NOT ALL ERRORS MATTER: SEMANTIC-AWARE SILENT DATA CORRUPTION DETECTION FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Although Large Language Models (LLMs) exhibit strong inherent fault tolerance and mask the effects of most GPU soft errors, a small fraction of these errors can still cause severe semantic deviations in their outputs, leaving LLM inference vulnerable to Silent Data Corruption (SDC) with potentially serious consequences for safety-critical applications such as autonomous driving and robotics. Detecting these consequential faults is challenging because the magnitude of numerical anomalies alone does not reliably indicate their semantic impact. Our key observation is that features in adjacent LLM layers exhibit strong predictability during fault-free execution, while Significant SDCs tend to disrupt this relationship, increasing the discrepancy between a layer's predicted and observed features. Building on this observation, we propose SIEVE, a lightweight framework for detecting Significant SDCs through deviations from learned inter-layer predictions. SIEVE trains a Nominal Inter-Layer Predictor on clean execution traces to predict the current layer's projected self-attention features from those of the preceding layer. At inference time, it extracts statistical and geometric discrepancies between the predicted and observed features of the current layer, aggregates them across monitored layer pairs and decoding steps into a Discrepancy Profile, and uses a lightweight XGBoost classifier to identify Significant SDCs. Across three LLMs and three VQA benchmarks, SIEVE achieves 96.38% precision, 91.49% recall, and 93.78% F1 on the primary evaluation set, with a false positive rate of 0.11%. Compared with representative SDC detection baselines, SIEVE improves F1 by up to 11.38% while introducing an average end-to-end overhead of 6.65%.

1 INTRODUCTION

In recent years, Large Language Models (LLMs), with their strong reasoning and generation capabilities, have been increasingly deployed in safety-critical domains such as autonomous driving Sima et al. (2024); Tian et al. (2024), remote sensing analysis Wang et al. (2025), medical diagnosis Alsabbagh et al. (2025), and industrial fault diagnosis Chen et al. (2025a). These models typically rely on large-scale GPUs or other AI accelerators, whose execution is susceptible to transient hardware faults induced by cosmic rays, voltage fluctuations, and device aging Wang et al. (2023); He et al. (2023). Such faults can silently alter computational results without triggering any system exception, forming Silent Data Corruption (SDC). Enterprises such as Meta and Google have reported the reliability challenges posed by SDC in large-scale GPU cluster operations Grattafiori et al. (2024); Vallin et al. (2025). To address this, the Open Compute Project (OCP) established the Server Component Resilience Workstream, uniting Meta, Google, NVIDIA, and other companies to study SDC. Its whitepaper on SDC in AI explicitly points out that, in large-scale inference, SDC pollutes massive inference results and disrupts downstream task decisions, and that inference-side SDC further compromises data integrity and degrades model effectiveness George et al. (2025). SDC therefore remains a serious safety hazard in model inference scenarios.

To investigate the impact of SDC on LLM outputs, we conduct extensive fault injection experiments, with results shown in Figure 1. Figures 1a and b show cases where the SDC does not change the semantics of the output and thus does not alter the model's decision; in contrast, Figures 1c and d show faulty outputs with severe semantic deviations that substantially affect the model's decision. We define a **Significant SDC** as an SDC whose faulty output deviates from the fault-free output and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

for which the evaluation by an LLM-as-a-Judge (supplemented by manual spot-checks) identifies semantic deviations such as severe factual errors. In safety-critical scenarios such as autonomous driving, this class of errors may cause severe consequences. Overall, not all errors warrant attention; this work focuses solely on **Significant SDCs**, which severely impact the results.

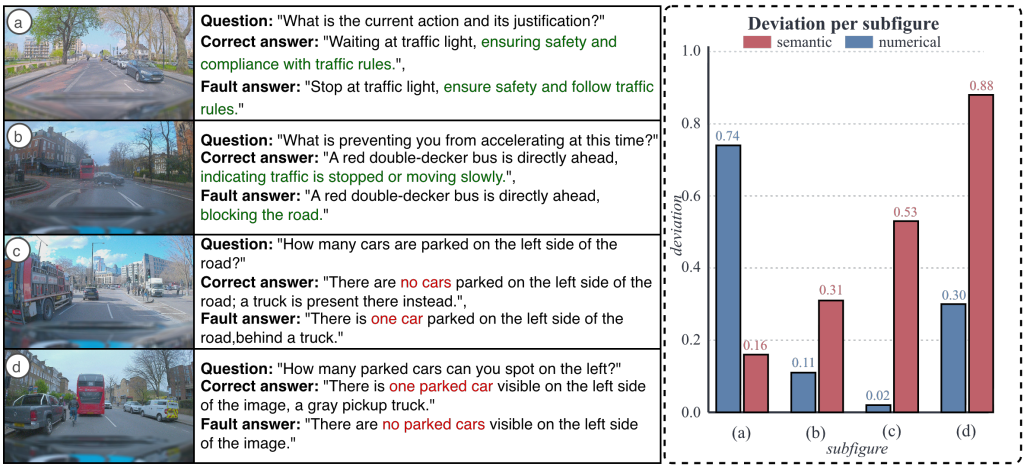


Figure 1: Examples of SDC outputs with different numerical and semantic deviations. The bar chart (right) compares the two deviation types for each subfigure, showing they do not correspond; numerical deviation is the change of the injected value before and after the bit flip, and semantic deviation is the semantic change relative to the fault-free output as judged by an LLM judge.

Existing soft error detection methods typically assess fault impact using fixed numerical thresholds: Ranger Chen et al. (2021) restricts activations to learned safe ranges and flags out-of-range deviations, while Dr.DNA Ma et al. (2024) computes detection thresholds from distribution-based deviations. However, as the bar chart in Figure 1 illustrates, there is no one-to-one correspondence between the numerical deviation and the semantic deviation of the output in LLMs: a large numerical perturbation may be absorbed by subsequent computation, while a small one may propagate through the network and alter key generated content. Methods with fixed numerical thresholds therefore tend to suffer from both missed Significant SDCs and false alarms on non-critical faults. An effective detector must further characterize how faults affect the semantic representations inside the model.

We propose **SIEVE**, a lightweight, semantics-aware Significant SDC detection framework for LLMs. SIEVE monitors selected adjacent decoder layers during inference, projects their intermediate representations into a low-dimensional space, and uses a Nominal Inter-Layer Predictor trained on clean execution traces to model the nominal inter-layer mapping. At runtime, the deviation of the actual representations from this mapping is aggregated into a layer-wise **Discrepancy Profile**; after further aggregation over multiple decoding steps, an independently calibrated lightweight classifier judges whether the current execution suffers a Significant SDC.

We systematically evaluate SIEVE on three representative LLMs and three VQA datasets. Under an evaluation protocol with semantically isolated splits and frozen thresholds (§4), SIEVE achieves an average F1 of 93.78% and a Non-significant-SDC false positive rate of 0.11% on the Test set of the nine tasks, with an end-to-end overhead of about 6.65%, substantially outperforming the Ranger and Dr.DNA baselines under identical configurations.

Our main contributions are summarized as follows:

- We formulate the Significant SDC problem for LLM inference, and reveal through systematic fault injection the inconsistency between the magnitude of numerical anomalies and the final semantic damage, thereby arguing the inherent limitations of the numerical-threshold detection paradigm.

- We propose SIEVE†, a lightweight online Significant SDC detection framework. SIEVE learns the nominal transformation of representations between adjacent layers from clean traces, and constructs a Discrepancy Profile from the deviation between predicted and actual representations as the detection signal for Significant SDCs, which a lightweight classifier then judges.
- We conduct a unified evaluation on three LLMs and three VQA datasets. Under the best configuration, SIEVE achieves an average F1 of 93.78%, a recall of 91.49%, and an FPR of 0.11% on the Test set, improving over same-configuration baselines by up to 11.38%.

2 Design Motivation

2.1 Focus on Significant SDCs

The inherent fault tolerance of models creates a fault-masking effect that substantially raises the difficulty of SDC detection: not every hardware fault that triggers an SDC leads to catastrophic consequences. Following the definition in §1, faults whose outputs suffer such severe semantic deviations are marked as *Significant SDCs*. As shown in Figure 2, when averaged over the three datasets, Significant SDCs account for 10.4%, 21.5%, and 67.8% of finite-value SDCs for Qwen2.5-VL-7B, InternVL3-8B, and LLaVA-1.5-7B, respectively. This indicates that most SDCs have limited semantic impact. Detection should therefore focus on the small subset of Significant SDCs with severe consequences, which also reduces the number of detection targets and the subsequent error-recovery cost.

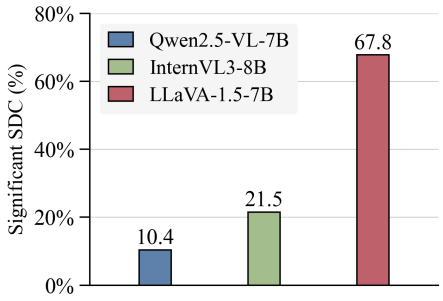


Figure 2: Mean fraction of Significant SDCs among finite-value SDCs.

2.2 Limitations of Numerical Deviation Detection

Existing soft error detection methods typically judge fault impact by whether activations exceed normal ranges or deviate from normal distributions, implicitly assuming that a larger numerical deviation corresponds to a more severe impact. However, in LLMs, numerical deviation and output semantic damage are correlated yet not equivalent. Figure 3 analyzes only faults whose injected scalar remains finite before and after the fault, and bins them by numerical deviation magnitude into three intervals: Significant SDCs already exist in the low-deviation interval, while in the high-deviation interval most faults cause no significant semantic damage. Setting a fixed numerical threshold at 1, for example, would miss 1.4% of the Significant SDCs and raise unnecessary alarms for 24.1% of the Non-significant SDCs. Therefore, Significant SDC detection cannot rely on the magnitude of numerical deviation itself, but must characterize how faults affect the model's internal features.

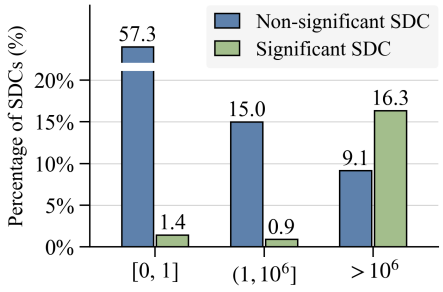


Figure 3: Distribution of Non-significant SDCs and Significant SDCs across three fault-induced value-deviation bins for finite-value faults.

3 Our Framework

3.1 Overview

The primary goal of runtime detection is to expose the difference between fault-free and faulty executions. SIEVE therefore uses a **Nominal Inter-Layer Predictor**, which predicts the output of the next decoder layer from that of the preceding one as a proxy for the fault-free reference; the

discrepancy between the predicted and observed features is then aggregated into a **Discrepancy Profile**, and a lightweight XGBoost detector performs the final decision.

The overall framework of SIEVE consists of three stages, as illustrated in Figure 4.

Stage I: Nominal Inter-Layer Predictor Training. At each decoding step, SIEVE collects the last-token representation of the self-attention output projection (`o_proj`) from the monitored decoder layers and reduces it to a low-dimensional space with a shared random orthogonal matrix. The predictor is then trained on clean execution traces to learn the nominal mapping between adjacent monitored layers, providing a per-step runtime reference for subsequent discrepancy computation.

Stage II: Significant SDC Detector Training. Under fault injection configurations, the Stage-I predictor estimates the nominal downstream representation for each selected layer pair, which is paired with the actually observed representation during faulty execution. Over the first K decoding steps, the cosine similarity, signed mean and standard-deviation differences, and L2 distance between the predicted and observed representations are aggregated into a Discrepancy Profile. Combined with semantic severity labels generated by an LLM judge, these profiles are used to train a lightweight XGBoost detector that identifies SDCs significantly impacting the semantic output of the model.

Stage III: Online Significant SDC Detection. During actual deployment, only the trained Nominal Inter-Layer Predictor and the XGBoost detector are executed online, performing real-time Significant SDC detection from runtime intermediate features.

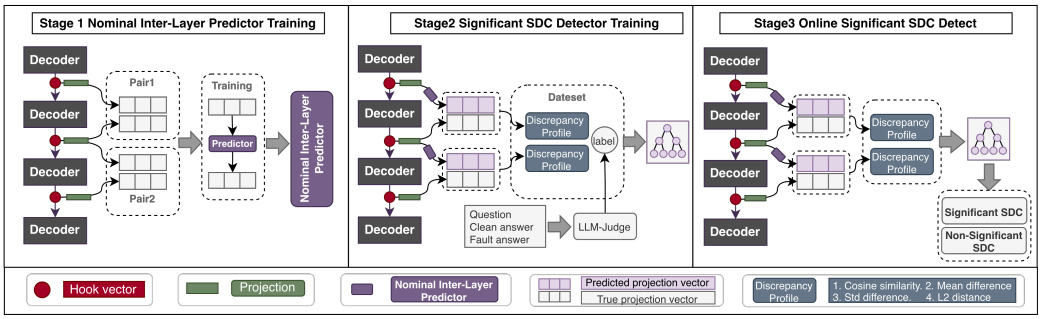


Figure 4: Overview of SIEVE. The outputs of each decoder layer are projected to a low-dimensional space, a Nominal Inter-Layer Predictor models the nominal mapping between adjacent layers, and the resulting Discrepancy Profile feeds an XGBoost detector to detect Significant SDCs.

3.2 Nominal Inter-Layer Predictor Training

3.2.1 Feature Collection

To establish comparable reference features for runtime detection, we leverage PyTorch forward hooks to collect, at each decoding step of clean execution, the last-token representation of the self-attention output projection (`o_proj`) of each monitored decoder layer, and organize adjacent monitored layers into feature pairs: the source-layer representation serves as the predictor's input while the target-layer representation acts as the supervision signal, learning the nominal inter-layer feature mapping. However, raw decoder features are extremely high-dimensional (e.g., 3,584 for Qwen2.5-VL-7B), making direct storage and training costly and increasing the difficulty of learning the mapping.

To this end, we project the features into a low-dimensional space. This design is inspired by the Johnson–Lindenstrauss (JL) lemma Johnson & Lindenstrauss (1984)—random low-dimensional projections approximately preserve the geometric relations among samples—and employs a shared column-orthogonal projection matrix that maps features into a 64-dimensional space. Figure A1 (appendix) verifies this empirically via heatmaps of adjacent-layer relations and correlation scatter plots before and after projection: the reduced representations remain highly consistent with the original space, with Pearson and Spearman correlations of 0.909 and 0.902. The 64-dimensional

projection thus preserves the dominant cross-layer geometric structure while substantially reducing both feature storage and predictor size.

3.2.2 PREDICTOR TRAINING

The predictor is trained to reconstruct the nominal feature of a target layer from the low-dimensional feature of its source layer. Since intermediate features require both element-wise numerical accuracy and the semantic consistency carried by their direction, we adopt a combined objective. Let $\hat{\mathbf{Y}}$ denote the predictor's output and $\mathbf{Y}$ the corresponding nominal target feature; the training loss is

$$L = \alpha L_{\text{mse}} + \beta L_{\cos}, \qquad L_{\cos} = 1 - \cos(\hat{\mathbf{Y}}, \mathbf{Y}),$$

where α and β are hyperparameters that balance the two loss terms, both set to 1.0 in this work. The MSE term ensures numerical fidelity, while the cosine term constrains directional consistency in the representation space, yielding stable and semantically coherent predictions. Architecturally, the predictor is a layer-aware residual MLP: each 64-dimensional source feature is concatenated with a 16-dimensional layer embedding and passed through 8 residual blocks (hidden size 64, dropout 0.1) to produce the 64-dimensional nominal feature of the target layer, allowing a single shared network to distinguish the mapping of each monitored layer pair. After training, the predictor performs only one lightweight forward pass per step at runtime, and its output serves as the reference for the subsequent Discrepancy Profile.

3.3 SIGNIFICANT SDC DETECTOR TRAINING

3.3.1 DISCREPANCY PROFILE

To capture Significant-SDC signatures from the nominal reference features provided by the predictor, we construct a Discrepancy Profile that characterizes each predicted–actual feature pair from two perspectives: (1) *statistical discrepancy*—the signed differences of mean and standard deviation, capturing global numerical shifts with their direction and changes in dispersion; and (2) *geometric discrepancy*—the cosine similarity and L2 distance, capturing directional consistency and absolute deviation magnitude. Each layer pair thus yields a four-dimensional discrepancy vector, which we aggregate over the first K decoding steps into the mean, maximum, and minimum of every metric and concatenate in a fixed order, giving a fixed 12-dimensional feature vector per pair; a configuration that monitors P layer pairs therefore presents a $12P$-dimensional input to the Detector.

3.3.2 DETECTOR DATASET CONSTRUCTION

Training data is produced by the fault injection experiments: for each faulty execution, the Discrepancy Profiles extracted from all monitored layer pairs form one training sample, which consists solely of aggregated low-dimensional statistics and is therefore compact and easy to train on.

Semantic labels are generated by Prometheus-7B Kim et al. (2024) as the LLM judge, which receives the original task instruction, the faulty output, and the fault-free baseline output, and scores the pair with a fixed three-level rubric: 2 for semantic equivalence, 1 for slight deviation with the core answer largely preserved, and 0 for a changed, missing, or contradictory core answer. A score of 0 is labeled Significant SDC, 1 as Non-significant SDC, and 2 (semantically equivalent) as Non-SDC. The label measures the incremental semantic impact of the fault relative to fault-free execution, not whether the model's answer is correct; all labels are spot-checked by manual review, and a unified system prompt, template, and rubric are used to ensure reproducibility.

3.3.3 SIGNIFICANT SDC DETECTOR

We adopt XGBoost as the detector, which efficiently models nonlinear interactions, and whose tree inference is far cheaper than a deep network, suiting online detection. After training, the Detector directly consumes the Discrepancy Profile and outputs the Significant SDC probability and binary decision, providing the basis for subsequent error recovery. Through this process, SIEVE reduces semantic error detection to a supervised classification problem over low-dimensional features, maintaining accuracy while keeping runtime overhead low.

3.4 Online Significant SDC Detection

During online deployment, SIEVE attaches to the LLM in a read-only bypass via PyTorch forward hooks. At each decoding step, the last-token `o_proj` features of the monitored layers are projected into a shared 64-dimensional space by a fixed column-orthogonal matrix; each pair's Nominal Inter-Layer Predictor estimates the target layer's nominal feature from the source layer, and the predicted–actual comparison forms the Discrepancy Profile. SIEVE aggregates features once over the first K decoding steps; if EOS occurs earlier, all available steps are used, ensuring coverage for short outputs. For each layer pair, the mean, maximum, and minimum of the four metrics are concatenated into a 12-dimensional Discrepancy Profile; this work uses 3 layer pairs, giving 36-dimensional features in total, which are fed into XGBoost to judge the Significant-SDC probability.

4 Experiments

4.1 Experimental Setup

Models, datasets, and splits. We evaluate SIEVE on Qwen2.5-VL-7B Wang et al. (2024b), InternVL3-8B Chen et al. (2024), and LLaVA-1.5-7B Liu et al. (2023a) using EarthVQA Wang et al. (2024a), LingoQA Marcu et al. (2024), and VQAv2 Goyal et al. (2017), yielding nine model–dataset tasks. Each task targets 5,000 fault-free executions and 50,000 random dual-bit fault injections; valid-execution counts are reported in Table A1 (appendix). Executions sharing the same image or question are assigned as a whole to disjoint Training / Validation / Test partitions (70% / 15% / 15%). Faulty Training records train the Detector; fault-free telemetry within Training is further split into predictor fit / early-stopping / evaluation subsets under the same group rule. Validation freezes thresholds; Test is used only for final scoring.

Fault injection. We emulate transient soft errors by software fault injection: during prefill, we randomly select a `torch.nn.Linear` module and flip two random bits of a random scalar in its output tensor. Only executions that complete normally and yield extractable features are counted. Each faulty output is labeled Non-SDC, Non-significant SDC, or Significant SDC by an LLM judge, from its semantic change relative to the fault-free baseline on the same input.

4.2 SDC Detector Evaluation

4.2.1 Detection Performance

We perform nested selection over 40 feature-dimension/window combinations within the outer Training set only, adopting 36D with $K = 28$: monitored pairs $(6, 7)$, $(24, 25)$, and $(26, 27)$, aggregated over the first $\min(28, T)$ actual decoding steps, where T is the total number of generated decoding steps (Appendix C). Neither Ranger nor Dr.DNA is publicly available, so we re-implement their detection mechanisms following the original papers to ensure a fair comparison under identical configurations rather than full reproduction of the original systems. All three methods share the same monitored layers, decoding steps, fault instances, and outer data partitions, and freeze their thresholds on the Validation set only; clean reference profiles for Ranger and Dr.DNA are built from a 20% Training-set group subset (details in Appendix B). At evaluation, all methods are scored on identical executions, matched row-by-row by sample identity, output, and fault metadata (coverage in Table A1). Each task's XGBoost Detector is trained on the Training set, with its threshold maximizing Significant-SDC F1 on Validation. As shown in Table 1, on the Test set SIEVE improves the nine-task average F1 over Ranger and Dr.DNA by 10.32% and 11.38%.

Some executions produce NaN or $\pm$Inf values among the monitored features after injection, and such extreme values are trivially detected by any method. To verify whether the methods genuinely distinguish the harder cases, we filter out these executions and re-score all three methods on the remaining ones whose 36 features are all finite, with no retraining or recalibration; Table 2 reports the result. The average F1 of Ranger and Dr.DNA drops from 83.46 and 82.40 in Table 1 to 37.74 and 22.74, whereas SIEVE degrades far less (93.78 to 68.78) and exceeds both baselines by a wide margin on most tasks, attaining the highest F1 on seven of the nine. Its advantage therefore comes from discriminating genuinely ambiguous finite-value faults rather than from obvious non-finite signals.

Table 1: Per-task Significant-SDC detection results on the Test set (%); P/R/F1 denote Precision/Recall/F1.

		EarthVQA			LingoQA			VQAv2		
Model	**Method**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Qwen2.5-VL-7B	Ranger	94.85	81.85	87.87	49.87	92.92	64.91	63.02	81.61	71.12
	Dr.DNA	100.00	61.48	76.15	52.33	74.06	61.33	79.71	63.22	70.51
	SIEVE	89.54	85.56	87.50	99.49	92.93	96.10	100.00	84.29	91.48
InternVL3-8B	Ranger	85.77	95.76	90.49	66.22	89.54	76.13	94.57	72.76	82.25
	Dr.DNA	97.63	78.39	86.96	99.43	70.02	82.17	99.12	67.00	79.95
	SIEVE	99.57	97.67	98.61	99.35	91.75	95.40	97.06	91.85	94.38
LLaVA-1.5-7B	Ranger	86.67	95.97	91.08	100.00	85.03	91.91	100.00	91.12	95.36
	Dr.DNA	100.00	95.97	97.95	100.00	85.56	92.22	100.00	89.35	94.38
	SIEVE	95.36	96.64	96.00	99.41	90.37	94.68	87.64	92.31	89.91
Average	Ranger	89.10	91.19	89.81	72.03	89.16	77.65	85.86	81.83	82.91
	Dr.DNA	99.21	78.61	87.02	83.92	76.55	78.57	92.94	73.19	81.61
	SIEVE	**94.82**	**93.29**	**94.04**	**99.42**	**91.68**	**95.39**	**94.90**	**89.48**	**91.92**

Table 2: Per-task Significant-SDC detection on the Test set after excluding executions with non-finite features (%).

		EarthVQA			LingoQA			VQAv2		
Model	**Method**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Qwen2.5-VL-7B	Ranger	83.78	55.86	67.03	20.80	77.61	32.81	27.75	50.00	35.69
	Dr.DNA	100.00	18.92	31.82	11.18	26.87	15.79	8.70	4.17	5.63
	SIEVE	72.73	64.86	68.57	98.11	77.61	86.67	100.00	57.29	72.85
InternVL3-8B	Ranger	58.56	84.13	69.06	30.37	65.56	41.51	60.38	18.93	28.83
	Dr.DNA	79.07	26.98	40.24	85.71	7.95	14.55	80.00	7.10	13.04
	SIEVE	98.29	91.27	94.65	97.35	72.85	83.33	90.14	75.74	82.32
LLaVA-1.5-7B	Ranger	21.43	50.00	30.00	0.00	0.00	0.00	100.00	21.05	34.78
	Dr.DNA	100.00	50.00	66.67	100.00	3.57	6.90	100.00	5.26	10.00
	SIEVE	50.00	58.33	53.85	90.91	35.71	51.28	21.43	31.58	25.53
Average	Ranger	54.59	63.33	55.36	17.06	47.72	24.77	62.71	29.99	33.10
	Dr.DNA	93.02	31.97	46.24	65.63	12.80	12.41	62.90	5.51	9.56
	SIEVE	**73.67**	**71.49**	**72.36**	**95.46**	**62.06**	**73.76**	**70.52**	**54.87**	**60.23**

4.2.2 Detection Overhead

Table 3: End-to-end overhead (%); the Baseline column reports mean latency in ms.

Model	Baseline (ms)	Ranger	Dr.DNA	SIEVE
Qwen2.5-VL-7B	764.4	+1.77	+3.82	+6.44
InternVL3-8B	984.3	+1.78	+5.20	+6.53
LLaVA-1.5-7B	363.8	+2.75	+5.67	+6.98

We measure end-to-end overhead on a single NVIDIA H20 GPU with batch size 1. All four detection modes use the same six monitored layers and run over the first $\min(28, T)$ actual decoding steps. Each model executes 10 warmup and 10 forward repetitions on 50 fixed LingoQA samples, yielding 500 observations per model–method pair. Answers under all monitored modes are identical to the baseline without detection. SIEVE's average overhead is 6.65%, versus 2.10% for Ranger and 4.89% for Dr.DNA.

4.3 Nominal Inter-Layer Predictor Evaluation

As shown in Table A3 (appendix), the trained Nominal Inter-Layer Predictor attains MSE from 0.0066 to 0.2207 and CosSim from 0.6367 to 0.9138 across the nine tasks; because activation scales differ across models, MSE is not comparable across models. We further examine how the prediction residual behaves under faults. Figure 5 shows the CosSim averaged over all actual decoding steps for each faulty run: Significant SDCs exhibit the lowest mean CosSim, while Non-SDC and Non-significant SDC overlap clearly, confirming that the predictor separates Significant SDCs from Non-significant SDCs well.

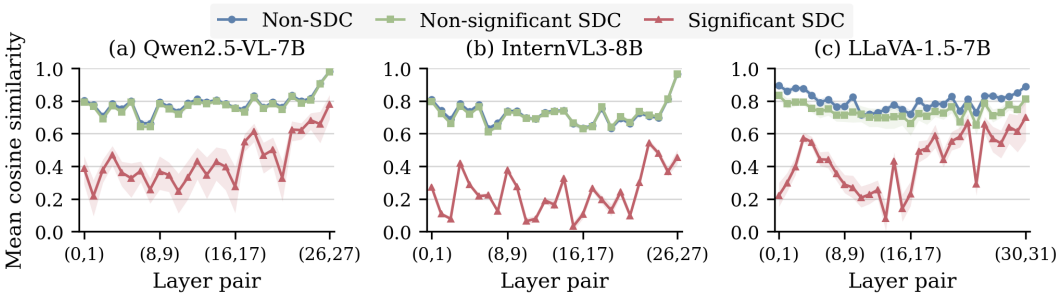


Figure 5: Average finite CosSim per adjacent layer pair over actual decoding steps on LingoQA, grouped by execution category. Shaded bands are 95% cluster bootstrap confidence intervals over images.

4.4 Ablation Study

We ablate layer pairs and metrics at the fixed 36D/K=28 configuration. Layer-pair ablation removes all 12 features of one pair; metric ablation removes the 9 features of CosSim, Mean Difference, Std Difference, or L2 Distance across the three pairs. We additionally score only the executions whose 36 features are all finite, after the configuration has been frozen. As Figure 6 shows, removing any component lowers the nine-task average F1. Removing $(26, 27)$ and $(24, 25)$ costs 2.47% and 2.16%, respectively; among metrics, Mean Difference is most critical (-3.74) and Std Difference contributes least (-0.46). The finite-value diagnostic shows the same overall trend. Complete per-task results are in Table A6, and those of the finite-value diagnostic are in Table A7 (appendix).

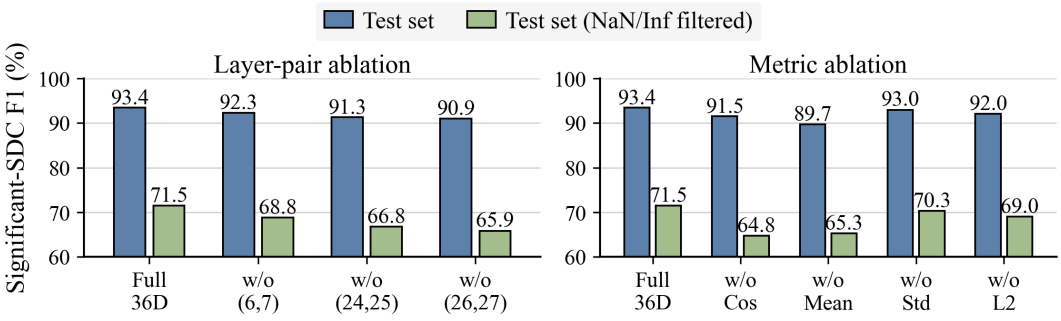


Figure 6: Component ablation at 36D/K=28. Left: layer-pair ablation; right: metric ablation. Bars show nine-task average Significant-SDC F1.

5 Related Work

Silent Data Corruption in Deep Learning Systems. Transient bit flips induced by cosmic rays, voltage noise, and device aging have long been studied as soft errors in digital microelectronics Dodd & Massengill (2003). As computational demand grows, such faults occur regularly in large

GPU fleets and can silently corrupt results without raising any system exception Wang et al. (2023; 2024c); He et al. (2023); Tan et al. (2025). For neural workloads, empirical studies further reveal that although most faults are naturally masked by the redundancy of activations and weights, a small fraction propagates into visibly wrong predictions Xue et al. (2023b); Chen et al. (2025b); in LLM training, SDC events have been reported to disrupt large-scale runs Ma et al. (2025). In LLM inference, a corrupted execution may surface as a fluent but semantically wrong answer, which is particularly dangerous in safety-critical deployments. These findings motivate a severity-aware view: the detection target should be the rare faults that materially change semantics, rather than all numerical deviations.

Fault Detection and Resilience Techniques for DNNs. Classical resilience mechanisms—ECC memory Sullivan et al. (2023), triple modular redundancy Mahmood & Khairullah (2025), and GPU checkpoint/restore Wei et al. (2025); Zeng et al. (2026)—protect hardware uniformly and incur substantial performance or resource overhead at LLM scale. Lightweight alternatives exploit the error tolerance of neural networks themselves: algorithm-based fault tolerance embeds checksums or consistency constraints into linear layers and attention Xue et al. (2023a); Liang et al. (2025); Ranger restricts activations to learned safe ranges to suppress fault propagation Chen et al. (2021); Dr.DNA detects SDCs from deviations of neuron-activation distributions Ma et al. (2024); and structural hardening designs include fault-tolerant attention Dai et al. (2025), first-token-inspired online protection of critical layers Sun et al. (2025), and sensitivity-guided selective protection Xie et al. (2025). All of these designs judge faults at the numerical or structural level, without asking whether a fault materially changes the generated semantics. As shown in §2.2, the two views diverge for LLMs: semantic severity is not monotonic in numerical deviation.

Semantic Measures of Generated Text. A rich literature quantifies the semantic equivalence of texts. Embedding-based measures, such as BERT-based similarity Devlin et al. (2019), Sentence-BERT Reimers & Gurevych (2019), SimCSE Gao et al. (2021), and the Universal Sentence Encoder Cer et al. (2018), compute similarity in a latent space and handle paraphrasing gracefully; cross-encoders jointly encode text pairs to capture finer-grained relations such as negation and entity mismatch Nogueira & Cho (2019); Deshpande et al. (2023); Lin (2025); and LLM-as-a-Judge protocols evaluate consistency against fixed rubrics with strong agreement with human judgment Liu et al. (2023b); Zheng et al. (2023). This work therefore uses an LLM-as-a-Judge with a fixed rubric offline to define and label Significant SDCs (§2.1).

6 Conclusion

This paper studied the semantic impact of soft errors in LLM inference. Numerical anomaly magnitude does not consistently track output severity, so detectors based on activation ranges or distribution shifts misjudge which faults matter. We proposed SIEVE, which models nominal inter-layer feature transitions from clean traces and judges Significant SDCs from the statistical and geometric discrepancy profile of prediction errors, without ground-truth answers or model modification. On three LLMs and three VQA datasets, SIEVE reaches an average F1 of 93.78% at an FPR of 0.11% with 6.65% end-to-end overhead, outperforming configuration-matched Ranger and Dr.DNA baselines by more than 10%. Scope limitations and future work are discussed in Appendix G.

AI use statement

In this work, we used generative AI tools for improving grammar and clarity, for drafting portions of the manuscript text, and for generating part of the experiment code. We have not used generative AI tools for research ideation or for analyzing results, and the remaining tasks requiring disclosure under the ICLR 2027 AI Policy for Authors are not applicable to this work. We have reviewed all AI-assisted work: all AI-assisted text was checked by the authors for accuracy and originality, and all AI-generated code was manually verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how to improve the reliability of large language model inference under hardware soft errors, with the goal of supporting safer deployment in safety-critical applications such as autonomous driving and robotics. We do not foresee any negative ethical consequences of our methods or findings. All experiments use publicly available models (Qwen2.5-VL, InternVL3, LLaVA) and publicly released VQA benchmarks (Appendix A); no human subjects were involved, no personal or user data were collected, and no new dataset is released. Severity labels are produced offline by an LLM-as-a-Judge protocol with a fixed rubric (§2.1), which is used solely for evaluation and never as part of the deployed detector. The authors declare no competing interests and confirm that the research was conducted in compliance with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide complete implementation details—model configurations, monitored layers, feature extraction, detector training, and evaluation protocols—in Appendix B, the full configuration-selection procedure in Appendix C, and dataset and evaluation-scope statistics in Appendix A. The fault-injection and evaluation code, together with scripts to regenerate all figures and tables, is provided in an anonymized repository: `https://anonymous.4open.science/r/SIEVE-FB16/`.

REFERENCES

Abdel Rahman Alsabbagh, Tariq Mansour, Mohammad Al-Kharabsheh, Abdel Salam Ebdah, Roa'a Al-Emaryeen, Sara Al-Nahhas, Waleed Mahafza, and Omar Al-Kadi. Minimedgpt: efficient large vision–language model for medical visual question answering. *Pattern Recognition Letters*, 189: 8–16, 2025.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

Jiao Chen, Ruyi Huang, Zuohong Lv, Jianhua Tang, and Weihua Li. Faultgpt: Industrial fault diagnosis question answering system by vision language models. *arXiv preprint arXiv:2502.15481*, 2025a.

Yuhang Chen, Zhen Tan, Ajay Kumar Jaiswal, Huaizhi Qu, Xinyu Zhao, Qi Lin, Yu Cheng, Andrew Kwong, Zhichao Cao, and Tianlong Chen. Bit-flip error resilience in llms: A comprehensive analysis and defense framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10425–10435, 2025b.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24185–24198, 2024.

Zitao Chen, Guanpeng Li, and Karthik Pattabiraman. 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)*, pp. 1–13. IEEE, 2021.

Huangliang Dai, Shixun Wu, Jiajun Huang, Zizhe Jian, Yue Zhu, Haiyang Hu, and Zizhong Chen. Ft-transformer: Resilient and reliable transformer with end-to-end fault tolerant attention. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1085–1098, 2025.

Ameet Deshpande, Carlos Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. C-sts: Conditional semantic textual similarity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5669–5690, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Paul E Dodd and Lloyd W Massengill. Basic mechanisms and modeling of single-event upset in digital microelectronics. *IEEE Transactions on nuclear Science*, 50(3):583–602, 2003.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6894–6910, 2021.

Nishant George, Harish Dattatraya Dixit, Emel Goksu, Bharath Parthasarathy, Amber Huffman, Sudhanva Gurumurthi, Vilas Sridharan, Thiago Macieira, Arani Sinha, Lisa Minwell, Dean Liberty, and Rob Chappell. Silent data corruption in ai. Technical report, Open Compute Project, 2025. Whitepaper, version 1.1.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Aaron Grattafiori et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yi He, Mike Hutton, Steven Chan, Robert De Gruijl, Rama Govindaraju, Nishant Patil, and Yanjing Li. Understanding and mitigating hardware failures in deep learning training systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1–16, 2023.

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26(1984):189–206, 1984.

Seungone Kim, Juyoung Shin, Yoonjoo Cho, Juri Levitan, Jimmy Xiong, Lixinyu Gao, Sangmee Park, Lian Pan, Jiawn Chen, Leonard Sigal, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuhang Liang, Xinyi Li, Jie Ren, Ang Li, Bo Fang, and Jieyang Chen. 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*, pp. 252–266, 2025.

Gechun Lin. Using cross-encoders to measure the similarity of short texts in political science. *American Journal of Political Science*, 69(4):1600–1616, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023a.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 2511–2522, 2023b.

Dongning Ma, Fred Lin, Alban Desmaison, Joel Coburn, Daniel Moore, Sriram Sankar, and Xun Jiao. Dr. dna: Combating silent data corruptions in deep learning using distribution of neuron activations. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 239–252, 2024.

Jeffrey Jian Ma, Hengzhi Pei, Leonard Lausen, and George Karypis. Understanding silent data corruption in llm training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20372–20394, 2025.

Hamed A Mahmood and Shawkat S Khairullah. Design and simulation of a dependable architecture using triple modular redundancy for embedded cyber-physical systems. *Journal of Electronic Testing*, 41(1):63–74, 2025.

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, et al. Lingoqa: Visual question answering for autonomous driving. In *European Conference on Computer Vision*, pp. 252–269. Springer, 2024.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *European conference on computer vision*, pp. 256–274. Springer, 2024.

Michael B Sullivan, Mohamed Tarek Ibn Ziad, Aamer Jaleel, and Stephen W Keckler. Implicit memory tagging: No-overhead memory safety using alias-free tagged ecc. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1–13, 2023.

Yu Sun, Zhu Zhu, Cherish Mulpuru, Roberto Gioiosa, Zhao Zhang, Bo Fang, and Lishan Yang. 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*, pp. 1–14, 2025.

Jingweijia Tan, Xurui Li, An Zhong, Kaige Yan, Xiaohui Wei, and Guanpeng Li. Gerem: Fast and precise error resilience assessment for gpu microarchitectures. *IEEE transactions on parallel and distributed systems*, 2025.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

Carlos Vallin, Rama Bhimanadhuni, and Robert S Chappell. Silent data corruption (sdc) challenges in modern ai public cloud deployments. *IEEE Micro*, 2025.

Junjue Wang, Zhuo Zheng, Zihang Chen, Ailong Ma, and Yanfei Zhong. Earthvqa: Towards queryable earth via relational reasoning-based remote sensing visual question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 5481–5489, 2024a.

Junjue Wang, Weihao Xuan, Heli Qi, Zhihao Liu, Kunyi Liu, Yuhan Wu, Hongruixuan Chen, Jian Song, Junshi Xia, Zhuo Zheng, et al. Disasterm3: A remote sensing vision-language dataset for disaster damage assessment and response. *arXiv preprint arXiv:2505.21089*, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

Shaobu Wang, Guangyan Zhang, Junyu Wei, Yang Wang, Jiesheng Wu, and Qingchao Luo. Understanding silent data corruptions in a large production cpu population. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 216–230, 2023.

Shaobu Wang, Guangyan Zhang, Junyu Wei, Yang Wang, Jiesheng Wu, and Qingchao Luo. Understanding silent data corruption in processors for mitigating its effects. *ACM Transactions on Architecture and Code Optimization*, 21(4):1–27, 2024c.

Xingda Wei, Zhuobin Huang, Tianle Sun, Yingyi Hao, Rong Chen, Mingcong Han, Jinyu Gu, and Haibo Chen. Phoenixos: Concurrent os-level gpu checkpoint and restore with validated speculation. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, pp. 996–1013, 2025.

Tong Xie, Jiawang Zhao, Zishen Wan, Zuodong Zhang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. Realm: Reliable and efficient large language model inference with statistical algorithm-based fault tolerance. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, 2025.

Xinghua Xue, Cheng Liu, Feng Min, Tao Luo, and Yinhe Han. Approxabft: Approximate algorithm-based fault tolerance for neural network processing. *arXiv preprint arXiv:2302.10469*, 2023a.

Xinghua Xue, Cheng Liu, Ying Wang, Bing Yang, Tao Luo, Lei Zhang, Huawei Li, and Xiaowei Li. Soft error reliability analysis of vision transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 31(12):2126–2136, 2023b.

Shaoxun Zeng, Tingxu Ren, Jiwu Shu, and Youyou Lu. GPU checkpoint/restore made fast and lightweight. In *24th USENIX Conference on File and Storage Technologies (FAST 26)*, pp. 239–254, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Dataset and Evaluation-Scope Statistics

Reporting convention. Unless explicitly marked as excluding non-finite features, detection performance in the appendix is reported on all retained executions. Some executions exhibit NaN or $\pm\mathrm{Inf}$ in their post-injection monitored features, which are trivially flagged; re-scoring after removing these executions serves only as a supplementary conditional diagnostic under the frozen 36D/$K{=}28$ configuration: it requires all 36 features of the representation to be finite, and involves no retraining or threshold recalibration; it does not participate in candidate selection.

Each task targets 5,000 clean executions and 50,000 dual-bit injections. Table A1 counts executions with successfully extracted 36D features: 494,571 in total, of which 45,000 are clean and 449,571 injected. Non-SDC, Non-significant SDC, and Significant SDC are defined only among injected executions. After removing executions whose 36D/$K{=}28$ features contain NaN or $\pm\mathrm{Inf}$, the Test set retains 72,645/74,586 executions and 779/2,720 Significant-SDC executions. Because this filter removes 1,941 Significant-SDC executions, the all-finite subset does not represent the original fault distribution and serves only as a supplementary diagnostic.

Table A1: Valid executions with extractable 36D features, injection labels, and Test-set statistics.

Model	Dataset	Valid	Injected	Non-SDC	Non-sig.	Sig.	Sig. (%)	Test (all / all-finite)
Qwen2.5-VL-7B	EarthVQA	54,896	49,896	37,087	11,151	1,658	3.32	8,236 / 8,077
	LingoQA	54,962	49,962	39,907	8,456	1,599	3.20	8,249 / 8,104
	VQAv2	54,949	49,949	41,383	6,873	1,693	3.39	8,376 / 8,211
InternVL3-8B	EarthVQA	54,871	49,871	35,773	11,052	3,046	6.11	8,236 / 7,890
	LingoQA	54,955	49,955	38,172	8,570	3,213	6.43	8,237 / 7,891
	VQAv2	54,957	49,957	38,323	8,410	3,224	6.45	8,375 / 8,041
LLaVA-1.5-7B	EarthVQA	54,997	49,997	48,333	494	1,170	2.34	8,250 / 8,113
	LingoQA	54,987	49,987	48,061	698	1,228	2.46	8,245 / 8,086
	VQAv2	54,997	49,997	48,779	61	1,157	2.31	8,382 / 8,232

B Implementation Details

Baseline fairness. Neither Ranger nor Dr.DNA provides a public implementation, so both are configuration-matched adapted baselines that we re-implemented following the original papers, not full reproductions of the original systems. The three methods use the same six monitored layers, the first $\min(28, T)$ actual decoding steps, the same fault instances and outer splits, and each selects its

threshold on the full Validation set. Clean reference profiles for Ranger/Dr.DNA come from 20% groups of the outer Training set, capped at 1,000 clean samples; SIEVE's Predictor uses the fault-free clean telemetry of the outer Training set. The comparison therefore controls for monitoring scope, fault instances, and evaluation protocol, but does not claim identical clean-profiling budgets. Per-task results for all three methods under both evaluation scopes are in Tables 1 and 2 of §4; all scored rows are matched row-by-row by execution identity, output, and fault metadata.

Table A2: Fixed implementation and statistical settings.

Component	**Setting**
Fault injection	Random scalar in prefill intermediate activations; two random bits flipped per execution; seed 42.
Projection	Shared 64D column-orthogonal projection (single matrix per model, generated from seed 42) applied to all monitored layers.
Predictor	Layer-aware residual MLP: 16D source/target layer embeddings, hidden size 64, 8 residual blocks, dropout 0.1.
Predictor training	AdamW, learning rate 5e-4, weight decay 1e-4, batch size 2,048; at most 500 epochs; ReduceLROnPlateau and patience-10 early stopping.
Detector	XGBoost: learning rate 0.01, max depth 6, 10,000 trees, subsample/colsample 0.9, early stopping 500.
Threshold calibration and statistics	Maximize Significant-SDC F1 on the Validation set; nine-task average; per-task 95% CI from image/question-group bootstrap with 10,000 resamples.

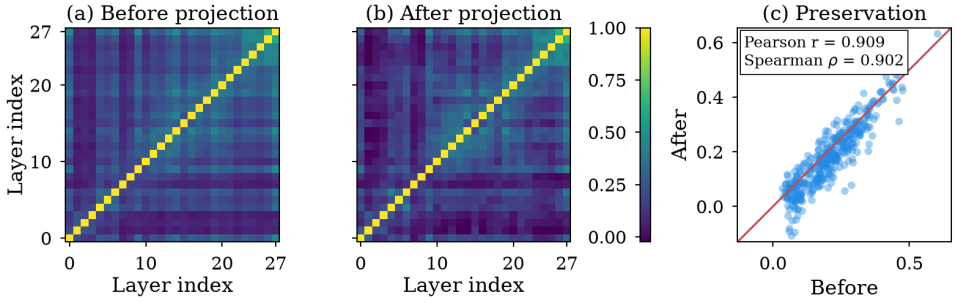


Figure A1: Feature structure preserved by the 64-dimensional column-orthogonal projection (Qwen2.5-VL-7B, LingoQA): pairwise similarity before vs. after projection.

Table A3: Nominal Inter-Layer Predictor performance on the group-disjoint 15% predictor evaluation subset within the outer Training set.

Model	EarthVQA		LingoQA		VQAv2	
	MSE	CosSim	MSE	CosSim	MSE	CosSim
Qwen2.5-VL-7B	0.0844	0.8589	0.1451	0.7761	0.2207	0.6934
InternVL3-8B	0.0684	0.8117	0.1094	0.7149	0.1399	0.6367
LLaVA-1.5-7B	0.0066	0.9138	0.0162	0.7557	0.0178	0.7326

C Configuration Selection

Selection criterion. We select among 40 candidates spanning 4 feature dimensions and 10 window sizes. All candidates are evaluated only within the outer Training set via nested validation; the nine-task average Significant-SDC F1 is the sole selection and ranking metric, and neither the internal Validation set nor the Test set participates. Figure A2 (left) gives the complete 4×10 matrix; the right panel re-scores the same candidate predictions against the fixed all-finite reference of the final

36D/K=28 configuration as an all-finite conditional diagnostic. Table A4 reports average metrics at fixed K=28, and Figure A3 gives window curves for both all retained executions and the all-finite subset at fixed 36D.



Figure A2: Training-only 4×10 configuration-selection diagnostic. Left: nine-task average Significant-SDC F1, the only result used for candidate selection and ranking; right: average F1 scored on the fixed subset with all 36 features finite, after configuration freezing. Red boxes mark the 36D/K=28 selected and frozen by the primary metric; the two panels use independent color ranges.

Table A4: Training-only average metrics (%) by feature dimension at fixed K=28.

Feature	Precision	Recall	F1	FPR
36D	**99.17**	**88.66**	**93.42**	**0.028**
48D	93.66	88.99	91.00	0.192
60D	92.77	88.22	89.93	0.270
72D	95.04	88.68	91.52	0.137

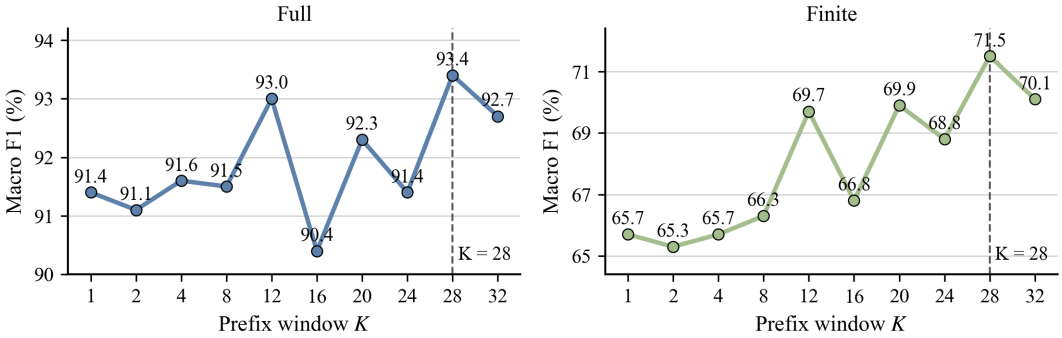


Figure A3: Training-only prefix-window diagnostic at fixed 36D. Left: average F1; right: average F1 scored on the fixed all-finite subset after configuration freezing. Each point is annotated with F1 (%); dashed lines mark K=28 selected by the primary metric.

Fixed at K=28, 36D attains the highest F1 and the lowest FPR. The left panel of Figure A3 shows that at fixed 36D, K=28 reaches the highest F1 (93.42%); the right-side all-finite diagnostic likewise peaks at K=28 (71.49%). We therefore freeze 36D/K=28, with the all-finite filter serving only as a post-freeze conditional diagnostic.

Post-freeze all-finite diagnostic. After configuration freezing, we run a conditional diagnostic within the same nested validation using the fixed 36D/K=28 all-finite reference vector. The filter changes only scored rows; training and threshold calibration are unchanged. Table A5 therefore

explains performance under the numerically finite condition and plays no role in selecting or ranking the 40 candidates.

Table A5: Training-only diagnostics of the frozen 36D/K=28 configuration on all retained executions and the all-finite subset (nine-task average, %).

Scope	Rows / Sig. SDC	Precision	Recall	F1	FPR
All	53,076 / 1,912	99.17	88.66	93.42	0.028
All-finite	51,753 / 589	94.86	60.16	71.49	0.028

D Component Ablation Details

Tables A6–A7 give the per-task F1 (%) behind Figure 6 in the main text. All variants are retrained, calibrated, and evaluated on the same nested partitions within the outer Training set; the outer Validation and Test sets do not participate. Table A6 covers all retained executions and is the primary result for component conclusions; Table A7 reports the conditional diagnostic on the fixed 36D/K=28 all-finite subset. The all-finite filter changes only scored rows; all variants are scored on the same subset, while thresholds remain frozen on the full internal Validation set. `−P` removes the 12 features of the corresponding layer pair; `−Cos/−Mean/−Std/−L2` removes the 9 features of that metric across all three pairs.

Table A6: Per-task component ablation, Significant-SDC F1 on all retained executions (%).

Task	36D (all)	-P6/7	-P24/25	-P26/27	-Cos	-Mean	-Std	-L2
InternVL3-8B / EarthVQA	97.02	97.05	97.37	97.20	97.04	96.85	97.19	97.02
InternVL3-8B / LingoQA	98.15	98.14	98.60	98.75	91.93	98.14	98.14	98.29
InternVL3-8B / VQAv2	96.70	96.70	96.12	95.45	94.86	96.27	96.27	96.27
LLaVA-1.5-7B / EarthVQA	96.85	95.45	96.85	96.47	95.69	96.88	96.09	96.88
LLaVA-1.5-7B / LingoQA	91.60	84.21	85.11	78.43	90.77	72.73	90.23	91.95
LLaVA-1.5-7B / VQAv2	92.70	92.24	81.06	88.52	92.24	84.71	92.70	84.71
Qwen2.5-VL-7B / EarthVQA	96.66	96.05	96.66	95.78	96.95	96.66	96.95	96.95
Qwen2.5-VL-7B / LingoQA	87.58	87.93	87.58	87.42	82.66	82.90	85.55	82.32
Qwen2.5-VL-7B / VQAv2	83.54	82.53	82.04	80.50	81.07	82.04	83.54	83.79
Average	**93.42**	**92.26**	**91.26**	**90.95**	**91.47**	**89.68**	**92.96**	**92.02**

Table A7: Per-task component ablation, Significant-SDC F1 on the all-finite subset (%).

Task	36D (all)	-P6/7	-P24/25	-P26/27	-Cos	-Mean	-Std	-L2
InternVL3-8B / EarthVQA	88.75	89.16	90.24	89.57	89.02	88.05	89.44	88.75
InternVL3-8B / LingoQA	93.62	93.48	95.03	95.56	75.86	93.48	93.41	93.99
InternVL3-8B / VQAv2	87.50	87.50	85.56	84.39	81.68	86.19	86.19	86.19
LLaVA-1.5-7B / EarthVQA	60.00	60.00	60.00	57.14	47.62	63.64	54.55	63.64
LLaVA-1.5-7B / LingoQA	45.00	28.57	30.00	21.43	36.84	16.67	40.91	46.15
LLaVA-1.5-7B / VQAv2	45.16	40.00	19.35	33.33	40.00	26.42	45.16	26.42
Qwen2.5-VL-7B / EarthVQA	89.32	87.38	89.32	86.79	90.20	89.32	90.20	90.20
Qwen2.5-VL-7B / LingoQA	71.01	71.94	71.01	70.15	62.96	63.35	69.82	62.11
Qwen2.5-VL-7B / VQAv2	63.01	61.33	60.53	54.41	58.97	60.53	63.01	63.45
Average	**71.49**	**68.82**	**66.78**	**65.86**	**64.80**	**65.29**	**70.30**	**68.99**

E Online Overhead Details

The overhead evaluation uses the final 36D/K=28 configuration: monitored pairs $(6, 7)$, $(24, 25)$, and $(26, 27)$, over the first $\min(28, T)$ actual decoding steps. Each model–method pair runs 10 warmup and 10 forward repetitions on 50 LingoQA samples, yielding 500 observations; all monitored modes retain 100% answer agreement.

Table A8: LingoQA online overhead details. Latency in ms, overhead in %.

Model	Method	Mean	P50	P95	Overhead
Qwen2.5-VL-7B	No Detection	764.4	727.0	951.3	0.00
	Ranger	777.9	738.9	971.2	1.77
	Dr.DNA	793.6	753.8	995.5	3.82
	SIEVE	813.7	773.1	1025.6	6.44
InternVL3-8B	No Detection	984.3	932.5	1374.9	0.00
	Ranger	1001.8	948.9	1397.2	1.78
	Dr.DNA	1035.5	981.2	1446.3	5.20
	SIEVE	1048.7	992.7	1460.4	6.53
LLaVA-1.5-7B	No Detection	363.8	249.3	640.8	0.00
	Ranger	373.9	254.1	663.8	2.75
	Dr.DNA	384.5	258.4	689.5	5.67
	SIEVE	389.2	263.2	698.5	6.98

F Detector Transfer

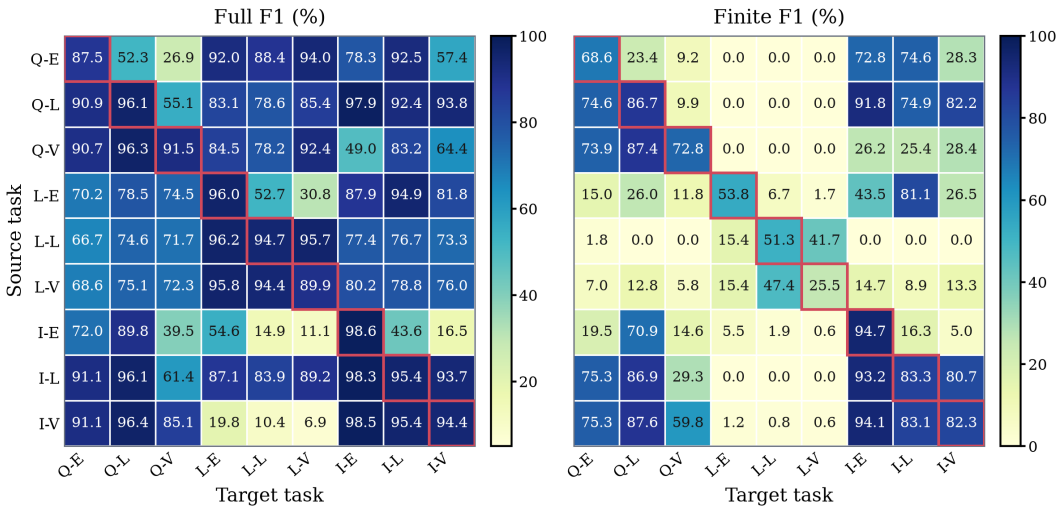
We evaluate the 9×9 transfer of the frozen 36D/K=28 SIEVE Detector, with target-specific clean reference profiles, across the nine model–dataset tasks. For each source $\rightarrow$ target cell, the Detector and its maximum-F1 threshold are trained and frozen on the source's Training/Validation; the target builds its clean reference profiles only from its own fault-free executions and generates 36D Test-set features. Target Training/Validation fault labels play no role in Detector training, threshold calibration, or tuning, and target Test-set labels are used only for offline scoring. All tasks use the same random dual-bit injections, with telemetry recorded for up to 50 decoding steps. The all-finite subset requires all 36 features of target Test-set executions to be finite, serving only as a conditional diagnostic without changing predictions, models, or thresholds.

The nine diagonal cells in Figure A4 (all retained executions) reproduce the §4.2.1 Test-set results item by item (average F1 = 93.78%). Across the 72 off-diagonal cells, the average F1 is 73.46% with an FPR of 3.97%, clearly higher than the 0.11% in-domain; the average F1 on the all-finite subset is 28.91%, a supplementary diagnostic under fixed conditions. These results indicate that some source–target combinations transfer, but target-specific clean reference profiles remain necessary, and there is a substantial calibration mismatch between the frozen source threshold and the target distribution; this does not constitute evidence of zero-shot deployment without target-side data.

Table A9: Averaged Significant-SDC metrics (%) by transfer relationship, all retained executions.

Relationship	Cells	Precision	Recall	F1	FPR
In-domain	9	96.38	91.49	93.78	0.11
Same model, cross dataset	18	71.95	89.14	73.55	5.81
Cross model, same dataset	18	90.06	72.64	75.24	3.19
Cross model, cross dataset	36	84.07	75.66	72.53	3.45
Off-diagonal average	**72**	**82.54**	**78.28**	**73.46**	**3.97**

Each matrix cell is bootstrapped 10,000 times over the target's image/question groups; per-cell F1 and 95% CIs are provided with the accompanying reproduction results. Because the same source



Q/L/I: Qwen2.5-VL/LLaVA-1.5/InternVL3; E/L/V: EarthVQA/LingoQA/VQAv2. Red outlines: in-domain diagonal.

Figure A4: 9×9 transfer F1 (%) of the frozen 36D/K=28 Detector. Rows are sources, columns are targets; source Detectors and thresholds are fixed, and each target uses only its own clean reference profiles to generate features. Left: all retained executions; right: all-finite conditional diagnostic after configuration freezing. Red boxes mark the in-domain diagonal. Q/L/I denote Qwen2.5-VL, LLaVA-1.5, InternVL3; E/L/V denote EarthVQA, LingoQA, VQAv2.

Table A10: Averaged Significant-SDC metrics (%) by transfer relationship, all-finite subset.

Relationship	**Cells**	**Precision**	**Recall**	**F1**	**FPR**
In-domain	9	79.88	62.81	68.78	0.11
Same model, cross dataset	18	62.09	54.09	43.28	5.81
Cross model, same dataset	18	49.68	25.53	23.67	3.19
Cross model, cross dataset	36	48.16	29.52	24.35	3.45
Off-diagonal average	**72**	**52.02**	**34.66**	**28.91**	**3.97**

threshold is applied directly to different target distributions, the off-diagonal FPR and performance gaps reflect the actual transfer calibration mismatch rather than a re-selected operating point.

G LIMITATIONS

Our conclusions are bounded by prefill-stage random dual-bit injection, fixed-rubric LLM-as-a-Judge labels, and target-side clean reference profiles; detector transfer shows calibration mismatch across tasks rather than zero-shot deployability. Future work includes robustness to diverse fault models and integration with recovery mechanisms.