

DiscoFI: Discovering Safety-Critical Faults of Autonomous Driving Systems through Two-Phase Fault Injection

DAEHYUN LEE, George Mason University, USA

YU SUN, George Mason University, USA

YIYANG LU, William and Mary, USA

ANNA SCHMEDDING, William and Mary, USA

HOMA ALEMZADEH, University of Virginia, USA

EVGENIA SMIRNI, William and Mary, USA

LISHAN YANG, George Mason University, USA

Autonomous driving systems increasingly rely on large machine-learning models for perception, prediction, planning, and decision making. While dense chip design and shrinking transistors of modern computing hardware enables faster computation, these systems are increasingly susceptible to transient hardware faults, i.e., soft errors. Discovering faults that can lead to safety-critical driving behavior is therefore essential for enhancing safety but challenging because of the enormous fault site space and the high cost of validating fault effects through closed-loop simulation.

In this paper, we present DiscoFI, a two-phase Fault-Injection framework for efficiently Discovering safety-critical faults in autonomous driving systems. DiscoFI is motivated by a key observation: average displacement error (ADE) measured during open-loop fault injection provides an effective signal for prioritizing fault sites that are more likely to cause closed-loop failures. In Phase-I, DiscoFI progressively selects fault sites likely to produce high ADE through importance-based weight selection, component-specific bit selection, and bit-flip-direction selection. Phase-II validates the selected fault sites through closed-loop simulation. We implement DiscoFI in NVIDIA Alpmayo and use AlpaSim for closed-loop evaluation. Experimental results show that DiscoFI discovers, on average, 10.5× and 6.8× as many safety-critical faults as Random and TGFI baselines, respectively. An in-depth characterization study further reveals component- and layer-specific fault patterns and practical opportunities for targeted protection in VLA-enabled autonomous driving systems.

CCS Concepts: • **Computer systems organization** → **Reliability; Robotic autonomy**; • **General and reference** → **Measurement; Experimentation**; • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: Safety; Resilience; Vision-Language-Action Model; Autonomous Driving Systems; Fault Injection

ACM Reference Format:

Daehyun Lee, Yu Sun, Yiyang Lu, Anna Schmedding, Homa Alemzadeh, Evgenia Smirni, and Lishan Yang. 2026. DiscoFI: Discovering Safety-Critical Faults of Autonomous Driving Systems through Two-Phase Fault Injection. *Proc. ACM Meas. Anal. Comput. Syst.* 10, 3, Article 62 (December 2026), 25 pages. <https://doi.org/10.1145/3857365>

Authors' Contact Information: Daehyun Lee, George Mason University, Fairfax, VA, USA, dlee217@gmu.edu; Yu Sun, George Mason University, Fairfax, VA, USA, ysun23@gmu.edu; Yiyang Lu, William and Mary, Williamsburg, VA, USA, ylu21@wm.edu; Anna Schmedding, William and Mary, Williamsburg, VA, USA, akschmedding@wm.edu; Homa Alemzadeh, University of Virginia, Charlottesville, VA, USA, ha4d@virginia.edu; Evgenia Smirni, William and Mary, Williamsburg, VA, USA, esmirni@cs.wm.edu; Lishan Yang, George Mason University, Fairfax, VA, USA, lyang28@gmu.edu.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2476-1249/2026/12-ART62

<https://doi.org/10.1145/3857365>

1 Introduction

Overview and motivation. Autonomous driving systems (ADSs) are rapidly advancing across commercial, industrial, and open-source ecosystems, as demonstrated by systems developed by Waymo, NVIDIA, and the Autoware community [1, 26, 42]. Modern ADSs increasingly rely on sophisticated machine-learning models for perception, prediction, planning, and decision making, with recent systems further incorporating vision-language-action models to support more complex driving reasoning [1, 26]. This growing reliance on large and computationally intensive models increases both the complexity of the underlying computing stack and the potential impact of hardware-induced errors on vehicle behavior.

Modern computing platforms are susceptible to transient hardware faults, or soft errors, which may corrupt intermediate computations without permanently damaging the hardware [25, 27, 41, 45]. For a safety-critical ADS, evaluating only aggregate fault rates or average model accuracy is insufficient. Even when most faults are masked or benign, a small subset may propagate through the driving pipeline and trigger collisions or other hazardous vehicle behavior [34]. The central problem is therefore not merely to estimate overall ADS reliability, but to efficiently discover the rare fault instances that can lead to safety-critical outcomes. This is essential for strategic placement of protection mechanisms and cost-efficient fault tolerance [13, 14, 34, 48].

Challenges. First, the fault space within the VLA model is enormous, spanning components, layers, weight locations, and bit positions, with fault effects further varying across inputs and driving scenarios. Second, the safety impact of a fault cannot be determined from the model output alone because downstream planning, control, vehicle dynamics, and environmental feedback may either mask or amplify the error. A closed-loop evaluation captures the effects of the fault within an interaction space of the driving environment, while an open-loop evaluation focuses specifically on the output of the AI model while ignoring its effect within the environment pipeline. While there is a clear advantage of the closed-loop evaluation, its cost can be prohibitive. As shown in Table 1, a closed-loop run is over 200× slower, making exhaustive closed-loop fault injection impractical. The focus of this work is to address this challenge, i.e., maximize the discovery of safety-critical faults under a limited closed-loop simulation budget.

Limitations of the state-of-the-art. Open-loop studies concentrate on the resilience of the AI models used as components in autonomous driving [33]. Closed-loop fault injection captures the interactions among the driving pipeline and the environment, but its high computational cost severely limits the number of faults that can be evaluated [14, 31, 34, 46]. Consequently, existing approaches lack an effective mechanism for allocating a limited closed-loop simulation budget to the fault sites most likely to become safety-critical.

Key Insights and Contributions. We present DiscoFI, a two-phase fault-injection framework for accelerating the discovery of safety-critical faults in autonomous driving systems. To our best knowledge, DiscoFI is the first to connect scalable open-loop fault effects with closed-loop safety failures caused by transient hardware faults. DiscoFI is motivated by *a key observation*: faults that produce larger average displacement error (ADE) during open-loop model execution are more likely to trigger collisions or other failures during closed-loop simulation. This relationship enables ADE to serve as an effective proxy for prioritizing potentially safety-critical fault sites, although ADE alone cannot determine their final safety impact. In Phase-I, DiscoFI performs scalable open-loop fault injection and progressively selects fault sites that are likely to produce high ADE, thereby narrowing the candidate set requiring expensive end-to-end evaluation. In Phase-II, DiscoFI injects the selected faults during closed-loop simulation and validates whether they lead to collisions or other safety-critical outcomes.

Table 1. Average execution time of different simulation and hardware settings.

Experiment Type	GPU Type	Execution Time Per Run
Closed-loop simulation	NVIDIA GeForce RTX 3090Ti (for VLA model) NVIDIA A4000 (for AlpaSim simulator)	7.1 min
Open-loop VLA inference	3090Ti	1.8 sec
	NVIDIA A100	1.3 sec

We implement DiscoFI and integrate it with NVIDIA Alpamayo, a recent autonomous driving system built around a vision-language-action model, and use AlpaSim for end-to-end closed-loop simulation. We evaluate DiscoFI across multiple driving scenarios and compare it with two baselines. Furthermore, we perform a characterization study in Alpamayo to understand the distribution and manifestations of safety-critical faults. Our study consists of 580.6 hours of open-loop execution on a cluster and 1,481.1 hours of closed-loop simulation on a dedicated server. Our contributions are summarized as follows:

- We develop DiscoFI, a *two-phase* fault-injection framework that combines progressive open-loop fault-site selection with closed-loop safety validation to improve the efficiency of discovering safety-critical faults. To the best of our knowledge, this is the first two-phase approach for accelerating the discovery of critical faults in autonomous driving systems.
- Our characterization study reveals that fault-injection trials with higher open-loop ADE are substantially more likely to cause failures during closed-loop execution. We therefore use the ADE provided by the open-loop analysis as a proxy to prioritize fault sites for closed-loop validation.
- To efficiently identify fault sites that produce high ADE during open-loop execution, we *progressively* select candidate fault sites through three stages: weight selection, bit selection, and flip-direction selection.
- We implement DiscoFI in NVIDIA Alpamayo and evaluate it using closed-loop AlpaSim, demonstrating that DiscoFI discovers an average of 10.5 $\times$ and 6.8 $\times$ as many safety-critical faults as the two evaluated baselines, Random and TGFI [34].
- We further use DiscoFI to characterize safety-critical faults in Alpamayo to reveal the spatial distributions, failure manifestations, and closed-loop behavior of safety-critical faults, and derive practical implications for protecting VLA-enabled autonomous driving systems.

Artifact Availability. DiscoFI is publicly released on GitHub¹. Additionally, we release the whole DiscoFI framework and its associated experimental data on Zenodo².

2 Background

This section presents the system and reliability background necessary for the rest of the paper. We first introduce the studied autonomous-driving system and its associated simulation platform, and then describe the fault model and fault-injection methodology used in our experiments.

2.1 Autonomous Driving System and Simulation Platform

Modern autonomous driving systems integrate perception, reasoning, planning, and control into an end-to-end software pipeline, with ML models increasingly supporting perception, trajectory prediction, and decision making. In this work, we study NVIDIA Alpamayo, an industry-developed autonomous driving system equipped with a vision-language-action (VLA) model, together with its

¹<https://github.com/dhyunlee/DiscoFI>

²<https://doi.org/10.5281/zenodo.23348343>

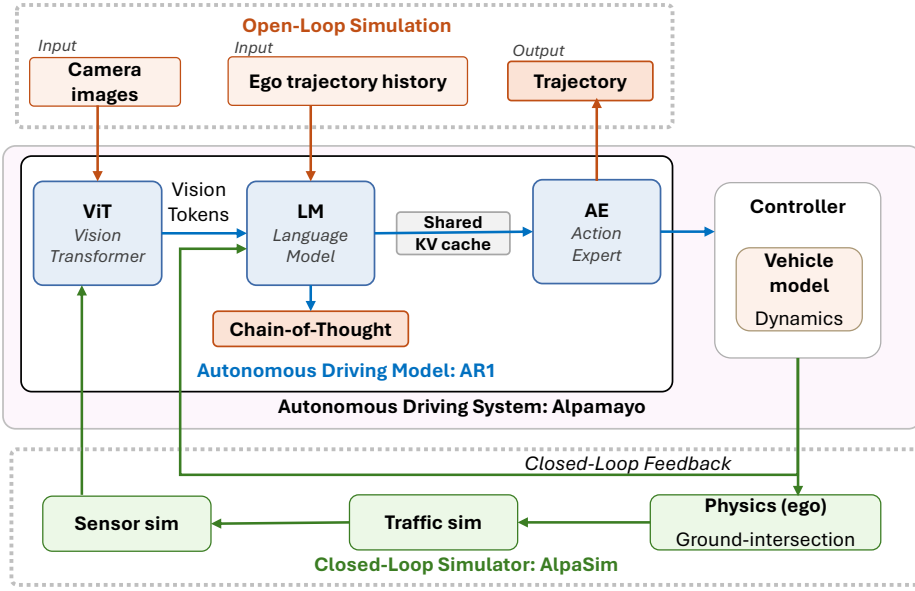


Fig. 1. Architecture of the Alpamayo autonomous driving system and its integration with open-loop and closed-loop simulation. In open-loop execution, AR1 processes camera images and ego-trajectory history to generate a future trajectory without environmental feedback. In closed-loop simulation, the generated trajectory is applied through the controller and vehicle model, and the resulting vehicle motion affects subsequent physics, traffic, and sensor observations.

associated AlpaSim simulation platform [26]. The overall architecture and the relationship between open-loop execution and closed-loop simulation are illustrated in Figure 1.

The core of Alpamayo is the open-source VLA model Alpamayo-R1 (AR1), which consists of three major components: a vision transformer (ViT), a language model (LM), and an action expert (AE). The ViT encodes camera observations into visual tokens, while the LM performs driving-related reasoning using the visual representation and ego-trajectory history. The LM produces a textual chain-of-thought (CoT) output and a shared key-value (KV) cache, which is consumed by the AE to generate the future ego-vehicle trajectory. Table 2 summarizes the parameter sizes of these components and their corresponding fault-site spaces, showing that exhaustive fault injection is impractical, since the correspond to approximately 7,300 years of open-loop and 2.4 million years of closed-loop experiments.

The focus of the open-loop evaluation is on the output of the AI model *only*. During open-loop execution, recorded camera data and ego-trajectory history are processed as inputs. The generated trajectory is compared with the corresponding ground-truth trajectory using **average displacement error (ADE)**, which measures the average Euclidean distance between corresponding predicted and ground-truth waypoints. A larger ADE indicates a greater deviation from the expected trajectory.

AlpaSim provides closed-loop simulation for evaluating the end-to-end behavior of Alpamayo. In each simulation step, the trajectory generated by AR1 is passed to the controller and vehicle model, which update the ego vehicle's motion. The resulting vehicle state affects the physics, traffic, and sensor simulations, producing new observations for the next AR1 inference. Through this feedback

Table 2. The number of weights and fault sites of the main components in driving model AR1.

Component	Number of Weights/Activations	Number of Fault Sites
Vision Transformer (ViT)	5.76E+08 (Weights)	9.22E+09
Language Model (LM)	8.22E+09 (Weights)	1.32E+11
Action Expert (AE)	2.28E+09 (Weights)	3.65E+10
KV Cache	2.23E+08 (Activations)	3.56E+09

loop, each inference iteration influences the subsequent vehicle state, sensor observations, and model decisions, enabling end-to-end evaluation of system behavior over time.

2.2 Fault Model and Fault Injection

Fault injection (FI) is widely used to study how transient hardware faults affect ML inference. A fault-injection experiment emulates a fault occurring in a memory or computation path and observes its impact on the application output [5–7, 17, 21]. KV cache uses 0.445GBs memory only, which implies that the probability of a fault there is diminished. Comparing to the ML components that occupy 24.63GBs, non-ML components of the ADS stack only occupy 2.17GBs, thus also small. Therefore, we focus on the components that occupies the majority of the memory: ViT, LM, and AE. We acknowledge that both perspectives are interesting to study, but it is out of scope of this work. **Assumptions.** We focus on inference-time faults in the studied ML model, specifically Alpamayo AR1, because the model is deployed for driving tasks after training has been completed and its parameters are fixed during execution [11, 26]. We assume at most one fault occurs during each simulated driving task. This assumption is reasonable because each task lasts only a few minutes, making the probability of multiple independent transient faults within the same execution window very low. It is also consistent with the single-fault model commonly adopted in prior reliability studies [3, 10, 19, 34, 36–38, 43]. The limitation of fault model is further discussed in section 7.

Tool. Several fault-injection frameworks for ML models are publicly available, including TensorFI [7], PyTorchFI [23], and SNIFF [4]. Our tool extends PyTorch’s hook mechanism to inject faults directly into selected model weights, following an approach similar to PyTorchFI. For each single-bit fault injection, we identify the target weight using its model component, block, layer, tensor index, and bit position. Before each execution, the selected bit is flipped to emulate a transient memory fault. After the execution completes, the same bit is restored to its original fault-free value, ensuring that each subsequent trial begins from an identical model state.

Statistical Reporting and Reproducibility Controls. For all sampling-based experiments, error bars represent 95% confidence intervals. To ensure deterministic and reproducible execution, we fix all random seeds and decoding configurations, disable stochastic sampling, and use identical inputs and simulation settings across repeated runs. We further repeat fault-free executions and verify that both model outputs and simulator outcomes remain identical across runs, ensuring that any differences between fault-injected and fault-free executions are attributable to the injected fault.

Outcome categories. An open-loop fault-injection trial can produce two possible outcomes. 1) The model may complete execution and generate a trajectory. Following NVIDIA’s standard evaluation, we quantify the deviation of the generated trajectory from the ground truth using ADE. 2) *NaN-induced model crash*: the injected fault causes a Not-a-Number (NaN) exception, resulting in model execution crash.

For closed-loop fault injection, we classify each execution as either benign or resulting in a failure. We use *collision* exclusively for physical vehicle impacts and *crash* for software-process termination. Failure outcomes are categorized as follows:

- *At-fault collision*: The fault causes the ego vehicle to behave abnormally and collide with another vehicle or object, typically through a front or lateral impact.

- *Rear-end collision*: Another vehicle collides with the rear of the ego vehicle, often after a fault induces abnormal braking, deceleration, or stopping. We distinguish rear-end collisions from at-fault collisions because non-ego vehicles in AlpaSim follow prerecorded behaviors.
- *Off-road event*: The fault causes the ego vehicle to leave the designated drivable area.
- *Driving-system crash*: The injected fault produces an excessively large intermediate value, causing the AR1 driver process to terminate with a CUDA error. The driving system consequently produces no trajectory output.
- *NaN-induced simulation halt*: The AR1 driver process remains active, but the corrupted computation overflows to Inf or NaN (Not-a-Number). The driver then emits an invalid trajectory, preventing the closed-loop simulation pipeline from advancing.

Executions that complete without any of these outcomes are classified as benign. A fault site that produces any failure outcome during closed-loop simulation is classified as *safety-critical*. In a real vehicle, a driving-system crash or invalid trajectory would not necessarily lead to the same physical outcome observed in simulation. Depending on the deployed safety monitors, fallback controller, and recovery mechanisms, such events could be detected and rejected, trigger an emergency stop or minimum-risk maneuver, or, if not handled promptly, result in loss of control or a collision.

3 Methodology: DiscoFI Design

Our methodology is motivated by the long execution time shown in Table 1: on the evaluated hardware configurations, an open-loop execution runs over 200× faster than the end-to-end closed-loop simulation. Performing large-scale random fault injection campaigns to discover collision cases is time-consuming and inefficient. The proposed DiscoFI accelerates the Discovery of safety-critical faults through two-phase Fault Injection, where Phase-I performs quick exploration on the ML model to identify a candidate set of critical fault sites through open-loop fault injection, and Phase-II uses closed-loop simulation to validate the safety criticality. Our two-phase methodology is based on a *key insight*: faults that produce larger average displacement error (ADE) during open-loop model execution are more likely to trigger collisions during closed-loop simulation. This relationship makes ADE an effective proxy for prioritizing potentially safety-critical faults, see Sec. 3.1 for details. The workflow of DiscoFI is shown in Figure 2 and briefly described below:

- **Phase-I: Open-Loop Exploration of High-ADE Fault Sites (Sec. 3.2)**. To efficiently identify fault sites that produce high ADE during open-loop execution, we progressively select candidate fault sites through three stages: weight selection, bit selection, and flip-direction selection. We then conduct an open-loop fault-injection campaign on the selected sites and measure the ADE produced by each injection.
- **Phase-II: Closed-Loop Validation of Safety Criticality (Sec. 3.3)**. Phase-II validates the selected fault sites in descending order of open-loop ADE, allowing a limited closed-loop simulation budget to be spent first on the candidates most likely to be safety-critical. The campaign records non-benign outcomes, including collisions and system crashes, to determine the closed-loop safety impact of each evaluated fault site.

3.1 ADE: An Open-Loop Proxy for Closed-Loop Safety Criticality

In this section, we investigate whether inexpensive open-loop fault-injection results can be used to predict the safety criticality of faults under closed-loop execution. This relationship is central to DiscoFI, as it enables the framework to prioritize promising fault sites before performing costly closed-loop validation. As introduced in Sec. 2.1, ADE is the standard metric used in the open-loop evaluation of NVIDIA AlpaMayo [26], measuring the average distance between the waypoints predicted by the VLA model and the corresponding ground truth waypoints. ADE quantifies the

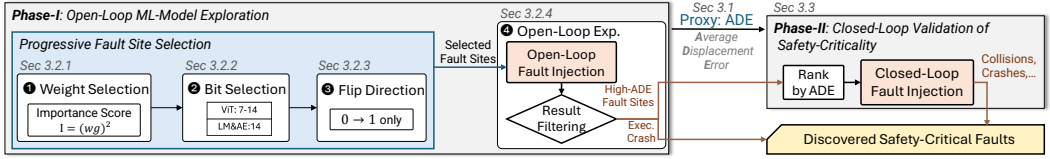


Fig. 2. Overview of DiscoFI. Phase-I progressively selects candidate fault sites through importance-based weight selection, component-specific bit selection, and bit-flip-direction selection, and evaluates them using open-loop fault injection and ADE-based filtering. Phase-II ranks the retained high-ADE sites by ADE and validates their safety criticality through closed-loop fault injection.

accuracy of the predicted trajectory, with a larger ADE indicating a greater deviation from the expected vehicle trajectory and, potentially, a higher likelihood of abnormal vehicle behavior.

To establish this correspondence, we conduct paired fault-injection experiments in open-loop ML inference and closed-loop end-to-end simulation. For each pair, we inject a fault at the same fault site in both settings, allowing the open-loop ADE to be directly associated with the corresponding closed-loop outcome. We sample fault sites using two complementary strategies. First, we uniformly sample across VLA components, weight locations, and bit positions to obtain an unbiased view of the fault space. Because random sampling alone produces too few high-ADE cases for reliable analysis, we additionally draw targeted samples using preliminary weight-importance and bit-sensitivity signals, which are formalized later in Phase-I. This enriched sampling provides sufficient coverage across both low- and high-ADE regions. Figure 3 groups these paired trials using progressively increasing open-loop ADE thresholds and reports the corresponding closed-loop failure rate. For each threshold, the failure rate is calculated as the fraction of trials with open-loop ADE above that threshold that result in a closed-loop failure. The results reveal a clear transition around an ADE of 1.0 m. When fault injection trials with an open-loop ADE below 1.0 m are excluded, the observed closed-loop failure rate increases sharply from around 25.7% to over 84.3%. Based on this observation, we classify fault sites into two groups:

- *High-ADE fault sites* with an open-loop ADE ≥ 1.0 m;
- *Low-ADE fault sites* with an open-loop ADE < 1.0 m.

The substantial difference in closed-loop failure rates between these two groups indicates that open-loop ADE provides an effective proxy for closed-loop safety criticality. Note that this 1.0 m threshold is empirical and DiscoFI does not depend on it. This directly motivates the goal of Phase-I: to efficiently identify high-ADE fault sites for subsequent closed-loop validation.

Observation #1: Fault-injection trials with higher open-loop ADE are substantially more likely to cause failures during closed-loop execution. This relationship supports using open-loop ADE as a proxy to prioritize fault sites for closed-loop validation.

3.2 Phase-I: Open-Loop Exploration of High-ADE Fault Sites

As motivated by Sec. 3.1, the purpose of Phase-I is to efficiently identify high-ADE fault sites to prepare Phase-II closed-loop validation. In this section, we progressively refine the fault site candidates through three stages: weight element selection (Sec. 3.2.1), bit position (Sec. 3.2.2), and binary bit direction (Sec. 3.2.3). Fault injection campaign is performed on these progressively selected fault sites to obtain their corresponding open-loop ADE.

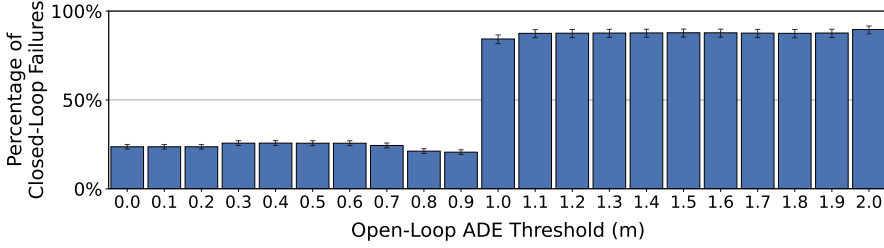


Fig. 3. Failure rate of closed-loop fault injection simulation under progressively increasing open-loop ADE thresholds. For each threshold, the failure rate is computed over fault-injection trials whose open-loop ADE exceeds that threshold. The sharp increase near 1.0 m supports using open-loop ADE as a proxy for closed-loop safety criticality.

3.2.1 Weight Selection. The first stage of Phase-I determines which model weights should be considered for fault injection. Uniform random sampling provides an unbiased baseline, but it treats all weights as equally likely to produce high-ADE outcomes. In practice, different weights contribute unequally to the model output, as widely observed in model pruning and importance-estimation studies [24, 47]. We therefore hypothesize that perturbing a weight with greater influence on the model prediction is more likely to cause a large deviation in the predicted trajectory.

Motivated by this observation, we investigate whether model-level weight importance can guide the discovery of high-ADE fault sites. We compare two weight-selection mechanisms:

- *Random selection* uniformly samples weights from each model component
- *Importance-based selection* prioritizes weights according to their estimated contribution to the model loss using an importance score.

To compute weight importance, similar to [34, 35] we adopt the first-order Taylor importance criterion proposed by Molchanov et al. [24]. This criterion approximates the change in loss caused by removing an individual weight. For a weight w_m , the first-order importance score is computed from the product of the weight and its corresponding trajectory-prediction loss gradient, enabling all weights to be ranked without evaluating them through separate perturbation experiments. The resulting importance score is given by

$$I_m^{(1)}(W) = (g_m w_m)^2, \quad (1)$$

where I is the resulting importance, W is the set of neural network parameters, g_m are elements of the gradient, and w_m are the weight values. Taylor-based importance estimation was originally developed for neural-network pruning and has also been shown to effectively identify fault-sensitive weights in CNNs [35].

We compute component-specific importance scores for the weights in the selected tensors and rank the weights within each component. We then compare random sampling with importance-based weight selection by measuring the percentage of sampled weights that produce high-ADE outcomes, as shown in Figure 4. For the random baseline, we uniformly sample 10,000 fault sites from each component. For importance-based selection, we vary K from 50 to 2,000 and perform fault injection across all 16 bits of the corresponding top- K ranked weights. Figure 4 shows that importance-based selection consistently achieves a higher high-ADE discovery rate than random sampling across all evaluated components and selection sizes. The benefit is most pronounced for ViT, where the top-ranked weights produce substantially more high-ADE outcomes than

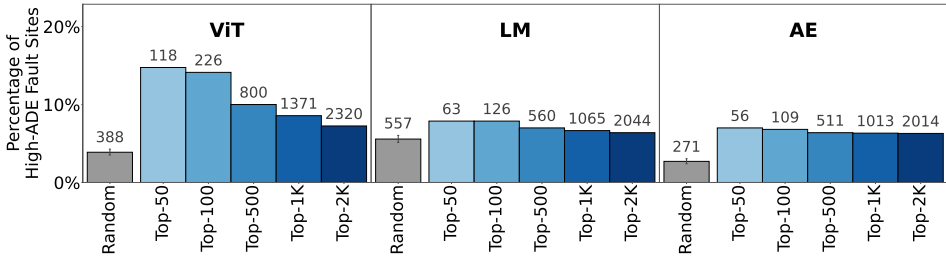


Fig. 4. Percentage of high-ADE fault sites identified by random sampling and importance-based weight selection across the ViT, LM, and AE components. Numbers above the bars indicate the corresponding number of open-loop high-ADE failures. Importance-based selection consistently achieves higher discovery rates than random sampling across all evaluated selection sizes.

random sampling. LM and AE exhibit smaller but still consistent improvements. Across all three components, the discovery rate generally decreases as K increases, indicating that high-ADE faults are concentrated among the most important weights. These results show that the importance score provides an effective signal for prioritizing weights in Phase-I. The higher top- K discovery rates for ViT reflect the effectiveness of importance-based selection rather than lower overall resilience. The random baseline provides a less biased comparison of component-level resilience, under which LM exhibits the highest high-ADE rate (least resilience), followed by ViT and AE.

Observation #2: Weight importance is an effective signal for guiding fault site selection. Compared with random sampling, importance-based selection achieves a higher high-ADE discovery rate with substantially fewer fault-injection trials.

3.2.2 Bit Selection. Prior studies have shown that faults at different floating-point bit positions can have substantially different effects on ML inference outcomes [9, 17, 37]. We therefore quantitatively characterize how faults at different bit positions contribute to high-ADE outcomes in a VLA-enabled ADS and use the resulting bit-level sensitivity profile to guide bit selection in Phase-I. Following the progressive fault-site selection process, this stage evaluates the top-2000 important weights identified during weight selection. The studied VLA model is trained by NVIDIA and executed in bfloat16 format, where bits 0–6 encode the mantissa, bits 7–14 encode the exponent, and bit 15 encodes the sign.

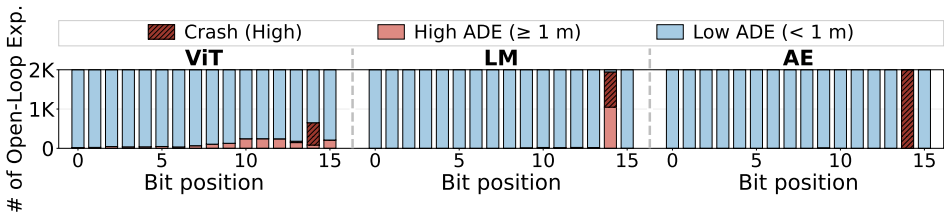


Fig. 5. Distribution of open-loop fault-injection outcomes across bit positions. High-ADE outcomes and crashes are broadly distributed across the exponent-bit region in ViT, whereas they occur primarily at bit 14 in LM and AE.

Figure 5 shows the distribution of open-loop fault-injection outcomes across bit positions. Across all three VLA components, bit 14 (the most significant bit of exponent) is the most critical bit position. However, the overall safety-criticality pattern differs substantially by component. ViT exhibits critical outcomes across multiple exponent bits, whereas harmful outcomes in LM and AE are concentrated primarily at bit 14. The outcome types also differ: faults in AE predominantly lead to NaN-induced model crash, while faults in ViT and LM can produce both application crashes and high-ADE results. We discuss the processing of these two outcome types in Sec. 3.2.4.

These results are consistent with prior observations that exponent bits are generally more vulnerable than mantissa bits [9, 17, 37]. Beyond this general trend, our analysis reveals component-specific critical-bit patterns in a VLA-enabled ADS, providing practical guidance for both fault-site selection and selective protection.

Observation #3: Exponent bits account for 96% of all observed high-ADE faults, but the specific critical exponent bits vary across model components. Phase-I therefore performs component-specific bit selection.

3.2.3 Selection of the Bit Flip Direction. The intuition behind flip-direction selection arises from the tolerance of neural networks to moderate numerical perturbations. Reductions in weight magnitude can often be masked by model redundancy and subsequent nonlinear operations, whereas an abnormally large weight can dominate downstream computations and substantially distort the model output. The two bit flip directions therefore have asymmetric effects: a $1 \rightarrow 0$ transition in an exponent bit typically reduces the weight magnitude, while a $0 \rightarrow 1$ transition may sharply increase the exponent and produce an extremely large corrupted weight.

We quantify how the bit flip direction affects the likelihood of producing high-ADE outcomes. Figure 6 compares the high-ADE discovery rates of the candidate fault sites retained after weight and bit selection, separated by bit-flip directions. Most of the high-ADE fault sites result from $0 \rightarrow 1$, although a small number of high-ADE faults caused by $1 \rightarrow 0$ flips are observed in ViT.

Observation #4: Most of the high-ADE fault sites result from $0 \rightarrow 1$ direction. Phase-I therefore selects $0 \rightarrow 1$ transitions during flip-direction selection.

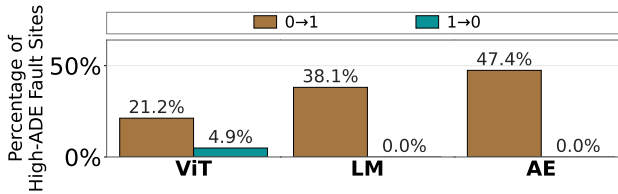


Fig. 6. High-ADE fault-site discovery rates for the two bit flip directions across the ViT, LM, and AE components. $0 \rightarrow 1$ consistently exhibits a much higher discovery rate.

3.2.4 Open-Loop Experiments and Outcome Processing. We conduct an open-loop fault-injection campaign on the fault sites identified through the three-stage progressive selection process. Each injection produces one of three outcomes: a high-ADE result, a low-ADE result, or a NaN-induced model crash. We retain fault sites that produce either high ADE or an application crash and discard those producing low ADE. As demonstrated in Sec. 3.1 and Figure 3, low-ADE faults have a

substantially lower likelihood of causing closed-loop failures. High-ADE fault sites are therefore forwarded to Phase-II for closed-loop safety validation.

NaN-induced model crashes (339 cases) are directly classified as failures rather than forwarding them to Phase-II for additional closed-loop validation. This treatment is supported by our paired experiments: all 339 NaN-induced cases produce non-benign closed-loop outcomes. All of these faults caused a system crash during closed-loop execution, except for one case that instead resulted in a rear-end collision.

3.3 Phase-II: Closed-Loop Validation of Safety Criticality

Phase-II validates the safety criticality of the high-ADE fault sites identified in Phase-I through fault injection in closed-loop autonomous-driving simulation. During each closed-loop injection, we monitor the simulation for non-benign outcomes, including vehicle collisions and system crashes. Fault sites that trigger such outcomes under the evaluated scenario are classified as safety-critical. Rather than evaluating these sites in an arbitrary order, we perform closed-loop validation in descending order of open-loop ADE, starting with the highest-ADE sites. As shown in Figure 3, fault sites with higher open-loop ADE exhibit a moderately higher closed-loop failure rate. Because closed-loop simulation is substantially more expensive than open-loop execution, the ADE-based ranking directs validation toward the candidates most likely to induce hazardous behavior. This ordering improves the efficiency of discovering safety-critical faults by evaluating the most promising candidates first.

3.3.1 Sensitivity of Phase-I Scenario selection. Both phases of DiscoFI operate on driving scenarios: Phase-I uses an open-loop scenario to identify high-ADE fault sites, while Phase-II evaluates the selected sites under a closed-loop scenario. This raises an important methodological question: Do high-ADE fault sites selected under different Phase-I scenarios remain similarly effective in closed-loop validation?

NVIDIA's open-loop execution environment provides 1,181 scenarios, while AlpaSim provides 920 closed-loop scenarios. Exhaustively evaluating the full scenario space is impractical because each additional Phase-I scenario requires a separate fault-injection campaign, consuming approximately 16,909 A100 GPU hours for the 1,181 scenarios. Moreover, the two environments do not provide identical scenario sets, making exact scenario matching between open-loop and closed-loop evaluation infeasible.

To study the sensitivity of Phase-I scenario selection, we carefully select seven representative scenarios for open-loop fault injection, including driving straight in traffic, turning left, turning right, stopping at a traffic signal, and yielding to pedestrians. For each scenario and VLA component, we execute Phase-I of DiscoFI using the same deterministic procedure. We begin with the top-2,000 importance-ranked weights profiled from each scenario and evaluate all 16 bit positions, yielding 32,000 initial fault sites per component. We then apply the bit-position and flip-direction selection criteria and rank the retained sites by their resulting open-loop ADE. This process produces one scenario-specific ADE-ranked fault list for each of the seven scenarios. For Phase-II, we select a straight-driving scenario in AlpaSim that is semantically similar to the Phase-I scenario S1-Open and evaluate all seven fault lists under this same closed-loop scenario. Figure 7 summarizes the selected scenarios.

Figure 8 compares the effectiveness of the fault lists generated from the seven Phase-I scenarios. Despite being derived from different open-loop driving conditions, the lists yield similar closed-loop failure rates under the same Phase-II scenario. In particular, fault lists generated from scenarios that are not semantically matched to the AlpaSim scenario perform comparably to the list generated from the matched straight-driving scenario. These results indicate that ADE-guided fault selection

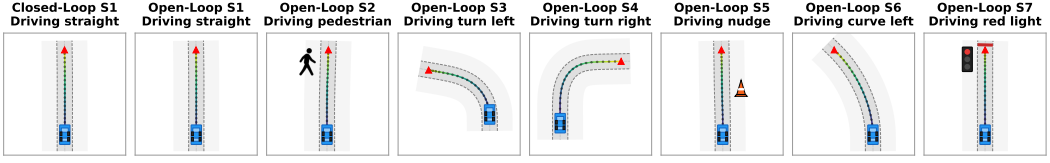


Fig. 7. Ground truth trajectory of studied scenarios.

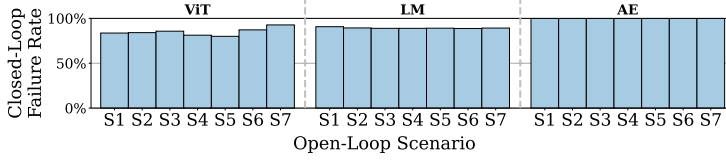


Fig. 8. Closed-loop failure rates of high-ADE fault sites selected under seven Phase-I open-loop scenarios, evaluated using the same Phase-II closed-loop scenario. The comparable rates across S1–S7 demonstrate the robustness of ADE-guided fault selection to Phase-I scenario choice.

is relatively insensitive to the choice of Phase-I scenario across the scenarios evaluated and does not require exact scenario matching between the two phases.

Observation #5: High-ADE fault sites selected under different Phase-I scenarios remain similarly effective in closed-loop validation. Therefore, Phase-I does not require an open-loop scenario that exactly matches the target Phase-II scenario.

4 Evaluation

In this section, we evaluate the effectiveness of DiscoFI. We first introduce the experimental setup and baselines, and then present the results in different settings.

4.1 Experimental setup and baseline.

All experiments are performed on two hardware configurations. The first one is a dedicated server with root access, consisting of an Intel Core i9-12900KF CPU, an NVIDIA GeForce RTX 3090 Ti GPU, and an NVIDIA RTX A4000 GPU, running Ubuntu 22.04.5 LTS as its operating system. The second configuration is a shared cluster without root access, consists of NVIDIA A100 GPUs, running Rocky Linux 8.10 as its operating system. The shared cluster does not support Docker required by the closed-loop AlpaSim due to lack of root access therefore is used for open-loop experiments only. Unless otherwise stated, Phase-I experiments are conducted on the A100 cluster, while Phase-II experiments are performed on the dedicated server equipped with RTX 3090 Ti (for executing AlpaMayo Autonomous Driving System) and A4000 (for running closed-loop AlpaSim simulator). The open-loop and closed-loop experiments required approximately 580.6 hours and 1,481.1 hours of execution time, respectively. We use the driving straight scenario (Closed Loop S1) shown in Figure 7 as the closed-loop testing scenario.

We consider two baselines: Random and TGFI [34]. Random fault injection uniformly samples from the entire fault-site space and serves as a simple, method-agnostic baseline. TGFI [34] is an importance-score-based fault injection technique for autonomous driving systems. It was evaluated on OpenPilot, a driver-assistance system requiring continuous driver supervision, and LBC, a research platform designed for fully autonomous driving tasks in controlled environments.

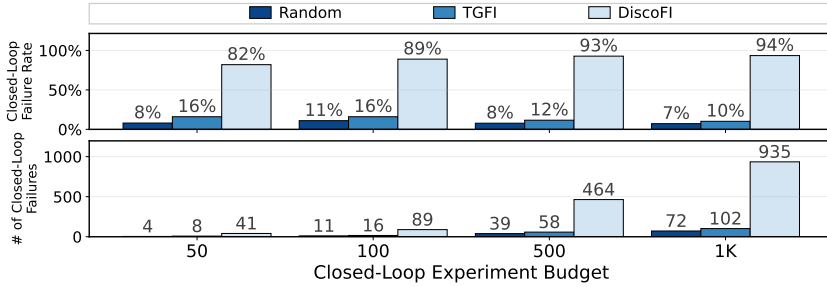


Fig. 9. Closed-loop failure rates and numbers of discovered safety-critical faults under different closed-loop experiment budgets. The top panel reports the failure rate, while the bottom panel reports the corresponding number of discovered safety-critical faults. DiscoFI consistently outperforms Random and TGFI across all evaluated budgets. The values above the bars report the corresponding failure rate (top) or the number of failures (bottom). The failure rate increases with closed-loop budget, indicating that increasing the budget allows more failure cases to be identified.

4.2 Effectiveness

We first evaluate the primary objective of DiscoFI: discovering as many safety-critical faults as possible under a limited closed-loop experiment budget, where the budget is defined as the maximum number of closed-loop fault-injection experiments allowed. We vary the number of closed-loop fault-injection experiments from 50 to 1,000 and compare DiscoFI with Random and TGFI [34]. For each budget, we report the closed-loop failure rate, i.e., the percentage of closed-loop fault-injection experiments that result in safety-critical failures. As shown in Figure 9, across budgets ranging from 50 to 1,000 closed-loop trials, DiscoFI discovers, on average, 10.5 $\times$ and 6.8 $\times$ as many safety-critical faults as Random and TGFI, respectively. These results demonstrate that DiscoFI remains effective across a wide range of closed-loop experiment budgets.

We further report the results of the full fault-injection campaigns conducted for each method. Unlike the budget-controlled comparison, these campaigns use different numbers of closed-loop trials because the methods produce candidate sets of different sizes. Random evaluates 3,000 uniformly sampled fault sites, with 1,000 sites from each VLA component. TGFI evaluates the top 50 importance-ranked weights across all 16 bit positions, resulting in 800 fault sites per component and 2,400 closed-loop trials in total. We include an additional baseline *Open-Loop Random, Closed-Loop High-ADE* where we perform 30,000 random open-loop fault injection trials and forwards all 142 resulting high-ADE fault sites to closed-loop validation. For DiscoFI, we validate the top 1,000 fault sites prioritized by Phase-I.

Because the closed-loop budgets differ, we use the closed-loop failure rate as the primary measure of fault-discovery efficiency and report the raw number of discovered safety-critical faults as a complementary measure. As shown in Figure 10, DiscoFI achieves the highest closed-loop failure rate and discovers the largest number of safety-critical faults overall and across the ViT, LM, and AE components. The Open-Loop Random+ADE baseline (the third bar of each group) isolates the benefit of using open-loop ADE as a proxy without progressive fault-site selection in Phase-I. Although ADE-based filtering has a relatively high closed-loop failure rate, this baseline discovers only a limited number of safety-critical faults because random open-loop exploration produces few high-ADE candidates. These results show that ADE effectively prioritizes candidates for closed-loop validation, while progressive selection is necessary to identify a sufficiently large set of high-ADE fault sites.

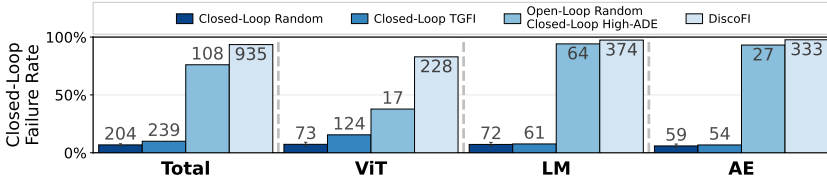


Fig. 10. Closed-loop failure rates achieved by baselines and DiscoFI. Numbers above the bars indicate the corresponding numbers of closed-loop failures. DiscoFI achieves both a high closed-loop failure rate and the largest number of discovered safety-critical faults.

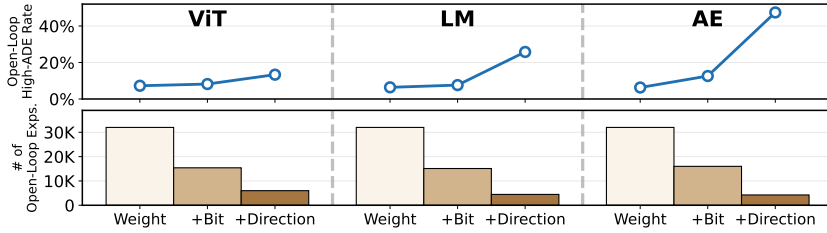


Fig. 11. Effectiveness of the three-stage progressive fault-site selection in Phase-I. The bottom panel reports the number of candidate fault sites after weight, bit-position, and flip-direction selection, while the top panel reports the corresponding discovery rate of open-loop high-ADE fault sites. Successive stages reduce the candidate set while increasing the concentration of high-ADE faults across ViT, LM, and AE.

We further evaluate the effectiveness of the progressive fault-site selection in Phase-I, as shown in Figure 11. The bottom panel reports the number of candidate fault sites retained after each selection stage, while the top panel reports the corresponding high-ADE discovery rate. Starting from importance-based weight selection, Phase-I progressively narrows the candidate set through bit-position selection and flip-direction selection. Across all three VLA components, each successive stage substantially reduces the number of candidate fault sites while increasing the percentage that produce high-ADE outcomes. In particular, the third stage, bit flip direction selecting 0→1 produces the smallest candidate set and the highest discovery rate. The simultaneous reduction in candidate-set size and increase in high-ADE discovery rate demonstrate the effectiveness of the three-stage progressive selection strategy.

4.3 Sensitivity to Closed-Loop Scenarios

In this section, we evaluate whether the high-ADE fault sites identified by DiscoFI remain effective across different closed-loop driving scenarios. Due to the high computational cost of closed-loop simulation, we conduct this experiment using a budget of 100 fault-injection trials per scenario.

For all evaluated closed-loop scenarios, we use the same set of high-ADE fault sites selected from the Phase-I S1-Open scenario (driving straight). This design isolates the impact of the Phase-II driving scenario, as our earlier analysis in Sec. 3.3.1 has already shown that fault sites selected from different Phase-I scenarios exhibit similar effectiveness under the same closed-loop scenario. In addition to the straight-driving scenario used in the preceding experiments, we evaluate a pedestrian-interaction scenario and a right-turn scenario. As shown in Figure 12, DiscoFI consistently achieves high closed-loop failure rates across all three scenarios and across the ViT, LM, and AE components. Although the exact failure rates vary by scenario and component, the results indicate that the

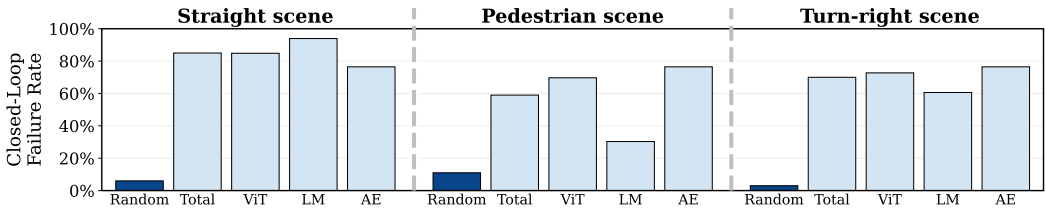


Fig. 12. Closed-loop failure rates of DiscoFI across different closed-loop scenarios under a budget of 100 closed-loop trials per scenario. The same high-ADE fault sites selected from the Phase-I S1-Open scenario are evaluated in all three closed-loop scenarios. DiscoFI consistently achieves higher closed-loop failure rates than the random baseline.

high-ADE fault sites identified by DiscoFI remain effective under diverse closed-loop driving conditions.

5 Characterizing Safety-Critical Faults with DiscoFI

Building on the preceding evaluation, we use the safety-critical fault sites identified by DiscoFI to characterize the resilience of NVIDIA Alpmayo [26]. The study addresses three questions:

- (Sec. 5.1) How are safety-critical fault sites distributed across the VLA model, and what failure modes do they induce?
- (Sec. 5.2) When do collision-producing faults manifest during closed-loop execution, and how do their trajectory effects differ from their open-loop manifestations?
- (Sec. 5.3) How consistently do chain-of-thought semantics reflect the resulting trajectory behavior and closed-loop safety outcome?

5.1 Distribution and Failure Manifestations of Safety-Critical Faults

Before presenting the per-layer results, we briefly summarize the relevant AR1 layer structures in Figure 13. The ViT uses LayerNorm, attention projection layers, and a two-layer MLP with `fc1` and `fc2`, whereas the LM and AE use RMSNorm and gated MLPs with `gate_proj`, `up_proj`, and `down_proj`. These layers correspond to the fault-injection targets analyzed below. We examine how the safety-critical fault sites identified by DiscoFI are distributed across the ViT, LM, and AE components, as well as how faults at these sites manifest during closed-loop execution.

Figure 14 reports the closed-loop failure rate for each evaluated layer, decomposed by failure type. The number above each bar denotes the raw number of closed-loop failures discovered in that layer. Although several layers exhibit similarly high failure rates, the resulting failure modes differ substantially across components and layers. In ViT, faults in the normalization and bias primarily cause at-fault or rear-end collisions. Faults in `mlp.fc1`, however, produce a broader mixture of outcomes, including driving-system crashes and NaN-induced simulation halts. The LM exhibits even stronger layer-level heterogeneity. Faults in `q_norm` and `k_norm` predominantly lead to rear-end collisions, whereas faults in `v_proj` result in driving-system crashes. The `o_proj` and `mlp.down_proj` layers produce mixtures of collisions and driving-system crashes. In contrast, all observed failures caused by AE fault sites are NaN-induced simulation halts. This result indicates that severe AE corruptions invalidate the generated trajectory rather than producing a finite but unsafe trajectory. It also suggests that finite-value and range checks at the AE output could provide a practical mechanism for detecting such failures before the invalid trajectory is propagated downstream.

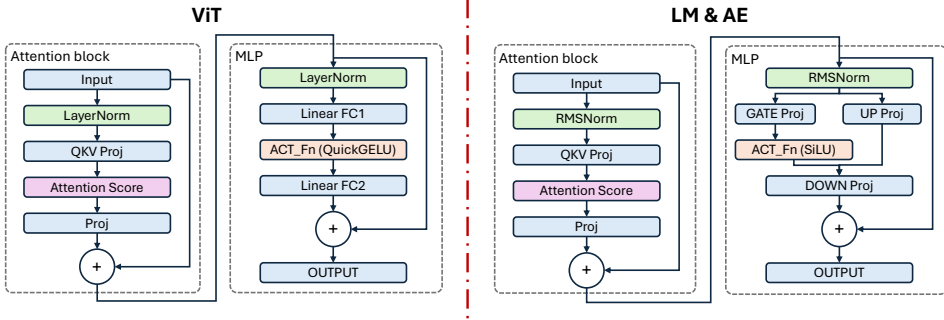


Fig. 13. Simplified layer structures of the AR1 model components. The ViT uses LayerNorm, attention projection layers, and a two-layer MLP with fc1 and fc2, whereas the LM and AE use RMSNorm and gated MLPs with gate_proj, up_proj, and down_proj. These layer names are referenced in the per-layer fault-vulnerability analysis that follows.

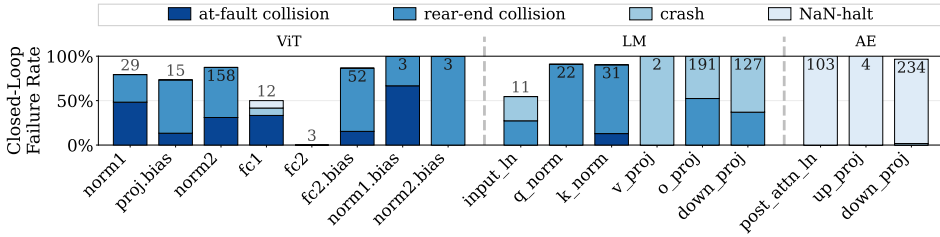


Fig. 14. Closed-loop failure rate by layer. The number on top of each bar denotes the raw number of closed-loop failures.

Figure 15 further localizes the selected and validated fault sites within their corresponding weight tensors. The yellow points denote the importance-guided candidate weight locations selected in Phase-I, while the blue points denote fault sites confirmed as safety-critical through closed-loop validation. The confirmed sites constitute only a sparse subset of the Phase-I candidates and are distributed non-uniformly within the tensors. The spatial patterns also vary across layers. Some tensors exhibit fault sites concentrated along particular rows, columns, or narrow bands, while others show more scattered distributions. For example, the safety-critical sites in ViT's mlp.fc1 are relatively scattered, whereas several LM and AE projection tensors contain concentrated regions or column-wise bands. In particular, the AE mlp.up_proj contains a narrow concentration of critical sites, while mlp.down_proj contains several distinct bands. These results show that safety-critical vulnerability is structured at the component, layer, and tensor-region levels. Selective protection can therefore focus on specific regions rather than uniformly protecting all model parameters.

Observation #6: Safety-critical fault sites are sparse and spatially structured, while their failure manifestations vary substantially across components and layers. ViT and LM faults produce collisions and driving-system crashes, whereas all observed AE faults result in NaN-induced simulation halts.

A NaN-induced halt prevents the closed-loop simulation from advancing. In a real vehicle, however, an invalid trajectory may lead to different outcomes depending on the available safety

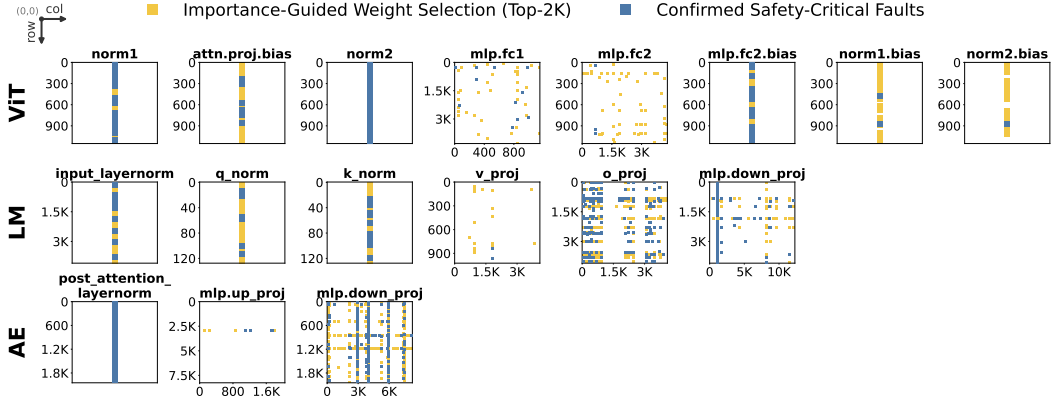


Fig. 15. Example patterns of Top-2000 important weights (yellow) and the safety-critical faults confirmed by closed-loop simulation (blue). For two-dimensional weight tensors, the x- and y-axes denote the column and row indices, respectively; for one-dimensional tensors, the y-axis denotes the element index.

mechanisms. It may be detected and rejected, triggering re-execution, a fallback controller, or a minimum-risk maneuver; if it is not detected or handled in time, it may instead result in unsafe vehicle behavior or a collision. Existing resilience techniques for transformer-based inference detect abnormal numerical states and trigger recovery through mechanisms such as recomputation or fallback execution [20, 38]. Similar mechanisms can be applied at the AE output interface by checking whether the generated trajectory contains Inf, NaN, or values outside physically valid ranges before it is consumed by downstream control components. Because these checks operate only on the generated trajectory rather than the full internal model state, they provide a potentially lightweight protection point. Upon detecting an invalid trajectory, the ADS can reject the output and initiate re-execution, use a fallback trajectory, or perform a minimum-risk maneuver.

Takeaway Message #1: AE protection should prioritize lightweight output-level numerical-validity checks, including Inf/NaN detection and trajectory-range validation, together with an appropriate recovery mechanism.

5.2 Temporal and Trajectory Manifestations of Collision-Producing Faults

We next examine the temporal and trajectory manifestations of collision-producing faults during closed-loop execution. Specifically, we analyze when collisions occur in the simulated driving scenario and how the resulting closed-loop trajectories differ from their open-loop counterparts.

Each AlpaSim run simulates approximately 20 seconds of vehicle motion and invokes the AR1 model 186 times. For each collision-producing trial, we measure the time from the beginning of the simulation to the first collision. Figure 16 shows the resulting distributions for faults injected into the ViT, LM, and AE components. To characterize collision timing rather than fault-site selection efficiency, we pool collision-producing trials from the importance-guided and random fault-injection campaigns.

The three components exhibit different temporal patterns. ViT-induced collisions occur over a relatively broad interval, with most appearing between approximately 5 and 15 seconds. LM-induced collisions generally occur later, with a strong concentration between approximately 10 and 15 seconds. AE produces substantially fewer collisions, consistent with the finding in subsection 5.1

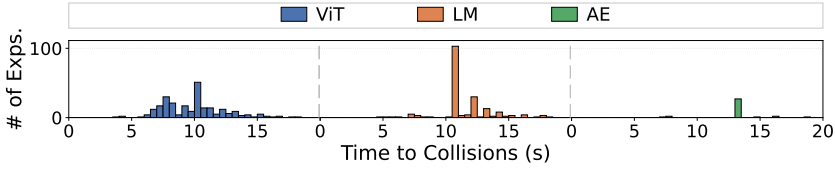


Fig. 16. Distribution of simulation time to the first collision for faults injected into the ViT, LM, and AE components. The histograms report the number of collision-producing trials as a function of the elapsed time within the simulated driving scenario, measured from the start of closed-loop simulation to the first collision.

that most AE failures instead manifest as NaN-induced simulation halts. The observed collision times for AE are therefore sparse and concentrated primarily in the later portion of the simulation. These results indicate that the temporal manifestation of a fault depends on the affected VLA component, although the precise collision time also depends on the driving scenario and vehicle interactions.

Observation #7: Collision-producing faults exhibit component-dependent temporal patterns: ViT collisions occur over a broad interval, LM collisions are concentrated later, and AE produces comparatively few collisions.

To examine how high-ADE faults evolve into collision-producing trajectory deviations under closed-loop feedback, Figure 17 presents two representative collision-producing cases from ViT and LM. Each example compares the fault-injected and fault-free trajectories during open-loop and closed-loop execution.

In the ViT example, the injected fault produces an open-loop ADE of 7.6 m, compared with 1.0 m for fault-free execution. The large open-loop ADE is primarily caused by a longitudinal deviation in the predicted trajectory. During closed-loop execution, however, the fault-injected trajectory has

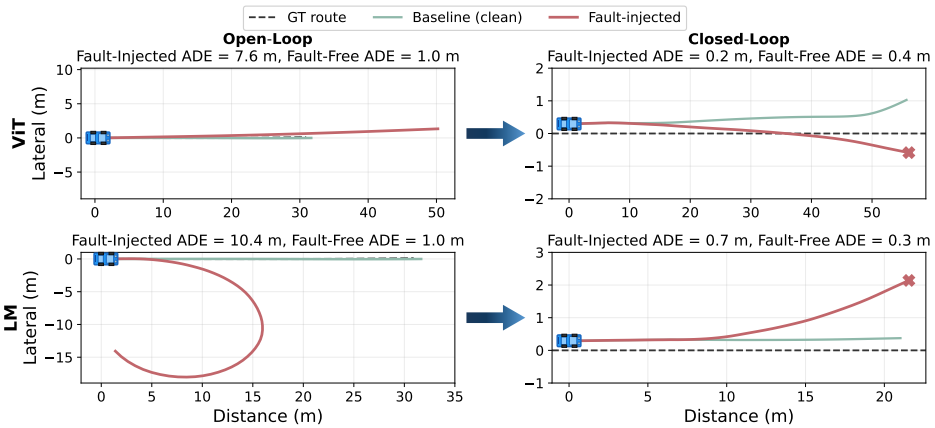


Fig. 17. Example comparison of open-loop predicted trajectories with closed-loop executed trajectories. The dashed line denotes the ground-truth route, while the green and red lines denote the fault-free and fault-injected trajectories, respectively. The red $\times$ denotes collision. Although both faults produce large open-loop ADE, their closed-loop trajectory deviations are substantially smaller while still resulting in collisions.

an ADE of only 0.2 m, lower than the 0.4 m ADE of the fault-free trajectory, yet the execution still results in an at-fault collision. Similarly, the LM example produces an open-loop ADE of 10.4 m, but its closed-loop trajectory ADE decreases to 0.7 m while still resulting in a collision.

These examples do not contradict the use of open-loop ADE in DiscoFI. Rather, they clarify its intended role. Open-loop ADE provides a statistical signal for prioritizing fault sites that are more likely to cause closed-loop failures; it is not a deterministic predictor of the trajectory deviation observed after closed-loop feedback. Once the predicted trajectory is applied to the vehicle, subsequent behavior is affected by vehicle dynamics, changing observations, and interactions with surrounding road users. Moreover, ADE measures average geometric displacement and does not fully capture safety-relevant factors such as speed, temporal alignment, headway, and proximity to other vehicles. Consequently, a trajectory that remains geometrically close to the reference route may still produce a collision.

Takeaway Message #2: Open-loop ADE is an effective but imperfect proxy for safety criticality. Closed-loop simulation remains necessary to validate whether ADE-prioritized faults lead to actual safety failures.

5.3 Divergence Between Chain-of-Thought and Trajectory Outcomes

We next examine whether the language model's chain-of-thought (CoT) output provides a reliable indication of the trajectory behavior and closed-loop safety outcome. This analysis is motivated by the structural difference between the two outputs. CoT generation consists of discrete token-level decisions, whereas trajectory generation produces continuous waypoints through the action expert. A fault may therefore leave the decoded CoT semantically plausible while still perturbing the internal representations used for trajectory generation. Conversely, a fault may corrupt the CoT output without causing an unsafe trajectory.

Figure 18 presents representative cases in which the open-loop CoT and the closed-loop safety outcome disagree. In Figure 18(a), the generated CoT remains correct for the driving context, for example advising the vehicle to maintain distance from a lead vehicle or slow down. However, the

Open-Loop CoT	Component	Closed-Loop
"Keep distance to the lead vehicle because it is ahead in the same lane"	ViT	REAR-END
"Keep distance to the lead vehicle because it is slowing."	ViT	AT-FAULT
"Keep distance to the lead vehicle because traffic ahead is slow"	LM	REAR-END
"Slow down to follow the lead vehicle"	ViT	AT-FAULT
"Slow down to follow the lead vehicle ahead."	ViT	AT-FAULT

(a) Correct open-loop CoT, but the same fault leads to closed-loop failure

Open-Loop CoT	Component	Closed-Loop
(no CoT emitted)	LM	BENIGN
"Keep lane to continue driving since no critical agent needs attention."	ViT	BENIGN
"Keep distance to the the traffic because the vehicles ahead is slow and the right-lane vehicles and a pedestrian on the in the the pedestrians, with the a gap for a lane change. no immediate safe gap. while the lanes."	LM	BENIGN
"Keep lane to continue driving since the lane ahead is clear."	ViT	BENIGN

(b) Corrupted or missing open-loop CoT, but the same fault results in benign closed-loop simulation

Fig. 18. Examples where open-loop chain-of-thought (CoT) outputs and closed-loop safety outcomes disagree.

same injected faults lead to rear-end or at-fault collisions during closed-loop execution. These cases show that a correct CoT does not guarantee a safe vehicle trajectory or benign closed-loop outcome. The opposite pattern is shown in Figure 18(b). Some injected faults corrupt the CoT semantics or suppress CoT generation entirely, yet the corresponding closed-loop executions remain benign. Thus, a corrupted or missing CoT does not necessarily imply a safety-critical trajectory error. These mismatches are observed for faults injected into both the ViT and LM components. The results indicate that CoT correctness and closed-loop trajectory safety are only partially coupled: monitoring CoT alone may miss unsafe executions and may also incorrectly flag benign ones.

Observation #8: CoT semantics and closed-loop safety outcomes can diverge. A contextually appropriate CoT may accompany a collision or system failure, while a corrupted or missing CoT may still result in benign vehicle behavior.

These results indicate that CoT semantics and trajectory safety are only partially coupled. Monitoring the CoT output alone would therefore produce both false negatives, where a plausible CoT accompanies an unsafe trajectory, and false positives, where a corrupted CoT accompanies benign vehicle behavior. CoT-based monitoring may provide useful auxiliary evidence, but it cannot replace direct validation of the generated trajectory and closed-loop behavior.

Takeaway Message #3: CoT monitoring should not be used as a standalone safety mechanism. It should be combined with trajectory-level validity checks and closed-loop safety monitoring.

6 Related Work

Testing and reliability of autonomous driving systems. Prior work has investigated the safety and robustness of autonomous driving systems from several perspectives. DeepXplore and DeepTest develop testing techniques for identifying erroneous or corner-case behaviors in deep-learning-based systems and autonomous-driving applications [29, 40]. Other studies examine safety checks, hazard coverage, and fault propagation in autonomous-driving stacks [28, 31, 39]. AVFuzzer and related approaches search for safety-critical driving scenarios by perturbing vehicle behaviors or environmental conditions [18]. However, these approaches generally treat the ADS as a black box and do not identify internal fault sites that can induce safety-critical behavior.

Several studies have applied fault injection to autonomous-driving systems [13–15, 31, 34]. Jha et al. [14] use Bayesian fault injection to explore the fault space. Rubaiyat et al. [31] develop an STPA-guided software fault-injection framework for OpenPilot that injects sensor-data faults based on identified hazard scenarios. TGFI [34], the closest work to ours, uses model-level importance scores to prioritize fault injection in autonomous-driving systems. TGFI only conducts closed-loop experiments without using any open-loop heuristics to accelerate the fault injection. However, existing studies either stop at model-level or open-loop outcomes without closed-loop validation, rely heavily on random fault injection, or evaluate closed-loop behavior without an explicit mechanism for connecting inexpensive open-loop evidence to safety-critical closed-loop outcomes. In contrast, DiscoFI establishes and exploits the relationship between open-loop ADE and closed-loop failures to guide a two-phase fault-injection process for a VLA-enabled ADS.

Reliability characterization of ML models. A large body of work studies the resilience of neural networks to transient hardware faults through fault injection and radiation testing [2, 8, 12, 16, 17, 20, 22, 30, 35, 37, 38]. Early studies characterize error propagation in CNNs and GPU kernels [12, 17], while more recent work evaluates transformer and language-model components using software-level fault injection, checksum-based analysis, or beam testing [2, 20, 22, 30, 37, 38].

These studies provide important insights into model-level vulnerability, but generally evaluate correctness using task-specific output metrics and do not trace faults through downstream planning, control, vehicle dynamics, and environmental feedback for autonomous driving tasks.

Unlike prior studies, DiscoFI unifies the two evaluation levels in a progressive, two-phase methodology. Its key novelty is to establish open-loop ADE as a practical proxy for closed-loop safety criticality and use this proxy to prioritize fault sites for expensive closed-loop validation. This design concentrates closed-loop simulation on faults that are substantially more likely to induce collisions or system failures. *To the best of our knowledge, DiscoFI is the first work to connect open-loop fault effects with closed-loop safety failures for transient hardware faults in a VLA-enabled autonomous-driving system.*

7 Limitations

Despite our efforts to mitigate potential threats to validity, several limitations remain. We discuss them below together with corresponding directions for future work.

Fault-model abstraction. Our study uses software-level single-bit fault injection in model weights and therefore does not capture all possible manifestations of hardware faults at the microarchitectural level. Establishing a precise correspondence between microarchitectural faults and software-visible corruptions would require cycle-accurate or hardware-aware fault injection at a scale that is impractical for our open-loop and closed-loop experiments. We use single-bit flips as a controlled perturbation primitive for studying how corrupted model states propagate through the VLA model and the downstream autonomous-driving pipeline. In deployed systems, some single-bit memory errors may be corrected by ECC when they occur in protected storage. However, not all model states and fault manifestations are necessarily covered by such protection, and multi-bit or logic-level faults can still induce software-visible corruptions. Prior work has also shown that single-bit and double-bit faults exhibit similar qualitative vulnerability trends, with double-bit faults generally producing more severe outcomes [32, 37, 38, 44]. Thus, our single-bit model is intended to characterize relative fault sensitivity and error-to-failure propagation rather than estimate hardware-level fault occurrence rates. This work focuses on the effect of single-bit flips in the AR1 system but they can be used as a guide for multi-bit faults. DiscoFI can be easily extended to a multi-bit fault model by selecting the exponent bit, followed by the first (highest bit) and then by its neighbouring lower bit(s). Extending the evaluation to multi-bit faults, computation faults, and hardware-specific fault models remains future work.

Evaluation Scope and Generalizability. Our evaluation is limited to simulation, the tested scenarios, and a single autonomous driving platform. Although closed-loop simulation captures vehicle dynamics and interactions with the driving environment, it cannot fully reproduce real-world deployments. Large-scale fault injection studies on physical vehicles are also impractical because of cost and safety constraints. Moreover, the distributions of high-ADE and safety-critical faults may differ across autonomous driving system architectures, model structures, software stacks, and driving conditions. Nevertheless, the two-phase design of DiscoFI is not specific to Alpamayo. ADE and related trajectory-displacement metrics are widely used for trajectory evaluation, and modern AI-enabled autonomous driving systems generally follow a pipeline from model inference to planning, control, and vehicle behavior. The central methodology of using inexpensive open-loop outcomes to prioritize costly closed-loop validation can be adapted to other autonomous driving platforms, although the ADE threshold and fault-selection strategy may require system-specific calibration. Broader validation across additional ADS platforms, traffic and weather conditions, rare driving events, and carefully controlled physical-vehicle experiments is subject to future work.

8 Conclusion

We present DiscoFI, a two-phase fault-injection framework for efficiently discovering safety-critical faults in autonomous driving systems. We establish that open-loop average displacement error (ADE) provides an effective proxy for closed-loop safety criticality: fault sites with higher open-loop ADE are substantially more likely to cause collisions or system failures during closed-loop execution. Based on this insight, Phase-I progressively identifies high-ADE fault sites through importance-based weight selection, component-specific bit selection, and bit-flip-direction selection, while Phase-II validates the selected sites through closed-loop autonomous-driving simulation. We implement DiscoFI in NVIDIA Alpamayo and evaluate it using AlpaSim under different experiment budgets and driving scenarios. DiscoFI consistently outperforms Random and TGFI, identifying $10.5\times$ and $6.8\times$ as many safety-critical faults as Random and TGFI baselines, respectively. These results demonstrate that DiscoFI effectively connects scalable open-loop fault exploration with costly closed-loop safety validation, enabling efficient reliability assessment of emerging VLA-enabled autonomous driving systems. We further use DiscoFI to characterize safety-critical faults in Alpamayo, revealing sparse and component-dependent safety-critical fault sites, limitations of ADE-only assessment, and inconsistencies between chain-of-thought outputs and actual trajectory safety.

Acknowledgement

This project is partially supported by the National Science Foundation (NSF) grants (#2402940, #2402941, #2402942, and #2410856) and the Commonwealth Cyber Initiative (CCI) grant (#HC-3Q24-047). This project is supported by resources provided by the Office of Research Computing at George Mason University (URL: <https://orc.gmu.edu>) and funded in part by grants from the National Science Foundation (Award Number 2018631).

Use of generative AI. ChatGPT and Claude are used to generate the figure-plotting scripts, slurm scripts, and to polish the writing. All AI-assisted scripts and text were reviewed, tested, and revised by the authors. The research methodology, implementation, experiments, data analysis, and conclusions were developed and verified by the authors.

References

- [1] [n. d.]. Autoware - the world's leading open-source software project for autonomous driving. <https://autoware.org/autoware-overview/>.
- [2] Udit Kumar Agarwal, Abraham Chan, and Karthik Pattabiraman. 2023. Resilience Assessment of Large Language Models under Transient Hardware Faults. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 659–670.
- [3] Rizwan A Ashraf, Roberto Gioiosa, Gokcen Kestor, Ronald F DeMara, Chen-Yong Cher, and Pradip Bose. 2015. Understanding the propagation of transient errors in HPC applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.
- [4] Jakub Breier, Dirmanto Jap, Xiaolu Hou, Shivam Bhasin, and Yang Liu. 2021. SNIFF: reverse engineering of neural networks with fault attacks. *IEEE Transactions on Reliability* 71, 4 (2021), 1527–1539.
- [5] Zitao Chen, Guanpeng Li, and Karthik Pattabiraman. 2021. A low-cost fault corrector for deep neural networks through range restriction. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 1–13.
- [6] Zitao Chen, Guanpeng Li, Karthik Pattabiraman, and Nathan DeBardeleben. 2019. BinFI: an efficient fault injector for safety-critical machine learning systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–23.
- [7] Zitao Chen, Niranjhana Narayanan, Bo Fang, Guanpeng Li, Karthik Pattabiraman, and Nathan DeBardeleben. 2020. Tensorfi: A flexible fault injection framework for tensorflow applications. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 426–435.
- [8] Fernando Fernandes dos Santos, Caio Lunardi, Daniel Oliveira, Fabiano Libano, and Paolo Rech. 2019. Reliability evaluation of mixed-precision architectures. In *2019 IEEE International Symposium on High Performance Computer*

- Architecture (HPCA)*. IEEE, 238–249.
- [9] Bo Fang, Xinyi Li, Harvey Dam, Cheng Tan, Siva Kumar Sastry Hari, Timothy Tsai, Ignacio Laguna, Dingwen Tao, Ganesh Gopalakrishnan, Prashant Nair, et al. 2024. Understanding Mixed Precision GEMM with MPGemmiFI: Insights into Fault Resilience. In *2024 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 166–178.
 - [10] Bo Fang, Karthik Pattabiraman, Matei Ripeanu, and Sudhanva Gurumurthi. 2014. GPU-Qin: A methodology for evaluating the error resilience of GPGPU applications. In *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*. IEEE, 221–230.
 - [11] The Autoware Foundation. 2015. Autoware: Open-source software for self-driving vehicles. <https://github.com/autowarefoundation>.
 - [12] Younis Ibrahim, Haibin Wang, Man Bai, Zhi Liu, Jianan Wang, Zhiming Yang, and Zhengming Chen. 2020. Soft Error Resilience of Deep Residual Networks for Object Recognition. *IEEE Access* 8 (2020), 19490–19503.
 - [13] Saurabh Jha, Subho S Banerjee, James Cyriac, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. 2018. AVFI: Fault injection for autonomous vehicles. In *2018 48th annual IEEE/IFIP international conference on dependable systems and networks workshops (DSN-W)*. IEEE, 55–56.
 - [14] Saurabh Jha, Subho S. Banerjee, Timothy Tsai, Siva Kumar Sastry Hari, Michael B. Sullivan, Zbigniew T. Kalbarczyk, Stephen W. Keckler, and Ravishankar K. Iyer. 2019. ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*. IEEE, 112–124. doi:10.1109/DSN.2019.00025
 - [15] Saurabh Jha, Timothy Tsai, Siva Hari, Michael Sullivan, Zbigniew Kalbarczyk, Stephen W Keckler, and Ravishankar K Iyer. 2019. Kayotee: A fault injection-based system to assess the safety and reliability of autonomous vehicles to faults and errors. *arXiv preprint arXiv:1907.01024* (2019).
 - [16] Jonathan Lew, Deval A Shah, Suchita Pati, Shaylin Cattell, Mengchi Zhang, Amruth Sandhupatla, Christopher Ng, Negar Goli, Matthew D Sinclair, Timothy G Rogers, et al. 2019. Analyzing machine learning workloads using a detailed GPU simulator. In *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 151–152.
 - [17] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W Keckler. 2017. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 1–12.
 - [18] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2020. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st international symposium on software reliability engineering (ISSRE)*. IEEE, 25–36.
 - [19] Guanpeng Li and Karthik Pattabiraman. 2018. Modeling input-dependent error propagation in programs. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 279–290.
 - [20] Yuhang Liang, Xinyi Li, Jie Ren, Ang Li, Bo Fang, and Jieyang Chen. 2025. ATTNChecker: Highly-Optimized Fault Tolerant Attention for Large Language Model Training. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*. 252–266.
 - [21] Haoxuan Liu, Vasu Singh, Michał Filipiuk, and Siva Kumar Sastry Hari. 2024. ALBERTA: ALgorithm-Based Error Resilience in Transformer Architectures. *IEEE Open Journal of the Computer Society* (2024).
 - [22] Kwondo Ma, Chandramouli Amarnath, and Abhijit Chatterjee. 2023. Error Resilient Transformers: A Novel Soft Error Vulnerability Guided Approach to Error Checking and Suppression. In *2023 IEEE European Test Symposium (ETS)*. IEEE, 1–6.
 - [23] Abdulrahman Mahmoud, Neeraj Aggarwal, Alex Nobbe, Jose Rodrigo Sanchez Vicarte, Sarita V Adve, Christopher W Fletcher, Iuri Frosio, and Siva Kumar Sastry Hari. 2020. Pytorchfi: A runtime perturbation tool for dnns. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 25–31.
 - [24] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11264–11272.
 - [25] Bin Nie, Devesh Tiwari, Saurabh Gupta, Evgenia Smirni, and James H Rogers. 2016. A large-scale study of soft-errors on GPUs in the field. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*. IEEE, 519–530.
 - [26] NVIDIA, Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, Liang Feng, Greg Heinrich, Jack Huang, Peter Karkus, Boyi Li, Pinyi Li, Tsung-Yi Lin, Dongran Liu, Ming-Yu Liu, Langechuan Liu, Zhijian Liu, Jason Lu, Yunxiang Mao, Pavlo Molchanov, Lindsey Pavao, Zhenghao Peng, Mike Ranzinger, Ed Schmerling, Shida Shen, Yunfei Shi, Sarah Tariq, Ran Tian, Tilman Wekel, Xinshuo Weng, Tianjun Xiao, Eric Yang, Xiaodong Yang, Yurong You, Xiaohui Zeng, Wenyuan Zhang, Boris Ivanovic, and Marco Pavone. 2025. Alpamayo-R1: Bridging Reasoning and Action Prediction for Generalizable Autonomous Driving in the Long Tail. *arXiv preprint arXiv:2511.00088* (2025).

- [27] Vladyslav Oles, Anna Schmedding, George Ostrouchov, Woong Shin, Evgenia Smirni, and Christian Engelmann. 2024. Understanding GPU Memory Corruption at Extreme Scale: The Summit Case Study. In *Proceedings of the 38th ACM International Conference on Supercomputing*. 188–200.
- [28] Matt Osborne, Richard Hawkins, and John McDermid. 2022. Analysing the safety of decision-making in autonomous systems. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 3–16.
- [29] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.
- [30] Lucas Roquet, Fernando Fernandes dos Santos, Paolo Rech, Marcello Traiola, Olivier Sentieys, and Angeliki Kritikakou. 2024. Cross-Layer Reliability Evaluation and Efficient Hardening of Large Vision Transformers Models. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 1–6.
- [31] Abu Hasnat Mohammad Rubaiyat, Yongming Qin, and Homa Alemzadeh. 2018. Experimental resilience assessment of an open-source driving agent. In *2018 IEEE 23rd Pacific rim international symposium on dependable computing (PRDC)*. IEEE, 54–63.
- [32] Behrooz Sangchoolie, Karthik Pattabiraman, and Johan Karlsson. 2017. One Bit is (Not) Enough: An Empirical Study of the Impact of Single and Multiple Bit-Flip Errors. In *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*. IEEE, 97–108.
- [33] Anna Schmedding. 2026. *Risk Assessment and Mitigation of Soft Errors in Autonomous Vehicle Neural Networks*. Ph. D. Dissertation. The College of William and Mary.
- [34] Anna Schmedding, Philip Schowitz, Xugui Zhou, Yiyang Lu, Lishan Yang, Homa Alemzadeh, and Evgenia Smirni. 2024. Strategic Resilience Evaluation of Neural Networks within Autonomous Vehicle Software. In *2024 43rd International Conference on Computer Safety, Reliability and Security (SafeComp)*.
- [35] Anna Schmedding, Lishan Yang, Adwait Jog, and Evgenia Smirni. 2024. Aspis: Lightweight Neural Network Protection Against Soft Errors. In *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 248–259.
- [36] Christoph Schorn, Andre Guntoro, and Gerd Ascheid. 2018. Efficient on-line error detection and mitigation for deep neural network accelerators. In *Computer Safety, Reliability, and Security: 37th International Conference, SAFECOMP 2018, Västerås, Sweden, September 19-21, 2018, Proceedings 37*. Springer, 205–219.
- [37] Yu Sun, Zachary Coalson, Shiyang Chen, Hang Liu, Zhao Zhang, Sanghyun Hong, Bo Fang, and Lishan Yang. 2025. Demystifying the Resilience of Large Language Model Inference: An End-to-End Perspective. *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (2025).
- [38] Yu Sun, Zhu Zhu, Cherish Mulpuru, Roberto Gioiosa, Zhao Zhang, Bo Fang, and Lishan Yang. 2025. Ft2: First-token-inspired online fault tolerance on critical layers for generative large language models. In *Proceedings of the 34th International Symposium on High-Performance Parallel and Distributed Computing*. 1–14.
- [39] Francesco Terrosi, Lorenzo Strigini, and Andrea Bondavalli. 2022. Impact of machine learning on safety monitors. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 129–143.
- [40] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314.
- [41] Devesh Tiwari, Saurabh Gupta, George Gallarno, Jim Rogers, and Don Maxwell. 2015. Reliability lessons learned from GPU experience with the Titan supercomputer at Oak Ridge leadership computing facility. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, Austin, TX, USA, November 15-20, 2015*. 38:1–38:12. doi:10.1145/2807591.2807666
- [42] Google Waymo. 2023. Waymo Driver. <https://waymo.com/>.
- [43] Lishan Yang, Bin Nie, Adwait Jog, and Evgenia Smirni. 2021. Enabling Software Resilience in GPGPU Applications via Partial Thread Protection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1248–1259.
- [44] Lishan Yang, Bin Nie, Adwait Jog, and Evgenia Smirni. 2021. Practical Resilience Analysis of GPGPU Applications in the Presence of Single- and Multi-Bit Faults. *IEEE Trans. Comput.* 70, 1 (2021), 30–44.
- [45] Lishan Yang, Bin Nie, Adwait Jog, and Evgenia Smirni. 2021. SUGAR: Speeding Up GPGPU Application Resilience Estimation with Input Sizing. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 1 (2021), 1–29.
- [46] Guangdong You, Shuncheng Tang, Jixiang Zhou, Hezhen Liu, Junfang Jiang, Yan-Fu Li, and Yinxing Xue. 2025. LOFT: An LLM-Enhanced Multi-Objective Search Framework for Fault Injection Testing of Autonomous Driving Systems. In *2025 IEEE 36th International Symposium on Software Reliability Engineering (ISSRE)*. 142–153. doi:10.1109/ISSRE66568.2025.00027
- [47] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9194–9203.

- [48] Xugui Zhou, Anna Schmedding, Haotian Ren, Lishan Yang, Philip Schowitz, Evgenia Smirni, and Homa Alemzadeh. 2022. Strategic safety-critical attacks against an advanced driver assistance system. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 79–87.

Received 10 July 2026; revised 5 October 2026; accepted 9 October 2026