Jiajia JIAO, Yixu YU, 2026. RetryTrigger: intelligent inference duplication for enhancing LLM resilience to hardware transient faults. *ENGINEERING Information Technology & Electronic Engineering*, *27*(4):250104.
https://doi.org/10.1631/ENG.ITEE.2025.0104

RetryTrigger: intelligent inference duplication for enhancing LLM resilience to hardware transient faults

Key words: Large language models; System resilience; Intelligent fault detection; Inference duplication; Transient faults

Corresponding author: Jiajia JIAO
E-mail: jiaojiajia@shmtu.edu.cn
ORCID: https://orcid.org/0000-0003-3680-787X

Motivation

• Large language models (LLMs) are increasingly deployed in translation, QA, sentiment analysis, text completion, and code generation, but their inference pipelines are vulnerable to hardware transient faults.

• Transient bit flips during inference can silently corrupt intermediate activations and produce silent data corruptions (SDCs), degrading output quality without raising an exception.

• To address the limitations of existing inference-phase protection schemes, this paper investigates how to reduce fault-induced SDC while maintaining low latency overhead, without requiring hardware modifications or model retraining.

Main idea

- **Fault-aware detection via runtime feature exploitation**

A lightweight LightGBM-based classifier uses runtime features such as maximum probability, entropy, and logits statistics to detect suspicious inference behaviors with high accuracy and minimal overhead.

- **Flexible plug-and-play re-execution framework**

A pure-software two-stage framework dynamically flags suspicious inferences and selectively triggers online recomputation, without hardware modifications or model retraining.

- **Superior resilience–overhead tradeoff**

RetryTrigger reduces SDC rates by 92.97% on average and up to 95.33%, while maintaining only 4.1167% mean runtime overhead and as low as 2.4012% in the best case.

Method

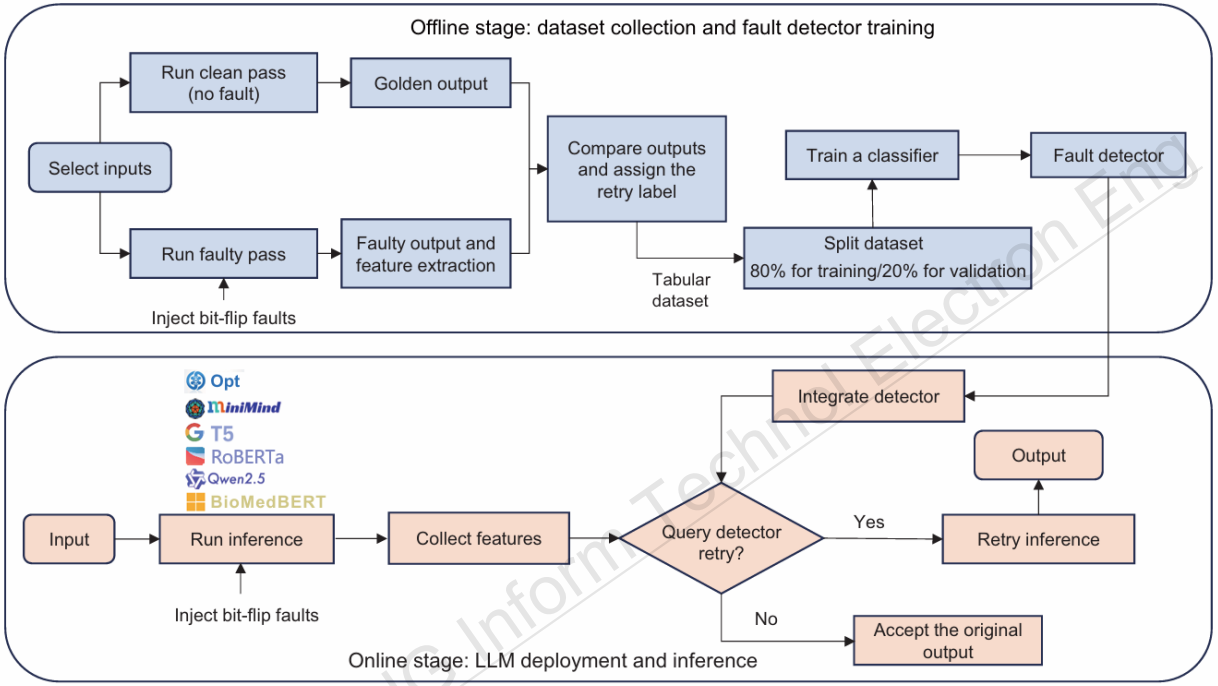

Fig. 1 The RetryTrigger pipeline, which comprises two main stages: (1) offline data curation, which involves collecting runtime features from the model under controlled fault injection to build a dataset for the LightGBM detector; (2) online deployment, where the trained detector analyzes an LLM's runtime features in real time to predict whether a retry is necessary

- Offline stage: run clean inference and fault-injected inference, extract runtime features, compare outputs with the golden reference, and train a LightGBM detector on the resulting tabular dataset.
- Online stage: during deployment, collect the same runtime features in real time and query the detector to decide whether to accept the output or trigger one retry.
- Because transient faults are non-persistent, one re-execution is usually sufficient; a strict single-retry budget keeps latency bounded.

Method

Table 1 Summary of the 11 runtime features used in RetryTrigger. Each feature is computed from tensors readily available during inference (logits z, probabilities p, runtime Δt) and incurs negligible extra cost. Features are grouped into three categories with corresponding a priori hypotheses regarding their universality across models and environments

Feature	Definition/Formula	Rationale	Group	Hypothesis
max_prob	$p_{(1)} = \max_t p_t$	Proxy for local confidence; faults can spuriously inflate or depress confidence	Confidence	Universal
top2_gap	$p_{(1)} - p_{(2)}$	Measures separation between the best and the second-best tokens; faults distort typical gap patterns	Confidence	Universal
entropy	$-\sum_t p_t \log p_t$	Quantifies global uncertainty; faults either flatten or sharpen the distribution abnormally	Distributional shape	Universal
top2_prob	$p_{(2)}$	Complements $p_{(1)}$, capturing secondary probability peaks	Confidence	Universal
top3_prob	$p_{(3)}$	Further complements $p_{(1)}$, revealing abnormal tail behaviors	Confidence	Universal
logits_mean	$\mu = \frac{1}{V}\sum_t z_t$	First moment of logits; faults can shift the baseline mean	Distributional shape	Model-specific
logits_std	$\sigma = \sqrt{\frac{1}{V}\sum_t (z_t - \mu)^2}$	Dispersion of logits; extreme values from faults often inflate the standard deviation	Distributional shape	Universal
runtime	Δt per forward pass	Behavioral side-channel; faults may co-occur with micro-stalls or hardware hiccups	Behavioral	Environment-specific
top10_prob_mass	$\sum_{t=1}^{10} p_{(t)}$	Concentration of probability mass; flattening from faults leaks mass to irrelevant tokens	Distributional shape	Model-specific/Task-dependent
skewness	$\mathbb{E}\left[\left(\frac{z_t-\mu}{\sigma}\right)^3\right]$	Asymmetry of logits; confident outputs are right-skewed, and faults reduce skewness	Distributional shape	Model-specific
kurtosis	$\mathbb{E}\left[\left(\frac{z_t-\mu}{\sigma}\right)^4\right] - 3$	Peakedness of logits; faults flatten peaks, lowering kurtosis	Distributional shape	Model-specific

Method

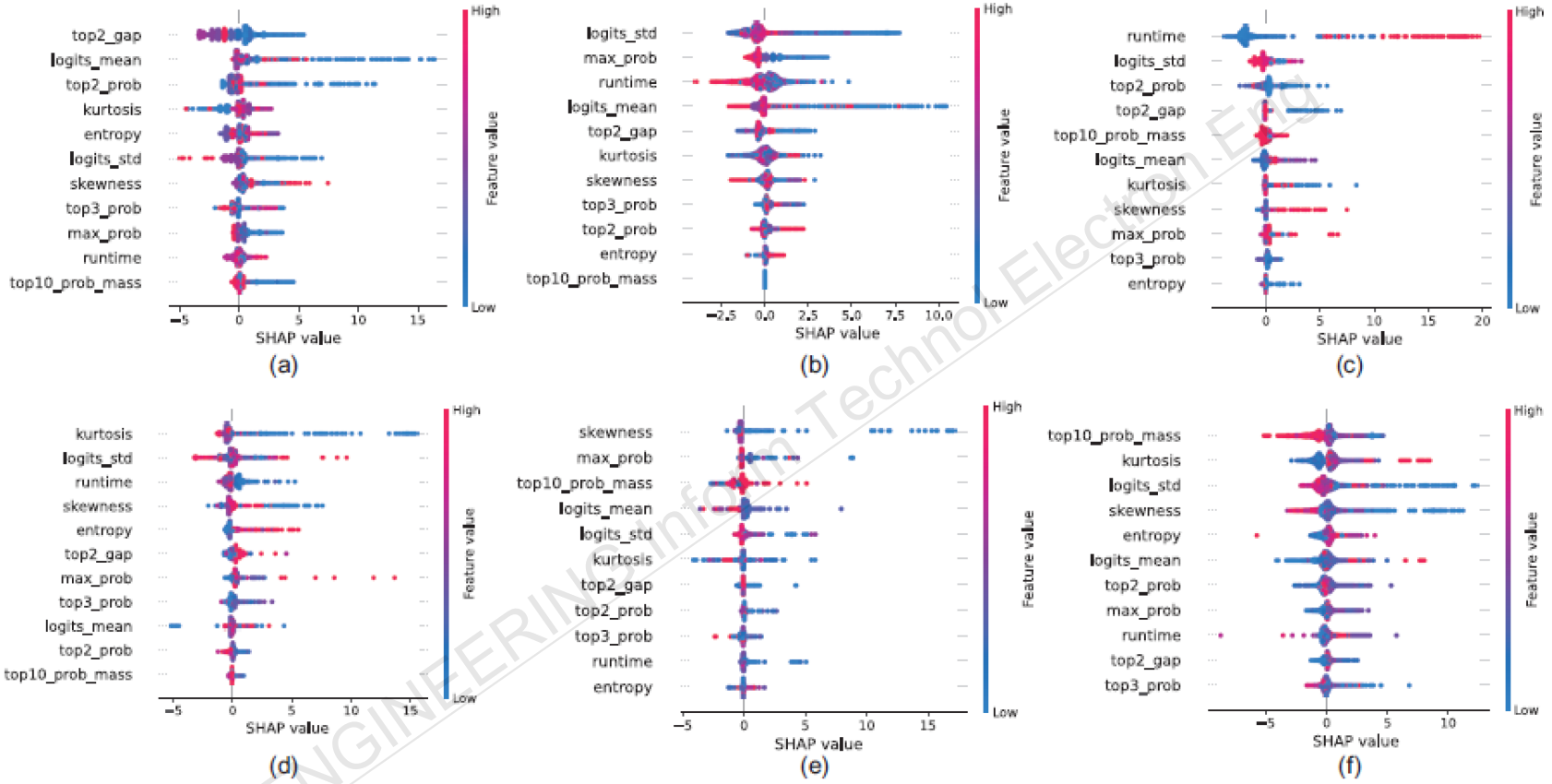

Fig. 2 SHAP summary plots for the trained RetryTrigger detector on each of the seven LLMs, based on the models trained with the full feature set. For each feature, the plot shows the impact of its value (color-coded from low/blue to high/red) on the model's prediction for the positive class (SDC=1). Positive SHAP values push the prediction toward "retry", while negative values push it toward "no-retry". Subfigures (a–f) correspond to the seven LLMs: (a) BioMedBERT; (b) RoBERTa; (c) T5-Small; (d) Qwen2.5-Coder-0.5B/7B; (e) MiniMind; (f) Opt

Results

Table 6 Detailed comparison of meta-classifier performance on the validation set for detecting SDCs. For each LLM, we reported precision, recall, and F1-score for both class 0 (no-retry) and class 1 (retry)

LLM	Classifier	Class 0 (no-retry)			Class 1 (retry)		
		Precision	Recall	F1-score	Precision	Recall	F1-score
MiniMind	LightGBM	0.9998	1.0000	0.9999	1.0000	0.9900	0.9898
	XGBoost	1.0000	0.9998	0.9999	0.9803	1.0000	0.9900
	Logistic regression	1.0000	0.9970	0.9985	0.6578	1.0000	0.7936
T5-Small	LightGBM	0.9990	0.9995	0.9993	0.9943	0.9887	0.9915
	XGBoost	0.9990	0.9995	0.9993	0.9943	0.9887	0.9915
	Logistic regression	0.9950	0.9347	0.9639	0.5439	0.9435	0.6900
RoBERTa	LightGBM	0.9989	0.9864	0.9926	0.7657	0.9770	0.8585
	XGBoost	0.9994	0.9837	0.9915	0.7350	0.9885	0.8431
	Logistic regression	0.9921	0.9210	0.9552	0.3258	0.8390	0.4694
BioMedBERT	LightGBM	0.9958	0.9973	0.9966	0.9390	0.9058	0.9221
	XGBoost	0.9953	0.9958	0.9955	0.9047	0.8941	0.8994
	Logistic regression	0.9850	0.8971	0.9390	0.2304	0.6941	0.3460
Qwen2.5-Coder-0.5B	LightGBM	0.9997	1.0000	0.9998	1.0000	0.9863	0.9931
	XGBoost	0.9997	0.9997	0.9997	0.9863	0.9863	0.9863
	Logistic regression	0.9997	0.9949	0.9973	0.7826	0.9863	0.8727
Qwen2.5-Coder-7B	LightGBM	0.9996	0.9998	0.9997	0.9924	0.9849	0.9886
	XGBoost	0.9990	0.9990	0.9990	0.9629	0.9629	0.9629
	Logistic regression	0.9995	0.9950	0.9972	0.7824	0.9753	0.8682
Opt	LightGBM	0.9998	1.0000	0.9999	1.0000	0.9890	0.9944
	XGBoost	0.9998	1.0000	0.9999	1.0000	0.9890	0.9944
	Logistic regression	0.9998	0.9840	0.9918	0.5172	0.9890	0.6792
Average	LightGBM	**0.9989**	**0.9976**	**0.9983**	**0.9559**	**0.9745**	**0.9626**
	XGBoost	**0.9989**	0.9968	0.9978	0.9376	0.9728	0.9525
	Logistic regression	0.9959	0.9605	0.9776	0.5486	0.9182	0.6742

The best result for each metric is highlighted in bold

Results

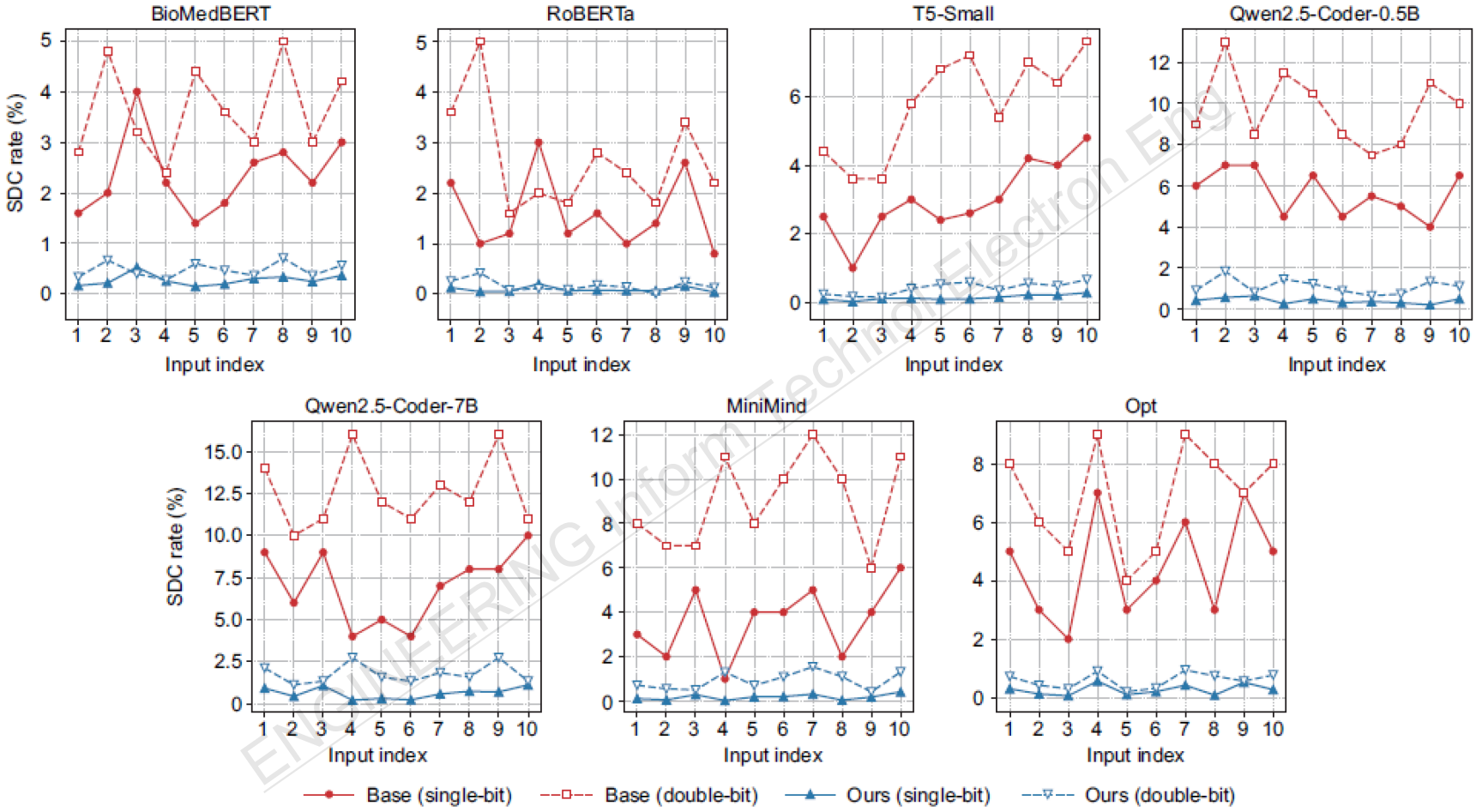

Fig. 3 Comparison of SDC rates across seven LLMs under SBU and DBU transient fault injections. Each model is tested with 10 distinct input samples. The red lines represent the baseline method without protection, showing high volatility and SDC rates. The blue lines represent our proposed RetryTrigger method. Solid lines denote SBU fault scenarios, while dashed lines denote DBU fault scenarios. The visualization highlights that RetryTrigger effectively mitigates SDC rates under different fault models compared to the baseline

Results

Table 7 Relative overhead analysis of the RetryTrigger method under SBU fault injection. The total overhead is the percentage of runs that trigger a retry, composed of TP-induced and FP-induced retries

Model	Baseline SDC (P_{err})	$Recall_1$ (TPR)	$1 - Recall_0$ (FPR)	TP-induced overhead (%)	FP-induced overhead (%)	Total overhead (retry rate (%))
BioMedBERT	0.0236	0.9058	0.0027	2.1376	0.2636	**2.4012**
RoBERTa	0.0160	0.9770	0.0136	1.5632	1.3382	**2.9014**
T5-Small	0.0300	0.9887	0.0005	2.9661	0.0485	**3.0146**
Qwen2.5-Coder-0.5B	0.0565	0.9863	0.0000	5.5725	0.0000	**5.5725**
Qwen2.5-Coder-7B	0.0700	0.9849	0.0002	6.8943	0.0186	**6.9129**
MiniMind	0.0360	0.9900	0.0000	3.5640	0.0000	**3.5640**
Opt	0.0450	0.9890	0.0000	4.4505	0.0000	**4.4505**
Average	0.0396	0.9745	0.0024	3.8783	0.2384	**4.1167**

Table 9 Extra latency comparison for encoder–decoder and decoder-only models

Model	Policy	$T_{baseline}$ (s)	T_{faulty_avg} (s)	P_{retry} (%)	T_{avg_total} (s)	Absolute overhead (s)
BioMedBERT	Per-token					
	Post-hoc	0.0067	0.0171	2.4012	0.0172	0.0105
RoBERTa	Per-token					
	Post-hoc	0.0068	0.0157	2.9014	0.0158	0.0090
Qwen2.5-Coder-0.5B	Per-token	0.5451	0.7594	5.5725	0.7897	0.2446
	Post-hoc	0.6172	1.5662	5.5725	1.6005	0.9833
Qwen2.5-Coder-7B	Per-token	0.7814	1.0228	6.9129	1.0768	0.2954
	Post-hoc	0.6506	1.6527	6.9129	1.6976	1.0470
Opt	Per-token	0.1678	0.4624	4.4505	0.4698	0.3020
	Post-hoc	0.1560	0.8637	4.4505	0.8706	0.7146
T5-Small	Per-token	0.0482	0.0966	3.0146	0.0980	0.0498
	Post-hoc	0.0519	0.0706	3.0146	0.0721	0.0202
MiniMind	Per-token	0.7408	0.8043	3.5640	0.8307	0.0899
	Post-hoc	0.6990	0.8001	3.5640	0.8250	0.1260

$T_{baseline}$ and T_{faulty_avg} are the measured average time in second; P_{retry} is the total overhead (retry rate) for each model, as reported in Table 7; T_{avg_total} is the calculated final average latency considering retries; absolute overhead is calculated as $T_{avg\_total} - T_{baseline}$

Conclusions

RetryTrigger is proposed for enhancing LLM resilience via dynamically collecting runtime features and leveraging a LightGBM meta-model to accurately predict whether inference outputs should be retried. Extensive evaluations on seven representative LLMs demonstrated that RetryTrigger achieves up to 95.33% and an average of 92.97% SDC reduction, with minimal performance overhead averaging only 4.1167%. This good balance between robustness and efficiency makes RetryTrigger a practical solution for deploying robust LLMs in resource-constrained or real-time environments.

Future work will focus on enhancing compatibility with black-box application programming interfaces via logit-agnostic variants, and validating with real hardware fault traces to further strengthen ecological validity.

Jiajia JIAO is an associate professor at the College of Information Engineering, Shanghai Maritime University. She received her PhD from Shanghai Jiao Tong University and was a visiting scholar for one year each at Carnegie Mellon University and Cornell University in the United States. Her main research interests focus on large language model resilience and machine learning assisted optimization of computer reliability and hardware security.

Yixu YU received his BS degree in 2024 from Jinling Institute of Technology, China. Currently he is pursuing his MS degree. His research interests include reliable large language models, hardware transient fault tolerance, and the robustness of machine learning systems under fault injection.