Finding Vulnerabilities of Autonomous Vehicle Stacks to Physical Adversaries

Final Report

by

Z. Berkay Celik
Purdue University

Satish V. Ukkusuri, Purdue University
Alvaro Cardenas, University of California, Santa Cruz
Daniel J. Fremont, University of California, Santa Cruz

May 2025

NATIONAL CENTER FOR TRANSPORTATION CYBERSECURITY AND RESILIENCY (TraCR)

DISCLAIMER

CONTACTS

For more information:

Z. Berkay Celik
LWSN 1203, 305 N. University Street
West Lafayette, IN 47907-2107, USA
Phone: 765-496-1761
Email: zcelik@purdue.edu

TraCR
Clemson University
One Research Dr
Greenville, SC 29607
tracr@clemson.edu

ACKNOWLEDGMENT

This work is based upon the work supported by the National Center for Transportation Cybersecurity and Resiliency (TraCR), a U.S. Department of Transportation National University Transportation Center headquartered at Clemson University, Clemson, South Carolina, USA. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of TraCR. The U.S. Government assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No. 5	**2. Government Accession No.** N/A	**3. Recipient's Catalog No.** N/A
4. Title and Subtitle Finding Vulnerabilities of Autonomous Vehicle Stacks to Physical Adversaries		**5. Report Date:** May 2025 **6. Performing Organization Code:** N/A
7. Author(s) Z. Berkay Celik, Ph.D.; https://orcid.org/0000-0001-7362-8905 Alvaro Cardenas, Ph.D.; https://orcid.org/0000-0002-5142-9750 Daniel J. Fremont, Ph.D.; https://orcid.org/0000-0002-9992-9965 Satish V. Ukkusuri, Ph.D.; https://orcid.org/0000-0001-8754-9925		**8. Performing Organization Report No.** 5
9. Performing Organization Name and Address National Center for Transportation Cybersecurity and Resiliency (TraCR), Clemson University, 414 A One Research Dr, Greenville, SC 29607 Purdue University, 610 Purdue Mall, West Lafayette, IN 47907 University of California, Santa Cruz, 1156 High St, Santa Cruz, CA 95064		**10. Work Unit No.** N/A **11. Contract or Grant No.** 69A3552344812 and 69A3552348317
12. Sponsoring Agency Name and Address U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology, 1200 New Jersey Avenue, SE, Washington, DC 20590		**13. Type of Report and Period Covered** Final Report, 01/01/2024 - 12/31/2024 **14. Sponsoring Agency Code** OST-R

15. Supplementary Notes
Conducted under the U.S. DOT Office of the Assistant Secretary for Research and Technology's (OST-R) University Transportation Centers (UTC) program.

16. Abstract

Autonomous Driving (AD) vehicles must interact and respond in real-time to multiple sensor signals indicating the behavior of other agents in the environment, such as other vehicles, and pedestrians near the ego vehicle (i.e., the vehicle itself). While autonomous vehicle (AV) developers tend to generate numerous test cases in simulations to detect safety and security problems, to the best of our knowledge, they are not testing for malicious physical interactions from attackers, such as by placing emergency cones in the hood of an AV or driving maneuvers that nearby human drivers or other AVs might perform.

The main goal of our project is to develop automatic testing tools to evaluate the safety and security of autonomous vehicle stacks against unanticipated critical physical conditions created by attackers. Specifically, we aim to demonstrate adversarial driving maneuvers in different real-world scenarios, highlighting the potential consequences for AV safety and security, build an attack framework in a simulation environment to study the optimal discovery of adversarial driving maneuvers, and contribute to the development of a skilled AV security workforce. In this way, this effort aims to enable the deployment of increasingly trustworthy transportation systems.

17. Keywords Autonomous Driving, Autonomous Vehicles, AV Security, Adversarial Attacks, Simulation, Automated Testing, AV Safety	**18. Distribution Statement** No restrictions.

19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 21	**22. Price** N/A

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

List of Tables

List of Figures

EXECUTIVE SUMMARY

Autonomous Driving (AD) vehicles are designed to navigate and interact with their environment through the analysis of real-time sensor data. However, this reliance on sensor input introduces a potential vulnerability: adversarial physical attacks. These attacks, where malicious actors carefully craft maneuvers can induce AVs to operate unsafely, potentially leading to collisions or other dangerous situations. This research sought to expose and mitigate this critical vulnerability by developing a suite of automatic testing tools. These tools are designed for the rigorous evaluation of the security and resilience of Autonomous Vehicle (AV) stacks when confronted with adversarial conditions.

The methodology of this project was structured around three key phases. First, the research involved the formalization of safety properties using Linear Temporal Logic (LTL). This process established precise definitions of safe AV operation, drawing upon established standards from the National Highway Traffic Safety Administration (NHTSA) and relevant driving regulations. To facilitate this formalization, Large Language Models (LLMs) were employed to extract and summarize safety, security and functional properties (the requirements a vehicle should satisfy to operate correctly), ensuring a systematic and comprehensive approach. Second, the project focused on the construction of a simulation framework capable of generating a wide array of adversarial scenarios. This framework, built upon the Scenic programming language, allowed for the precise parameterization of initial conditions, including the behavior of attacking vehicles and various environmental factors. A significant component of this phase was the integration of AV control software, such as OpenPilot and Autoware, with the CARLA simulator. Third, the research involved the development and application of search algorithms, specifically Generative Flow Networks (GFlowNets), to efficiently explore the generated scenarios. This exploration aimed to identify adversarial maneuvers capable of violating the defined safety properties, enabling the systematic discovery of vulnerabilities and the assessment of their potential consequences.

The key findings of this research are multifaceted. The study successfully demonstrated the feasibility and potential severity of adversarial driving maneuvers across a range of simulated, yet realistic, scenarios. Furthermore, the developed simulation framework, in conjunction with GFlowNets, proved to be effective in the systematic discovery of adversarial maneuvers capable of compromising AV safety. The research also identified and addressed challenges associated with integrating complex AV control software with simulation environments, an essential step towards achieving realistic testing conditions. Finally, the project yielded new techniques to improve the performance of GFlowNets within sparse reward environments, a common characteristic of adversarial scenario generation.

In conclusion, this research provides a valuable framework, along with a set of associated tools and methodologies, for the rigorous security testing of AVs. The findings of the project show the critical importance of explicitly considering adversarial physical attacks throughout the development lifecycle of autonomous vehicles. To further advance this field, future work would focus on several key areas. These include expanding the library of formalized safety properties to encompass a broader spectrum of driving scenarios and edge cases, developing more sophisticated

models of attacker behavior and capabilities to generate more realistic and challenging adversarial scenarios, and exploring the application of the developed testing methodologies to real-world AV systems within controlled testing environments.

CHAPTER 1
Background

1.1 Background and Motivation

Autonomous Driving (AD) vehicles have to interact and respond in real-time to multiple sensor signals indicating how other autonomous robots, targets, and the environment behave near the ego vehicle. While autonomous vehicle (AV) developers tend to generate numerous test cases in simulations to detect problems, to our best knowledge, they are not testing for malicious physical interactions from attackers, such as placing emergency cones in the hood of an AV or driving maneuvers that nearby human vehicle drivers or other AV manufacturers can create. For example, a hostile driving maneuver causing the victim vehicle to crash (while the malicious driver does not crash) can be identified by malicious actors, and then spread and reproduced by multiple people worldwide, causing traffic accidents on vehicles with vulnerable AD stacks.

Recently, TraCR members of UCSC and Purdue have introduced two different frameworks ([1], [2]) to explore the practicability of adversarial physical conditions in real-world environments. They focused on adversarial driving maneuvers, which are a new class of physical attack against AD software. Here, the attacker aims to find a (plausible) trajectory near the victim vehicle with the goal of causing it to behave in an unintended way, such as crashing or driving off the road.

The frameworks proposed by UCSC and Purdue differ in their assumptions about the attacker and the target AV software components. However, both provide an overview of the challenges, a means of discovering adversarial driving maneuvers in practice, and potential solutions to defend against them. While both frameworks have been shown, to some extent, to be effective in discovering adversarial driving maneuvers against a variety of AD software, the research on adversarial driving maneuvers is still in its early stages. In this proposal, we will study the weaknesses and strengths of both frameworks. Guided by our findings, we will explore creating a unified framework leveraging the best ideas from each university and explore rigorous measures of adversarial maneuvers for building a safe and secure AD software stack.

1.2 Project Objectives and Impacts

The main goal of our project is to develop automatic testing tools to evaluate the security of autonomous vehicle stacks against unanticipated critical conditions created by attackers. Specifically, we aim to (a) Demonstrate adversarial driving maneuvers in different real-world scenarios, highlighting the potential consequences to the security of AVs. (b) Build an attack framework in a simulation environment to study the optimal discovery of adversarial driving maneuvers. (c) Contribute to creating a workforce skilled in securing AVs and allow the deployment of increasingly trustworthy transportation systems.

We also inform policymakers of the risks posed by adversarial maneuvers by participating in public debates and government efforts, and providing technical talks for the target audience.

In the following we introduce three chapters to achieve these objectives. Table 1 illustrates the sequential dependency between Chapters 2, 3, and 4. Specifically, it outlines the objectives of each chapter and clarifies how they logically connect to establish the methodology for evaluating autonomous vehicle safety against adversarial attacks.

Table 1: Illustration of chapter objectives and their connection

Chapter	Title	Objectives	Connections
2	Formalizing Safety Properties	We defined the requirements for safe operation of autonomous vehicles (AVs) by formalizing safety properties. The work established a foundation using temporal logic and involved collecting data from sources like NHTSA standards and driving regulations. Large Language Models were used to aid in creating property summaries, which were then expressed in a formal logic. The formalized properties serve as a basis for subsequent tasks like vulnerability analysis.	Chapter 2 lays the groundwork by formalizing safety properties. These properties define what constitutes a "failure" or "unsafe" behavior of the autonomous vehicle. This provides the foundation for the subsequent chapters.
3	Parameterizing Initial Conditions	We detailed the process of building a language to parameterize initial conditions for testing AVs. The Scenic framework was used to generate adversarial scenarios, and a wrapper was developed to connect safety property propositions with scenario elements. The chapter also explored methods for spawning agents in relevant locations and generating initial conditions for interactions with other vehicles. Furthermore, the integration of control software, including VerifAI, Scenic, and OpenPilot, with the CARLA simulator, was discussed.	Chapter 3 directly depends on Chapter 2. It uses the safety properties defined in Chapter 2 to create specific, parameterized scenarios in a simulation environment. The scenarios are designed to test the AV's behavior against the defined safety properties.
4	Exploring Search Algorithms	We explored the use of Generative Flow Networks (GFlowNets) as a search method for adversarial driving maneuvers. GFlowNets were chosen for their ability to generate a diverse set of adversarial scenarios, unlike methods that focus solely on maximizing rewards. The chapter also discusses the challenges encountered in training GFlowNet models, specifically addressing sparse rewards and numerical instability. To mitigate these challenges, the research proposed and evaluated several complementary methods.	Chapter 4 builds upon Chapter 3. It employs search algorithms to explore the scenarios generated in Chapter 3 and identify adversarial maneuvers. The goal is to find specific inputs or conditions that cause the AV to violate the safety properties defined in Chapter 2.

CHAPTER 2
Formalizing Safety Properties

2.1 Objective

The goal of this chapter is to formalize properties for Autonomous Vehicles (AVs) that define the functional requirements the Automated Driving (AD) stack must adhere to for safe operation.

A foundation was established by defining formal safety properties using temporal logic in our prior research [1, 2]. The process of collecting data from relevant sources to inform the generation of diverse safety properties included reviewing NHTSA standards, where the National Highway Traffic Safety Administration (NHTSA) safety standards for vehicles were adapted into formal properties, and gathering driving regulations used for licensing tests in various states. These resources were analyzed to extract essential safe driving behaviors.

To aid in this process, Large Language Models (LLMs) such as Gemini or ChatGPT were utilized to create summaries of properties from the given resources written in natural language. These summaries were then used to formally define the properties using propositional variables, logical operators, and temporal modal operators. Preliminary results indicated that the summaries and properties were sound, although expressing them in temporal logic presented challenges, particularly for complex properties.

The analysis and formal representation of a substantial number of safety properties in temporal logic has been completed. These formalized properties serve as a foundation for subsequent tasks in the project, such as vulnerability analysis and the development of mitigation strategies.

2.2 Example of Safety Properties

We provide two examples of formal properties. These examples are expressed in Linear Temporal Logic (LTL) and are illustrative, designed to be adapted to specific NHTSA standards or driving regulations relevant to your project.

First, we consider the **Lane Keeping** property:

G (vehicle_in_lane -> F (vehicle_in_lane U (lane_change_initiated -> X G vehicle_in_lane))).

This property states that globally (always), if the vehicle is in its lane, then eventually it will remain in its lane *until* a lane change is initiated.

After a lane change is initiated, then in the next state and globally thereafter, the vehicle should be in its lane (the new lane). Here, G stands for Globally (always), F for Finally (eventually), U for Until, and X for Next. The propositional variables vehicle_in_lane and lane_change_initiated represent that the AV is within its designated lane and that a lane change maneuver has been initiated, respectively. This captures the requirement that lane changes should be deliberate and controlled, and the vehicle should remain in its lane otherwise.

In a second example, we consider the **Pedestrian Crossing Safety** property:

G (pedestrian_crossing_at_crosswalk -> F (vehicle_stopped_before_crosswalk)).

This property expresses that globally (always), if a pedestrian is crossing at a crosswalk, then eventually the vehicle must come to a complete stop before the crosswalk. The propositional variables pedestrian_crossing_at_crosswalk and vehicle_stopped_before_crosswalk represent that a pedestrian is detected crossing the street at a designated crosswalk and that the AV has come to a complete stop before the crosswalk, respectively. This property emphasizes the AV's responsibility to yield to pedestrians in crosswalks.

CHAPTER 3

Building a Language to Parameterize Initial Conditions

3.1 Generating Scenarios for Specific Properties

We build upon our recent work, Scenic [3], a general-purpose probabilistic programming language for describing driving scenarios. We use the Scenic framework to achieve the adversarial scenarios. Most challenges are addressed, and initialize scenarios are initialized with different numbers of agents in different environments.

We developed a wrapper on Scenic's language to connect safety property propositions with the physical objects and agents within a scenario. This allows for targeted scenario generation relevant to specific safety properties.

We additionally explored methods to:

- Spawn specific agents (e.g., pedestrians) at appropriate locations based on the safety property being tested. For instance, scenarios testing pedestrian safety will involve pedestrians in reasonable locations like sidewalks or crosswalks.

- Generate initial conditions where the victim vehicle interacts with other vehicles (e.g., following another vehicle for testing adaptive cruise control).

- Placing vehicles with advanced controller software stacks, such as Autoware or Apollo, in positions that require several decisions (e.g., close to an intersection or traffic light)

3.2 Adversarial Vehicle Parameterization

We extended Scenic to include parameters that generate adversarial vehicles of different sizes (sedan, SUV, truck) and with varied initial positions. This allowed the generation of diverse adversarial maneuvers that could potentially lead to safety property violations. We also pursued the integration of control software into the CARLA simulator.

Initially, we integrated VerifAI and Scenic with full-stack controllers. This integration enabled the use of VerifAI capabilities to test the decisions of advanced controllers under various conditions (e.g., vehicle positions) and their response to attackers. While Scenic can handle simple controllers, connecting it with advanced controllers presented challenges. For instance, VerifAI and Scenic use discrete-time simulation, whereas autonomous vehicle controllers operate in real-time. We successfully implemented a synchronized connection manager that interfaces the Scenic language, the VerifAI toolkit, the Carla simulator, and a full-stack Autonomous Vehicle driving software, Autoware. However, this implementation had some limitations. The simulation took longer than expected because we could not yet make Autoware execute faster than real-time. We made efforts to parallelize the simulation execution. We also encountered issues with correctly placing the vehicles and explored ways within the CARLA API to resolve this bug. Furthermore, we expanded the implementation to simplify integration with other full-stack controllers and develop a more general solution.

Secondly, we aimed to enable OpenPilot, a real-world autonomous driving (AD) software, to function within the CARLA simulator. This integration sought to facilitate experiments using customized traffic and environmental settings, providing a controlled and flexible testing environment for autonomous driving systems. This integration process presented several challenges. Firstly, OpenPilot and CARLA do not officially support this integration, necessitating custom solutions. Secondly, the system requires a substantial amount of data for training, demanding parallel execution capabilities. Additionally, high memory requirements during builds and potential software version conflicts posed significant hurdles.

To address these challenges, we implemented a custom bridge to connect OpenPilot with CARLA, enabling automatic communication between the two systems. Furthermore, we dockerized the implementation, ensuring its compatibility with any x86 architecture with Docker support. This approach significantly simplified deployment across different servers and machines. The result of this integration effort is highly beneficial for projects involving OpenPilot testing. It provides a foundation for parallel data collection, eliminating the need for time-consuming system setups on multiple machines, and allows for more efficient and scalable testing processes. Moreover, the dockerized solution enhances the reproducibility and portability of experiments, which is crucial for our research.

CHAPTER 4

Exploring Search Algorithms

4.1 Overview

In this chapter, we detail on exploring the potential of using a method called generative flow network (GFlowNet) [4] to search for adversarial driving maneuvers. Unlike reinforcement learning (RL) and similar methods that focus on maximizing cumulative rewards (i.e., seeking the most severe scenarios), GFlowNet aligned outcomes with a predefined reward distribution. This alignment offered the advantage of generating diverse types of adversarial scenarios.

We developed a program to train GFlowNet and RL models for generating adversarial driving maneuvers across three simulation environments: (1) a single-lane toy simulation with two victim vehicles controlled by a random parameterized intelligent driver model (IDM); (2) a gym highway environment with one victim vehicle controlled by reinforcement learning and five other victim vehicles controlled by a default IDM; and (3) the CARLA & OpenPilot simulation with one victim vehicle controlled by OpenPilot and the remaining vehicles controlled by CARLA autopilot. Figure 1 demonstrates visualization examples of the gym highway environment and the CARLA & OpenPilot environment.

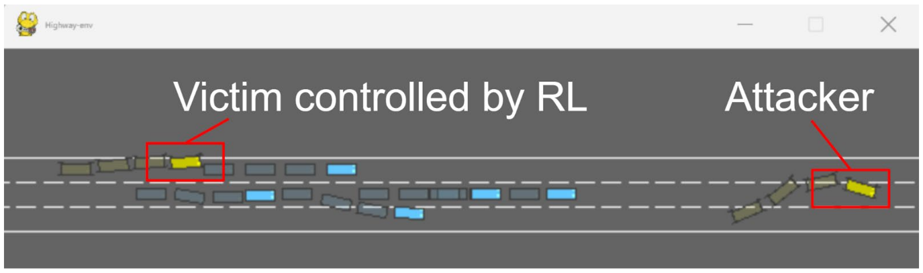

Figure 1: Illustration of highway environment in our experiments

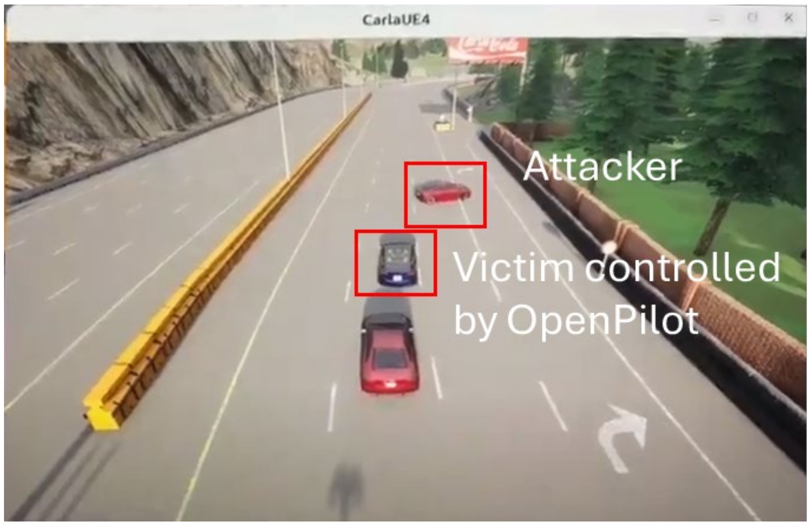

Figure 2: Illustration of the OpenPilot control software in the CARLA simulator

Our initial experiments revealed two challenges in effectively training GFlowNet models. Firstly, the scarcity of rewards in our problem made it challenging for the model to find useful samples for weight updates. Secondly, GFlowNet training was susceptible to numerical issues due to its use of logarithms in loss calculations. For trajectories with very small rewards, the logarithm of the reward could lead to excessively large values that distort gradient updates, causing the model to prioritize fitting unimportant low-reward observations.

We addressed these challenges. Specifically, we used a pre-trained RL model as the initial policy for GFlowNet, which ensured the coverage of high-reward regions. To address the second challenge, we filtered out samples with extremely low rewards and increased the sampling rate of high-reward trajectories when calculating the loss. These adjustments mitigated numerical instability and compensated for the larger gradients typically computed from low-reward trajectories.

4.1 Sparse Reward-Adaptive Generative Flow Networks

Based on our initial experiments with the vanilla GFlowNets, we addressed the challenge of improving Generative Flow Networks (GFlowNets) in environments where rewards are sparse. GFlowNets are a new class of algorithms that train policies to efficiently sample objects according to a specified reward distribution. However, their performance in sparse reward environments, such as those encountered in adversarial driving manuvers, often leads to suboptimal generative policies and incomplete learning of the target distribution. This limitation restricts their applicability to a wide range of problems where high-reward samples are valuable but sparse.

We identify three key challenges in training GFlowNets within sparse-reward environments:

- **Overfitting to Low-Reward Trajectories**: The GFlowNet may overfit to low-reward trajectories, hindering its ability to learn high-reward modes.

- **Missing High-Reward Trajectories**: High-reward trajectories may be missed during training due to their rarity.

- **High Variance in Loss Estimation**: The presence of the logarithm in the objective function can lead to high variance in loss estimation, especially in sparse reward scenarios where most trajectories have near-zero rewards.

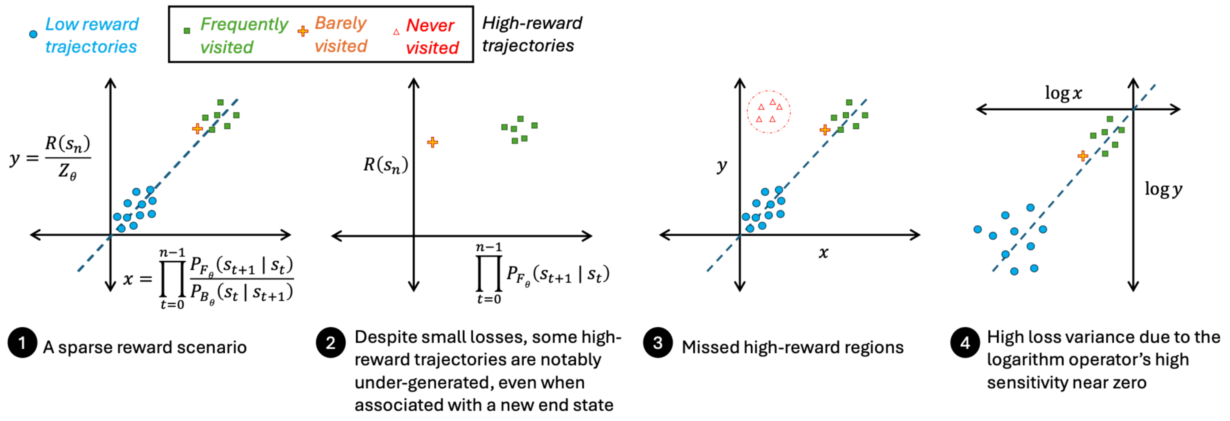
Low reward trajectories
Frequently visited
Barely visited
Never visited
High-reward trajectories
$y = \frac{R(s_n)}{Z_\theta}$
$x = \prod_{t=0}^{n-1} \frac{P_{F_\theta}(s_{t+1} \mid s_t)}{P_{B_\theta}(s_t \mid s_{t+1})}$
$R(s_n)$
$\prod_{t=0}^{n-1} P_{F_\theta}(s_{t+1} \mid s_t)$
y
x
$\log x$
$\log y$
1 A sparse reward scenario
2 Despite small losses, some high-reward trajectories are notably under-generated, even when associated with a new end state
3 Missed high-reward regions
4 High loss variance due to the logarithm operator's high sensitivity near zero

Figure 3: Illustration of the challenges caused by sparse rewards in GFlowNet training.

In Figure 3, we show the challenges caused by the sparse rewards Each point denotes a trajectory sampled during GFlowNet training. The dashed line denotes the optimal fitted line for minimizing the trajectory balance loss. The dashed circle marks the valid trajectories which have not been observed in training data.

To address these challenges, we proposed three complementary methods:

- **Batch Filtering (BF)**: This method identifies potentially overfit trajectories using batch-level statistics and modifies the associated loss to rectify it.

- **Sigmoid Temperature Decay (TD)**: Inspired by techniques in reinforcement learning and simulated annealing, this method introduces a temperature-based reward augmentation mechanism. This promotes exploration in the early stages of training and shifts toward exploitation as training progresses.

- **Mixed Priority (MP) Experience Replay Buffer**: This method defines the priority of a trajectory in the replay buffer by weighing the reward with the loss. This prioritizes high-loss trajectories while preventing excessive sampling of low-reward trajectories.

We conducted extensive empirical evaluations on a diverse suite of sparse-reward environments, including both discrete and continuous state and action spaces. The environments used are:

- **Hypergrid**: A discrete grid-like environment where the agent navigates to reach a high-reward state.
- **Gaussian Mixture**: A continuous environment with a reward function modeled as a Gaussian mixture.
- **Multi-objective Pusher**: A robotic arm environment where the goal is to move an object to one of two target positions.

The proposed solutions are compared against existing approaches, including Trajectory Balance (TB), Subtrajectory Balance (SubTB), and Generative Augmented Flow Networks (GAFN).

Key Findings:

- The proposed methods consistently learn effective policies and significantly outperform existing approaches across all tested environments.
- Batch Filtering effectively identifies and rectifies overfit trajectories.
- Sigmoid Temperature Decay encourages early exploration and stabilizes the loss function.
- The Mixed Priority Experience Replay Buffer prioritizes high-loss trajectories and prevents oversampling of low-reward trajectories.

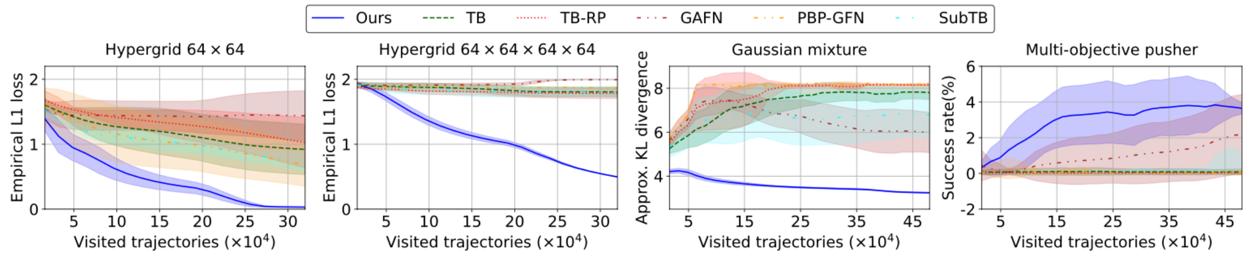

Figure 4: Performance comparison of our approach with baselines

Figure 4 shows the performance comparison of the proposed methods against several baseline methods in three simulation environments: Hypergrid, Gaussian Mixture, and Multi-objective Pusher. The baseline methods compared include:

- Trajectory balance (TB): This is a training objective for GFlowNets that has been shown to outperform earlier objectives like detailed balance and flow matching.

- Subtrajectory balance (SubTB): This is a more recent training objective that improves upon TB by enabling learning from partial action sequences of varying lengths.

- Trajectory balance with a prioritized replay buffer (TB-RP): This method enhances the sample efficiency of TB by prioritizing the replay of high-reward trajectories.

- Generative augmented flow networks (GAFN): This approach incorporates intrinsic rewards to provide more frequent feedback signals during training.

- Pessimistic backward policy GFlowNets (PBP-GFN): This method addresses an under-exploitation issue in GFlowNets by making the backward policy more pessimistic.

The x-axis of the figure represents the number of visited trajectories during training, and the y-axis represents different performance metrics for each environment. The performance metrics used are empirical L1 error for Hypergrid, approximate KL divergence for Gaussian Mixture, and success rate for Multi-objective Pusher. The proposed methods, labeled as "Ours", are shown to outperform the baseline methods in all three environments. The results demonstrate the effectiveness of the proposed methods in enhancing the sampling performance of GFlowNets in sparse reward scenarios.

Implications: Our work has provided valuable insights into the challenges of GFlowNet training in sparse reward environments and offers effective solutions to improve performance. The proposed methods have the potential to broaden their applicability to more complex and challenging tasks, such as robotic control, drug discovery, and adversarial example generation.

CHAPTER 5

Conclusions

This project contributed to the development of automatic testing tools designed to evaluate the security and safety of autonomous vehicle stacks against adversarial attacks. The formalization of safety properties, the creation of a language to parameterize initial conditions, and the exploration of search algorithms culminated in the establishment of a robust framework for identifying vulnerabilities and enhancing the safety of AVs. The successful incorporation of full-stack controllers into the CARLA simulator enabled more realistic and comprehensive testing, leading to the potential for more robust and secure AV deployments. Furthermore, our research on sparse reward-adaptive GFlowNets expanded the understanding of search algorithms, revealing potential applications beyond AV security.

While the project achieved its objectives, several avenues for future research emerged. Expanding the library of formalized safety properties to encompass a wider array of scenarios and edge cases remains a valuable direction. This expansion could draw upon resources such as NHTSA standards, driving regulations, and insights derived from LLMs. Another critical area involves the continued development of advanced methods for generating challenging adversarial scenarios. These methods should incorporate variations in environmental conditions, including weather and lighting, traffic density and complexity, and diverse interactions with various types of road users, such as pedestrians, cyclists, and other vehicles. Additionally, future research could model attacker behavior with increased sophistication, considering different levels of expertise, resources, and motivations. Further refinement of search algorithms, including the exploration of new model architectures, the incorporation of domain-specific knowledge, and the development of hybrid approaches, also presents a promising path. Finally, exploring real-world testing in controlled environments is essential to validate simulation results and comprehensively assess the performance of AV systems under real-world conditions. This could involve closed-track testing with human drivers simulating adversarial maneuvers or deploying instrumented vehicles in controlled environments with carefully designed scenarios.

REFERENCES

[1] Song, R., Ozmen, M.O., Kim, H., Muller, R., Celik, Z.B. and Bianchi, A., 2023. Discovering adversarial driving maneuvers against autonomous vehicles. In 32nd USENIX Security Symposium (USENIX Security 23) (pp. 2957-2974).

[2] Salgado, I.F., Quijano, N., Fremont, D.J. and Cardenas, A.A., 2022, June. Fuzzing malicious driving behavior to find vulnerabilities in collision avoidance systems. In 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (pp. 368-375). IEEE.

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

[4] Bengio, Y., Lahlou, S., Deleu, T., Hu, E.J., Tiwari, M. and Bengio, E., 2023. Gflownet foundations. The Journal of Machine Learning Research, 24(1), pp.10006-10060.