Kohli, ''Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures,'' in *International Conference on Learning Representations*, 2019.

[89] A. Corso, K.-Y. Kim, S. Gupta, G. Gao, and M. J. Kochenderfer, ''A deep reinforcement learning approach to rare event estimation,'' *arXiv preprint arXiv:2211.12470*, 2022.

[90] C. Naesseth, F. Lindsten, and D. Blei, ''Markovian score climbing: Variational inference with kl(p || q),'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[91] K. Kim, J. Oh, J. Gardner, A. B. Dieng, and H. Kim, ''Markov chain score ascent: A unifying framework of variational inference with markovian gradients,'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022.

[92] W. K. Hastings, ''Monte Carlo sampling methods using Markov chains and their applications,'' *Biometrika*, vol. 57, pp. 97–109, 1970.

[93] K. J. Åström and K. Furuta, ''Swinging up a pendulum by energy control,'' *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[94] E. Smyers, S. Katz, A. Corso, and M. J. Kochenderfer, ''AVOIDDS: Aircraft vision-based intruder detection dataset and simulator,'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.

[95] K. D. Julian, R. Lee, and M. J. Kochenderfer, ''Validation of image-based neural network controllers through adaptive stress testing,'' in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[96] H. Delecki, M. Itkina, B. Lange, R. Senanayake, and M. J. Kochenderfer, ''How do we fail? stress testing perception in autonomous vehicles,'' in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[97] S. Luo and W. Hu, ''Diffusion probabilistic models for 3D point cloud generation,'' in *IEEE conference on computer vision and pattern recognition*, 2021.

[98] Y. Jin, Z. Sun, N. Li, K. Xu, H. Jiang, N. Zhuang, Q. Huang, Y. Song, Y. Mu, and Z. Lin, ''Pyramidal flow matching for efficient video generative modeling,'' *arXiv preprint arXiv:2410.05954*, 2024.

[99] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[100] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, ''Hindsight experience replay,'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[101] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, ''Mastering diverse domains through world models,'' *arXiv preprint arXiv:2301.04104*, 2023.

[102] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, ''Large language models are zero-shot reasoners,'' *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[103] A. Corso and M. J. Kochenderfer, ''Interpretable safety validation for autonomous vehicles,'' in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.