LIGHTWEIGHT DETECTION OF SILENT DATA CORRUPTION IN DISTRIBUTED DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable detection of silent data corruption (SDC), such as bit-flip errors, is critical in large-scale neural network training, as undetected hardware faults can silently propagate and severely degrade model performance. We introduce a lightweight detection method integrated directly before collective communication steps, enabling localization of faulty devices with minimal runtime overhead. Our approach combines statistical modeling of gradient norms with divergence-based criteria to improve robustness. Experiments on large-scale training workloads, including LLaMA2 - 7B, show that our detector successfully identifies the vast majority of high-order bit-flip faults in bfloat16 while incurring only a very small computational overhead, offering a strong balance between detection accuracy and efficiency.

1 INTRODUCTION

As the size and complexity of machine learning models continue to grow, the probability of hardware-related failures and silent data corruption (SDC) Constantinescu et al. (2008) events increase substantially. Even single-bit errors, especially in high-order exponent or sign bits, can silently propagate through distributed training pipelines, leading to compromised model weights and unstable convergence Ghemawat et al. (2003), He et al. (2023). While some low-magnitude flips (e.g., mantissa $1 \rightarrow 0$) may be masked by optimizers, catastrophic flips often remain undetected. Traditional fallback strategies, such as complete checkpoint rollback or complete system restart, often prove costly and disruptive, especially for long-running, resource-intensive workloads involving hundreds or thousands of compute nodes Gemini Team (2025).

We propose a lightweight detector for SDC that is integrated into the communication layer of distributed training. By inserting checks before collective operations (e.g., all-reduce, all-gather), corrupted tensors can be identified before they propagate, enabling accurate device localization and real-time isolation. Our method combines multimodal detection of gradient norms with distance-based divergence metrics, achieving 99% detection accuracy with zero false positives at less than 0.5% runtime overhead in large-scale LLaMA2 - 7B training Touvron et al. (2023). We have implemented our solution in torch_npu adapter for Huawei Ascend devices, our approach requires no changes to high-level training code, offering a practical and scalable solution for reliable training.

2 BACKGROUND

SDC denotes an undetected error in the output of a computing device, often triggered by hardware defects and highly dependent on instruction sequences Constantinescu et al. (2008). While early discussions focused on CPUs Ghemawat et al. (2003), modern heterogeneous ML stacks extend this risk to NPUs, and other accelerators. In practice, an affected device may silently miscompute even basic arithmetic without raising runtime exceptions, absent systematic software checks, such faults remain invisible to the training pipeline. This work adopts that a broader, device-agnostic view of SDC and focuses on its implications for large-scale, distributed training where undetected errors can propagate across nodes via collective communication.

2.1 BIT-FLIP

Machine learning models are typically trained with IEEE 754–compliant floating-point data types. In this format, even a single bit inversion can cause drastically different numerical effects depending on its position. Flipping low-order mantissa bits usually produces minor perturbations that optimizers and normalization layers can absorb, whereas altering high-order exponent or sign bits leads to severe deviations in magnitude, often resulting in NaN/INF values or numbers many orders of magnitude away from the original He et al. (2023). For example, flipping the leading exponent bit of a small float32 value can inflate it from $\approx 10^{-11}$ to $\approx 10^{28}$, while flipping a low mantissa bit changes the value by less than 1%. Figure 1 illustrates how inverting different bit positions systematically alters numerical values, with exponent flips producing particularly extreme distortions.

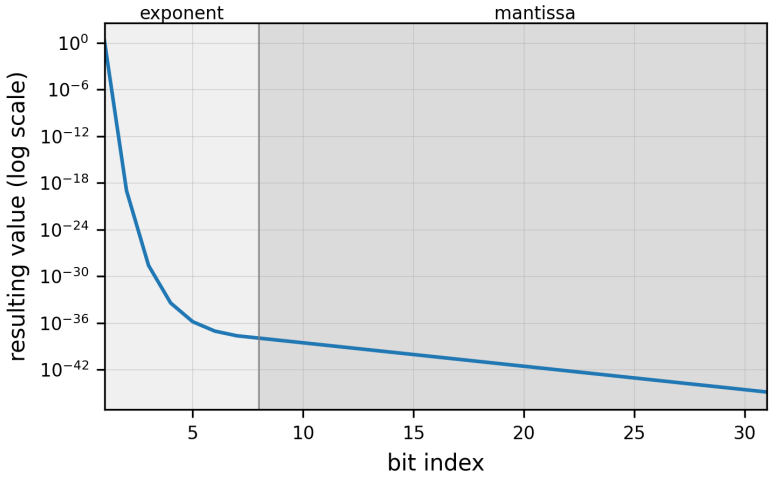


Figure 1: Impact of individual bit-flips in the IEEE 754 float32 representation of zero. The resulting values are plotted on a logarithmic scale. Shaded regions indicate mantissa and exponent fields.

The effect of a bit inversion can be expressed formally as

$$bff(x, i, n) = x \oplus 2^{n-i-1}, \tag{1}$$

where x is the original value represented with n bits, i is the flipped bit position (indexed from left to right and started from 0), and $\oplus$ denotes bitwise XOR applied to the binary representation of x.

For bitflips in the exponent of a float number:

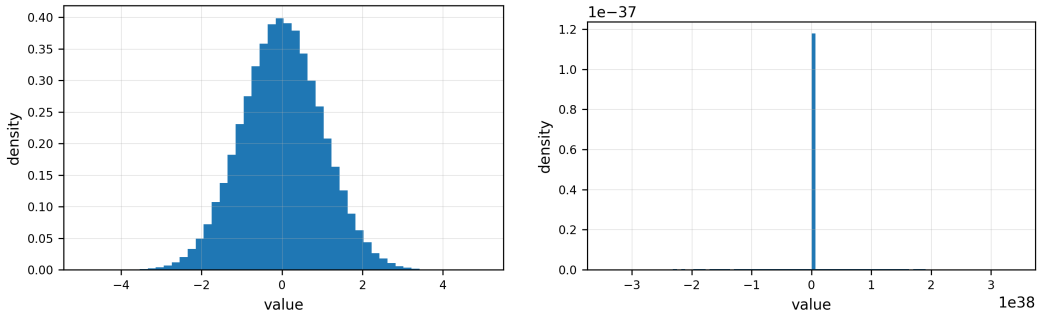
$$bff(x, i, n) = x \cdot 2^{2^{n-i-1}} \tag{2}$$

where n is a number of bits in the exponent, i is the position of the bit flip in the exponent.

To study aggregate effects, we apply (1) to float32 samples drawn from a standard normal distribution $x \sim \mathcal{N}(0, 1)$. Figure 2 shows the baseline distribution figure 2a alongside the corrupted one figure 2b, where 20% of the values have undergone a high-order exponent flip. The Gaussian bell shape is destroyed and the corrupted distribution exhibits pronounced heavy tails. An anomalous concentration of values near zero. These examples demonstrate a key property of SDC: high-order flips induce dramatic shifts that systematically distort statistical distributions.

2.2 DISTRIBUTED MODEL TRAINING

Distributed training refers to the process of training machine learning models across multiple computational units, often within one machine or across multiple machines in a cluster. The main objective is to accelerate training by leveraging the combined computational power and memory resources of multiple devices. There are many frameworks for distributed learning, such as Nagrecha (2023); Shoeybi et al. (2019); Narayanan et al. (2021); Korthikanti et al. (2022); Liu et al. (2023); Sergeev & Balso (2018), etc. However, with an increasing number of devices involved in training, the likelihood



(a) Original distribution (Standard Gaussian)

(b) Corrupted distribution after bit-flip

Figure 2: Distributional impact of a high-order bit-flip: (a) baseline $\mathcal{N}(0, 1)$ and (b) corrupted distribution after flipping the exponent bit in 20% of the samples.

of encountering hardware failures also grows, due to the added complexity and higher probability of component malfunctions across multiple units.

Even if a device appears healthy, it can still receive corrupted parameters during collective operations that merge outputs or gradients from all nodes in the cluster Ma et al. (2025). Because these updates are broadcast to and from every participant, a single node with silent data corruption can effectively contaminate otherwise healthy devices. This process unfolds through collective communication primitives (like all-reduce, all-gather, or reduce-scatter), which unify the outputs from all nodes and integrate them back into the shared model state. As a result, once an error is introduced in one device's computations, it can spread to the entire training cluster, even reaching nodes that were initially free of faults.

2.3 SDC IMPACT IN THE LEARNING PROCESS

Silent data corruption (SDC) typically manifests as bit-flips, i.e., random changes to individual bits in memory or during data transfers. He et al. (2023) investigated hardware failures in accelerator-based deep learning systems and found that training is most severely impacted by high-magnitude errors. Such errors can lead to a variety of outcomes, including:

- gradual degradation of accuracy,
- an abrupt accuracy drop followed by slow degradation,
- an instantaneous and severe accuracy collapse,
- training that appears normal but yields significantly reduced final accuracy.

These effects are often caused by transient faults or circuit degradation in accelerator logic. Some errors can be mitigated by scaling mechanisms inherent in normalization layers and optimizers that exploit gradient history, which partially compensates for numerical deviations and prevents their accumulation.

Nevertheless, detecting hardware-induced errors remains essential, as many can cause serious disruptions in the training process. Particularly critical are unrecoverable faults, such as sudden jumps producing NaN/INF values or corrupted weight updates, which can render the model unusable. Even errors that may eventually be compensated for by normalization or optimization still waste computational resources: they degrade convergence, temporarily reduce accuracy, and increase training cost. Timely detection and correction of such faults are therefore crucial for improving both the reliability and efficiency of large-scale training.

Liang et al. (2025) further studied hardware-induced faults in the attention mechanism of large language models, including BERT Devlin et al. (2018), GPT-2 Tan et al. (2020), GPT-Neo Kashyap et al. (2023), and RoBERTa Liu et al. (2019). The attention module is particularly vulnerable due to its computational complexity. Bit-flips in floating-point computations can introduce INF, NaN,

or near-INF values, which then propagate through key operations, such as matrix multiplication (GEMM), softmax, and normalization. This makes attention-related failures especially dangerous, as they can compromise the stability of the entire training pipeline.

3 Approach

3.1 Multimodality Emergence in Training under SDC Influence

Multimodality in the distribution of optimizer parameters or gradient statistics is often associated with transient phenomena during training and may indicate instability, such as sudden gradient spikes Molybog et al. (2023). Although such occurrences are relatively rare, they may also arise naturally from architectural factors (e.g., skip connections, multi-head attention) or optimization dynamics. Nevertheless, the unexpected appearance of multimodality provides a strong signal for detecting potential pathologies in the learning process.

Our approach focuses on analyzing the norms of the gradients after a logarithmic transformation, i.e., studying $\log(\|\nabla L\|)$. If the raw gradient norms form a mixture of values with substantially different scales, such as those induced by anomalously large updates due to SDC, the distribution may still appear unimodal or only weakly bimodal. Applying the logarithmic transformation amplifies these scale differences, causing the modes to separate and thereby making multimodality more pronounced.

Theorem 1 (Violation of Unimodality under High-Order Bit-Flip Corruption.) *Let X be a real-valued random variable with probability density function f_X and $\Pr(X = 0) = 0$. Assume that $Y = \log_2(|X|)$ is unimodal. Consider the corrupted variable G obtained by applying the bit-flip transformation equation 1 with probability $p \in (0, 1)$ in the exponent of a number with respect to IEEE 754:*

$$G = \begin{cases} \log_2(|X|) + \Delta, & \textit{with probability } p, \\ \log_2(|X|), & \textit{with probability } 1 - p, \end{cases}$$

where $\Delta = 2^{n-i-1}$ denotes the flipped bit position in the IEEE 754 representation of X. Then there exists a critical position $\Delta_{\mathrm{crit}}(p)$ such that for all $\Delta \geq \Delta_{\mathrm{crit}}(p)$, the distribution of G is multimodal.

A detailed proof of this theorem is provided in appendix B.1. The result formalizes the intuition that SDC, modeled as high-order bit-flips in floating-point values, inevitably creates distinct clusters in the log-transformed domain, thus violating unimodality. This provides a theoretical foundation for using multimodal behavior as a statistical indicator of anomalies during training. In this work, we used $\Delta_{crit} \approx 14.36$, which corresponds to the third bit of the exponent, i.e., we focused on detecting errors that occurred in the first three bits of the exponent. The conclusion can be found in appendix B.2.

Multimodality in gradient statistics does not necessarily imply the presence of SDC. Such patterns may also arise naturally from model dynamics, optimization artifacts, or data heterogeneity. Still, when the modes are well separated, i.e., when the inter-cluster distance is large, the emergence of multimodality can serve as a useful indicator of potential corruption. In practice, combining this signal with distance-based metrics further improves the reliability of SDC detection in large-scale training pipelines.

3.2 Diagnosing Multimodal Behavior in Training Dynamics

As established in the previous section, SDC can induce multimodality in the distributions of training-related statistics, such as gradient norms. Detecting such multimodal patterns in training dynamics therefore provides a valuable signal of potential hidden errors. Several statistical tests have been proposed for diagnosing multimodality, each offering different trade-offs in sensitivity, robustness, and computational cost. Classical approaches such as Hartigan's Dip Test Hartigan & Hartigan (1985) and Silverman's Test Silverman (1981) provide strong theoretical guarantees but are computationally expensive, often requiring bootstrapping. More recent methods, such as the UU-Test Chasani & Likas (2022), are robust to noise but less established in practice. On the other hand, the Folding Test Siffer et al. (2018) has been demonstrated to be effective when multimodality arises from symmetric deviations and has linear-time complexity, making it particularly well-suited for large-scale distributed training. A detailed comparison of these tests, including their advantages and

limitations, is provided in Appendix A. In the remainder of this work, we adopt the Folding Test as the basis for multimodality detection, as it provides the best balance between computational efficiency and detection reliability in our setting.

3.3 Metrics for distribution shift induced by gradient corruption

As established in theorem 1, high-order bit-flips caused by SDC produce additional modes in the distribution of gradient norms, with corrupted values displaced by several orders of magnitude from the clean ones. Such anomalies are meaningful only if the emerging modes are sufficiently separated, creating a measurable distributional shift rather than minor local fluctuations. To evaluate this effect, one requires a metric that captures both the presence of extra modes and the degree of their separation from the original distribution.

For this purpose, we adopt the Wasserstein distance Villani (2008) (Earth-Mover distance), which offers favorable theoretical and practical properties for detecting gradient corruption. In contrast to divergence-based measures such as Kullback - Leibler Kullback & Leibler (1951) or Jensen - Shannon Lin (1991), which may become undefined or unstable when distributions have disjoint support, the Wasserstein distance is a true metric. It measures the minimal transportation cost required to transform one distribution into another, and thus remains well defined when SDC shifts gradients into previously unoccupied regions.

Moreover, the Wasserstein distance is sensitive not only to the emergence of multiple modes but also to their displacement, capturing both local and global aspects of distributional divergence. This sensitivity is critical for distinguishing genuine corruption-induced shifts from benign variability in training dynamics.

For one-dimensional empirical distributions $P = \{x_1, \ldots, x_n\}$ and $Q = \{y_1, \ldots, y_n\}$, sorted in increasing order, the first-order Wasserstein distance is given by

$$W_1(P, Q) = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \,, \tag{3}$$

which can be computed in linear time after sorting. This efficiency makes the Wasserstein distance particularly suitable for online diagnostics of distributional shifts during large-scale training.

3.4 Construction of the SDC detection pipeline

To detect SDC during training, we propose a lightweight statistical pipeline based on the analysis of gradient norms. Algorithm 1 describes the entire pipeline of the SDC detection. The detection process consists of three stages:

1. **Logarithmic transformation.** For a batch of gradients $G = \{g_1, g_2, \ldots, g_m\}$, with $g_i \in \mathbb{R}^n$, we compute
$$V = \{\log \|g_1\|, \log \|g_2\|, \ldots, \log \|g_m\|\} \in \mathbb{R}^m.$$
This step normalizes scale differences and enhances separation between potential modes.
2. **Unimodality assessment.** The distribution of V is evaluated with the Folding Test. If the test accepts unimodality, the batch is considered clean. Complexity: $O(m)$.
3. **Intermodal separation.** If multimodality is detected, the Folding Test provides a pivot point $s^* \in \mathbb{R}$ between the modes. We partition
$$V_{\text{left}} = \{v_i \in V : v_i \leq s^*\}, \quad V_{\text{right}} = \{v_i \in V : v_i > s^*\},$$
and compute the Wasserstein distance
$$W_1(V_{\text{left}}, V_{\text{right}}) = \int |F_{V_{\text{left}}}(x) - F_{V_{\text{right}}}(x)| dx,$$
where $F_{V_{\text{left}}}$, $F_{V_{\text{right}}}$ are cumulative density functions for V_{left} and V_{left} correspondingly. If $W_1 > \tau$, with τ an empirically chosen threshold, the shift is deemed significant and SDC is flagged.

Each stage is computationally inexpensive ($O(m \log m)$ overall, dominated by sorting for Wasserstein distance), enabling integration into large-scale distributed training with negligible overhead.

Algorithm 1 SDC detection pipeline

Require: Gradients $G = \{g_1, \ldots, g_m\}$, threshold τ
Ensure: Detection flag
$V \leftarrow \{\log \|g_1\|, \ldots, \log \|g_m\|\}$
$(s^*, \text{is\_unimodal}) \leftarrow \texttt{FoldingTest}(V)$
if `is_unimodal` **then**
    return `No SDC detected`
end if
Partition V into $V_{\text{left}} = \{v_i < s^*\}$, $V_{\text{right}} = \{v_i > s^*\}$
$d \leftarrow W_1(V_{\text{left}}, V_{\text{right}})$
if $d > \tau$ **then**
    return `SDC detected`
else
    return `No SDC detected`
end if

4 Implementation

The proposed SDC detection pipeline was implemented on a cluster of 8 Huawei Ascend 910B devices. Each device combines specialized processing units, including AI Cores for dense linear algebra, AI CPUs for control logic, and dedicated memory subsystems. The overall processor organization is illustrated in Figure 3.

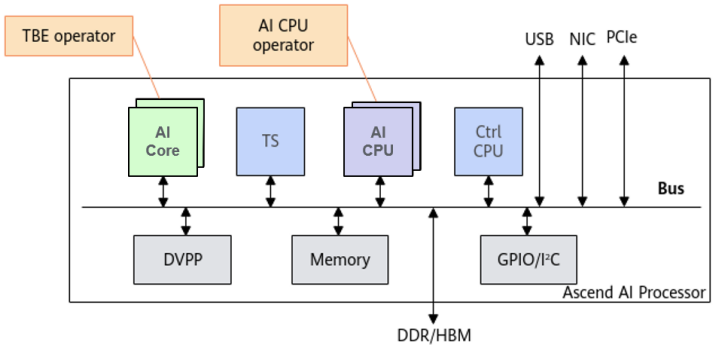


Figure 3: placement of the custom CANN operator for SDC detection within the Ascend AI processor.

In our implementation, computational tasks were distributed between the AI Core and AI CPU according to their architectural strengths:

- **AI Core.** The computation of logarithmic gradient norms was executed on the AI Core using custom Tensor Boost Engine (TBE) kernels, leveraging its high-throughput vector and matrix processing capabilities.
- **AI CPU.** Multimodality detection (Folding Test) and intermodal Wasserstein distance estimation were performed on the AI CPU. These operations involve control logic, for which the AI CPU is better suited.

The AI CPU operations were scheduled asynchronously and overlapped with AI Core execution, enabling pipelined processing and minimizing additional latency. This division of labor ensures that both dense numeric kernels and control-heavy statistical tests are executed on hardware optimized for their respective workloads.

To minimize integration effort, the SDC checker was embedded at the communication layer of the `torch_npu` adapter. Specifically, checks were inserted immediately before inter-device communication, allowing detection of corrupted tensors prior to their propagation across nodes. This

design avoids any modification to user-level training code while ensuring compatibility with existing distributed training pipelines.

5 EXPERIMENTS

5.1 FAULT INJECTOR

To evaluate the robustness of our SDC detection method under controlled conditions, we implemented a custom fault injection simulator. The simulator introduces single- and multi-bit flips into floating-point values across different model states. This design enables a systematic assessment of the sensitivity of the detection pipeline to diverse corruption patterns. The pseudocode of the bit-flip injection procedure is shown in Algorithm 2.

Faults were injected by modifying the byte-level representation of numerical values, flipping the specified bit positions, and reconstructing the corrupted values. In our experiments, bit positions were selected uniformly at random from the most significant portion of the exponent field. This choice reflects the fact that high-order bit flips have the strongest impact on numerical magnitude, often shifting values by many orders. Such errors are thus most likely to induce distributional shifts detectable by our method, whereas low-order flips typically behave as minor noise and do not materially affect training dynamics.

To ensure reproducibility, injections were performed on complete model dumps obtained from training checkpoints. This allowed us to target precise numerical values within the model state and emulate realistic corruption scenarios during training. The resulting setup provides a controlled yet representative environment for evaluating both the accuracy and computational overhead of the proposed SDC detection pipeline.

Algorithm 2 Bit-flip fault injection algorithm

Require: Numerical value v, bit positions $\mathcal{P}$
Ensure: Corrupted numerical value v'
1: Compute byte representation $B \leftarrow \text{bytes}(v)$
2: **for** each position $p \in \mathcal{P}$ **do**
3: Identify byte index q and bit index r: $q, r \leftarrow \text{divmod}(p, 8)$
4: Flip bit r of byte q: $B[q] \leftarrow B[q] \oplus (1 \ll r)$
5: **end for**
6: Reconstruct corrupted value v' from modified bytes B
7: **return** v'

5.2 RESULTS

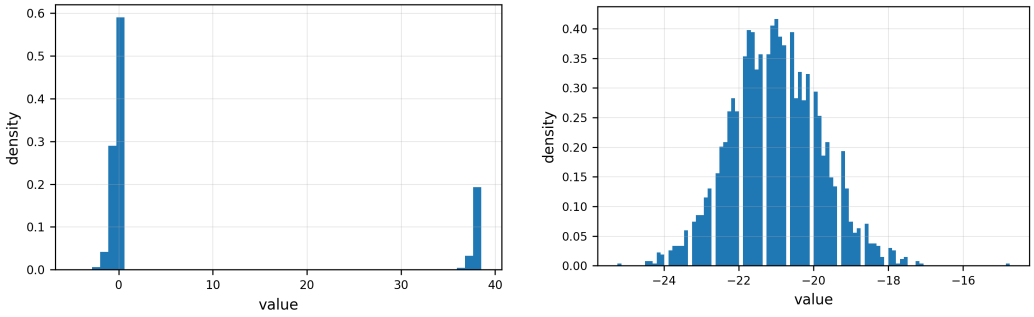
We evaluate the detector during LLaMA2 - 7B training under controlled single-bit exponent flips. We inject 300 faults in total, uniformly across three flip types (100 each), where type-1/2/3 means flipping the 1st/2nd/3rd bit of the IEEE 754 exponent, respectively. Table 1 reports the per-type breakdown at a fixed operating point ($\tau = 3$ for the Wasserstein test): overall, 297/300 faults are detected (99.0% TPR) with zero false positives; type-1/2 reach 100% TPR, and all three misses come from type-3. The false negatives arise because:

- the perturbation is attenuated before the pre-collective check by optimizer dynamics and normalization/residual mixing,
- the residual shift at test time remains within natural stochastic variability no clear multi-modality or a Wasserstein distance below τ.

The resulting time overhead is $\sim 0.46\%$ (86,890.60 ms per iteration for baseline vs. 87,294.26 ms with the detector). Figure 4 illustrates the distributional signal used by the detector: corrupted batches exhibit separated clusters, while missed cases stay within the baseline envelope.

Table 1: Per-type SDC detection on LLaMA2 - 7B (n=100 per type; total 300) at fixed threshold τ=3.

Bit-flip type	Injected	Detected (TP)	False Pos.	False Neg.	TPR
Type-1	100	100	0	0	100.0%
Type-2	100	100	0	0	100.0%
Type-3	100	97	0	3	97.0%
Total	**300**	**297**	**0**	**3**	**99.0%**



(a) Corrupted gradient norms

(b) Non-corrupted gradient norms

Figure 4: Gradient norm distributions with and without injected bit-flips.

6 Related works

Recent research has highlighted the critical challenges posed by hardware failures and silent data corruption (SDC) in large-scale deep neural network (DNN) and large language model (LLM) training. Ma et al. Ma et al. (2025) provided one of the first systematic investigations, showing that bit-flips at critical gradient or activation stages can lead to subtle but irreversible parameter drift. Their lock-step synchronization mechanism partially mitigates these effects, but at the cost of tight coupling between training processes.

He et al. He et al. (2023) conducted large-scale fault injection studies on accelerator hardware, demonstrating that even small defects in memory or logic pathways can propagate through linear algebra operations and compromise model accuracy. They further showed that naive checkpointing becomes prohibitively expensive under frequent failures, motivating the search for lightweight alternatives.

At the system level, Wu et al. Wu et al. (2023) proposed the TRANSOM framework, which combines adaptive checkpointing, anomaly detection, and on-the-fly rescheduling to manage node-level faults. While effective in large clusters, this approach requires extensive monitoring infrastructure and incurs non-negligible coordination overhead.

Liang et al. Liang et al. (2025) focused on the transformer attention mechanism and developed *ATTNChecker*, a submodule-level method that detects INF/NaN values and applies algorithm-based fault tolerance (ABFT) directly within matrix multiplications. Their approach achieves high accuracy in intercepting catastrophic errors, but remains specialized to a single model component and does not address broader distributional corruptions.

Taken together, prior work highlights two dominant directions: submodule-specific resilience mechanisms (e.g., synchronization, ABFT) and system-level frameworks (e.g., dynamic checkpointing, scheduling). Both classes of methods can be effective, but they either impose substantial runtime overhead or require intrusive architectural modifications. In contrast, our method operates at the communication boundary, introducing a lightweight statistical detector that requires no model-specific

changes and incurs only $\sim$2% overhead. This positions our approach as a complementary strategy that bridges the gap between fine-grained local protection and global system-level recovery.

7 Conclusion

We presented a lightweight statistical pipeline for detecting silent data corruption (SDC) during large-scale neural network training. By combining logarithmic transformation of gradient norms, unimodality assessment via the Folding Test, and intermodal separation with the Wasserstein distance, our method enables efficient online detection of corruption events. A key advantage of this approach is its integration at the communication layer, which allows faulty devices to be localized without modifying model code or disrupting the training pipeline. Experimental evaluation on LLaMA2 - 7B demonstrated a 99% detection rate with no false positives and only $\sim$2% runtime overhead.

At the same time, the current study has several limitations. First, the decision threshold for the Wasserstein distance could not be derived analytically and was instead determined empirically (set to $\tau = 3$ for LLaMA2-7B). Second, small-magnitude bit-flips, which do not significantly affect model convergence, remain outside the scope of our detection. Finally, evaluation was limited to a single model architecture; broader validation is required to assess generality.

Future work will explore adaptive thresholding strategies, integration with complementary resilience mechanisms (e.g., algorithm-based fault tolerance in attention layers), and experiments across diverse architectures and training scales. Together, these directions aim to extend the robustness and applicability of the proposed approach, providing a foundation for resilient large-scale training under increasingly heterogeneous and failure-prone hardware environments.

References

Paraskevi Chasani and Aristidis Likas. The uu-test for statistical modeling of unimodal data. *Pattern Recognition*, 122:108272, February 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108272. URL http://dx.doi.org/10.1016/j.patcog.2021.108272.

Cristian Constantinescu, Ishwar Parulkar, Rick Harper, and Sarah Michalak. Silent data corruption — myth or reality? In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pp. 108–109, 2008. doi: 10.1109/DSN.2008.4630077.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Gemini Team. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805. v5, May 9, 2025.

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 20–43, Bolton Landing, NY, 2003.

John A Hartigan and Pamela M Hartigan. The dip test of unimodality. *The Annals of Statistics*, 13(1): 70–84, 1985. doi: https://doi.org/10.1214/aos/1176346577.

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.

Rohan Kashyap, Vivek Kashyap, and Narendra C. P. Gpt-neo for commonsense reasoning – a theoretical and practical lens, 2023. URL https://arxiv.org/abs/2211.15593.

Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models, 2022. URL https://arxiv.org/abs/2205.05198.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. doi: 10.1214/aoms/1177729694.

Yuhang Liang, Xinyi Li, Jie Ren, Ang Li, Bo Fang, and Jieyang Chen. Attnchecker: Highly-optimized fault tolerant attention for large language model training, 2025. URL https://arxiv.org/abs/2410.11720.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Yuliang Liu, Shenggui Li, Jiarui Fang, Yanjun Shao, Boyuan Yao, and Yang You. Colossal-auto: Unified automation of parallelization and activation checkpoint for large-scale models, 2023. URL https://arxiv.org/abs/2302.02599.

Jeffrey Ma, Hengzhi Pei, Leonard Lausen, and George Karypis. Understanding silent data corruption in llm training, 2025. URL https://arxiv.org/abs/2502.12340.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, Binh Tang, Diana Liskovich, Puxin Xu, Yuchen Zhang, Melanie Kambadur, Stephen Roller, and Susan Zhang. A theory on adam instability in large-scale machine learning, 2023. URL https://arxiv.org/abs/2304.09871.

Kabir Nagrecha. Systems for parallel and distributed large-model deep learning training, 2023. URL https://arxiv.org/abs/2301.02691.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021. URL https://arxiv.org/abs/2104.04473.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow, 2018. URL https://arxiv.org/abs/1802.05799.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019. URL http://arxiv.org/abs/1909.08053.

Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouët. Are your data gathered? In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2210–2218. ACM, 2018. doi: https://doi.org/10.1145/3219819.3219994.

B. W. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society: Series B (Methodological)*, 43(1):97–99, 1981. URL https://www.jstor.org/stable/2985156.

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric P. Xing, and Zhiting Hu. Progressive generation of long text. *CoRR*, abs/2006.15720, 2020. URL https://arxiv.org/abs/2006.15720.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Shruti Bhosale, and et al. Llama 2: Open foundation and fine-tuned chat models. Meta AI technical report, 2023. https://ai.meta.com/llama.

Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2008. doi: 10.1007/978-3-540-71050-9.

Baodong Wu, Lei Xia, Qingping Li, Kangyu Li, Xu Chen, Yongqiang Guo, Tieyao Xiang, Yuheng Chen, and Shigang Li. Transom: An efficient fault-tolerant system for training llms, 2023. URL https://arxiv.org/abs/2310.10046.

A Comparison of Multimodality Tests

Table 2 summarizes several commonly used statistical tests for detecting multimodality. This extended comparison complements the discussion in Section 3.2, where the Folding Test was chosen as the most suitable option for our setting.

Table 2: comparison of statistical tests for multimodality detection

test name	**description**	**advantages**	**disadvantages**	**complexity**
Hartigan's Dip Test Hartigan & Hartigan (1985)	Measures maximum deviation of the empirical CDF from the closest unimodal distribution.	General-purpose; parameter-free; widely used.	Sensitive to noise; bootstrapping needed for p-values.	$\mathcal{O}(n \log n)$
Silverman's Test Silverman (1981)	KDE + bootstrapping to test the number of modes.	Highly sensitive; can test for a specific number of modes.	Bandwidth/kernel dependent; computationally expensive.	$\mathcal{O}(n^2 + Bn)$
Folding Test Siffer et al. (2018)	Compares variance of the "folded" distribution to the original.	Fast; simple; effective for symmetric deviations.	Less sensitive to asymmetry; no p-value.	$\mathcal{O}(n)$
UU-Test Chasani & Likas (2022)	Approximates the empirical CDF with uniform segments.	No bootstrapping; parameter-free; robust to noise.	Requires sorting; relatively new/less validated.	$\mathcal{O}(n \log n)$

B Proofs

B.1 Proof of the Theorem 1

Representation of the density G. By the law of total probability, the density f_G is a mixture of two components:

$$f_G(z) = (1-p)f_Y(z) + pf_Y(z-\Delta), \quad z \in \mathbb{R}.$$

Both functions $f_Y(z)$ and $f_Y(z-\Delta)$ are continuous and differentiable; therefore, f_G is also continuous and differentiable everywhere.

Sign of the derivative around the mode of the original density. Let m be the (unique) mode of f_Y. By unimodality we have

$$f'_Y(z) > 0 \quad \text{for } z < m, \quad f'_Y(m) = 0, \quad f'_Y(z) < 0 \quad \text{for } z > m.$$

Consider the derivative of the mixture:

$$f'_G(z) = (1-p)f'_Y(z) + pf'_Y(z-\Delta).$$

Now compute f'_G at two key points, $z = m$ and $z = m + \Delta$.

At $z = m$:

$$f'_G(m) = (1-p)f'_Y(m) + pf'_Y(m-\Delta) = 0 + pf'_Y(m-\Delta).$$

Since $m - \Delta < m$, by unimodality we have $f'_Y(m-\Delta) > 0$. Thus,

$$f'_G(m) = pf'_Y(m-\Delta) > 0.$$

At $z = m + \Delta$:

$$f'_G(m+\Delta) = (1-p)f'_Y(m+\Delta) + pf'_Y(m) = (1-p)f'_Y(m+\Delta) + 0.$$

Since $m + \Delta > m$, we have $f'_Y(m+\Delta) < 0$, hence

$$f'_G(m+\Delta) < 0.$$

Therefore, $f'_G(m) > 0$ and $f'_G(m+\Delta) < 0$. Therefore, $f'_G(m) > 0$ and $f'_G(m+\Delta) < 0$.

Since f'_G is continuous, the change of sign between m and $m+\Delta$ guarantees the existence of a point

$$c \in (m, m+\Delta) \quad \text{such that } f'_G(c) = 0.$$

Thus, inside the interval $(m, m+\Delta)$ there exists a stationary point of f_G.

Second derivative analysis and threshold $\Delta_{\text{crit}}(p)$. Let z_0 be a stationary point of f_G, i.e. $f'_G(z_0) = 0$. The Taylor expansion of f_G about z_0 reads

$$f_G(z) = f_G(z_0) + \tfrac{1}{2} f''_G(z_0)(z-z_0)^2 + \tfrac{1}{3!} f^{(3)}_G(z_0)(z-z_0)^3 + \tfrac{1}{4!} f^{(4)}_G(z_0)(z-z_0)^4 + \cdots .$$

The leading second-order term determines the local quadratic behaviour:

- If $f''_G(z_0) < 0$ and higher-order terms are negligible, then z_0 is a (strict) local maximum and the graph near z_0 resembles an inverted parabola.
- If $f''_G(z_0) > 0$, then z_0 is a local minimum.
- If $f''_G(z_0) = 0$, the quadratic term vanishes and the local shape is governed by higher-order terms. For example:
  - If $f^{(3)}_G(z_0) \neq 0$, the cubic term dominates and z_0 is an inflection point with asymmetric cubic behaviour.
  - If $f^{(3)}_G(z_0) = 0$ but $f^{(4)}_G(z_0) < 0$, then the quartic term dominates and a flattened local maximum/plateau may appear (typical for symmetric profile around z_0).

Definition of the critical shift Δ_{crit}. A natural criterion for the loss of unimodality is the emergence of a stationary point $z^* \in (m, m+\Delta)$ at which the second derivative changes sign from non-positive to non-negative. Thus we define $\Delta_{\text{crit}}(p)$ as the smallest shift $\Delta > 0$ for which there exists a point $z^* \in (m, m+\Delta)$ such that

$$f'_G(z^*) = 0 \quad \text{and} \quad f''_G(z^*) = 0.$$

The first condition identifies z^* as a stationary point; the second marks the transition in the character of the stationary point (a bifurcation boundary between "no interior minimum" and "interior minimum present").

When $\Delta < \Delta_{\text{crit}}(p)$ any stationary point inside $(m, m+\Delta)$ is either absent or corresponds to a point where $f''_G \leq 0$, so the mixture remains effectively unimodal (the peaks overlap). For $\Delta > \Delta_{\text{crit}}(p)$ the stationary point splits into a strict local minimum flanked by two distinct local maxima (one near m, the other near $m+\Delta$), i.e. the mixture becomes multimodal. Thus, the theorem is proved.

B.2 Conclusion Δ_{crit} for the case of normal distribution

We consider that $|X|$ has like lognormal distribution. And $Y = \log_2(|X|)$ is normal distribution then:

$$\phi_\sigma(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{-y^2}{2\sigma^2}}.$$

Then:

$$\phi^{'}(y) = -\frac{y}{\sigma^2}\phi(y), \quad \phi^{''}(y) = \frac{y^2-\sigma^2}{\sigma^4}\phi(y).$$

The condition for the critical point y^* is determined by the system:

$$\begin{cases} (1-p)\phi^{'}(y^*) + p\phi^{'}(y^*-\Delta) = 0, \\ (1-p)\phi^{''}(y^*) + p\phi^{''}(y^*-\Delta) = 0. \end{cases}$$

From the first we obtain:

$$\phi(y^*-\Delta) = \frac{(1-p)y}{p(\Delta-y^*)}\phi(y^*).$$

From the second we obtain:

$$(1-p)({y^*}^2-\sigma^2)\phi(y^*) + p((y^*-\Delta)^2-\sigma^2)\phi(y^*-\Delta) = 0.$$

After expanding the brackets and simplifying some terms, we obtain:

$$(1-p)\left(-\frac{y^*}{\sigma^2}\phi(y^*)\right)+p\left(-\frac{y^*-\Delta}{\sigma^2}\phi(y^*-\Delta)\right)=0$$
$$(1-p)y^*\phi(y^*)-p(\Delta-y^*)\phi(y^*-\Delta)=0$$
$$\frac{(1-p)y^*}{p(\Delta-y^*)}=\frac{\phi(y^*-\Delta)}{\phi(y^*)}.$$

Calculate the ratio $\frac{\phi(y^*-\Delta)}{\phi(y^*)}$. Since $\phi(y) \propto e^{-\frac{y^2}{2\sigma^2}}$, we have:

$$\frac{\phi(y^*-\Delta)}{\phi(y^*)}=\exp\left(-\frac{(y^*-\Delta)^2}{2\sigma^2}+\frac{y^{*2}}{2\sigma^2}\right)=\exp\left(\frac{y^{*2}-(y^*-\Delta)^2}{2\sigma^2}\right)=$$
$$=\exp\left(\frac{2z^*\Delta-\Delta^2}{2\sigma^2}\right).$$

Then:

$$\ln\left(\frac{(1-p)y^*}{p(\Delta-y^*)}\right)=\frac{y^*\Delta-\frac{1}{2}\Delta^2}{\sigma^2}.$$

As a result, we obtain that $\Delta_{crit}(p,\sigma)$ is defined as the minimum $\Delta \geq 2\sigma$ such that:

$$\ln\left(\frac{(1-p)y^*}{p(\Delta-y^*)}\right)=\frac{y^*\Delta-\frac{1}{2}\Delta^2}{\sigma^2},$$

when $y^*=\frac{\Delta\pm\sqrt{\Delta^2-4\sigma^2}}{2}$ and p is given. Figure 5 shows the values of Δ_{crit} depending on p.

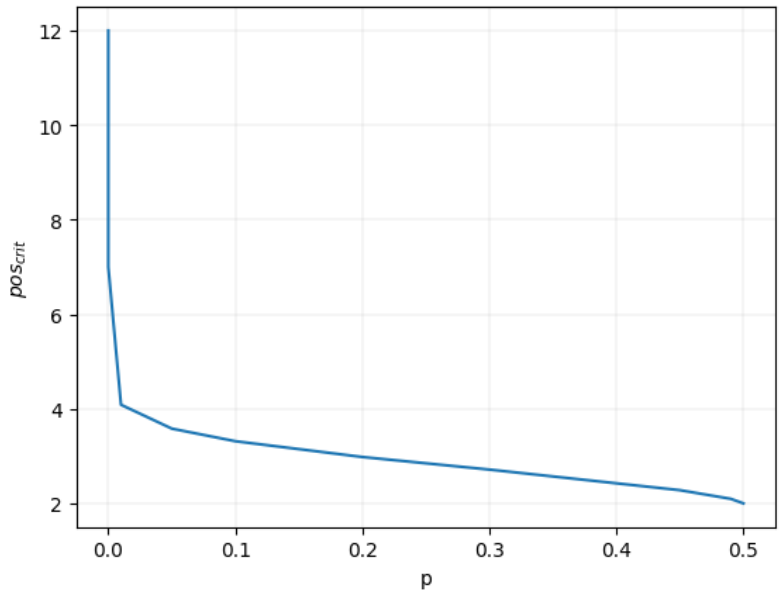


Figure 5: Dependence of Δ_{crit} on p when σ=1

.

In this work, we use $\Delta_{crit}=14.36$ because the probability of a bit-flip occurring is not that high in reality, and we have derived a theoretical upper limit, but we also check communications, and before the error reaches the communication, it will have time to smooth out. From equation 2, this corresponds to the third bit of the exponent.

We examined various tensors at $p=1e^{-10}$ for which we calculated Δ_{crit} and **position** $= n-1-\log_2\Delta_{crit}$; the results are shown in the table 3.

Table 3: Δ_{crit} for various tensors of the LLaMA2 - 7B model for $p = 1e^{-10}$

σ	Δ_{crit}	**position**	**shapes**
2.42	18.14	2.82	$1024 \times 4 \times 2048$
1.99	14.92	3.10	2560×2560
1.72	12.87	3.31	$4096 \times 2 \times 480$
1.81	13.57	3.24	$4096 \times 2 \times 480$
1.79	13.40	3.26	$4096 \times 2 \times 480$
1.77	13.28	3.27	$4096 \times 2 \times 480$

C USE OF LARGE LANGUAGE MODELS (LLMS)

Writing assistance. We used a large language model (OpenAI ChatGPT) strictly as a copy-editing aid to improve grammar, clarity, concision, and consistency of terminology. No mathematical statements, algorithms, experimental designs, or conclusions were authored by the LLM; all technical content and structure originated from the authors.

Retrieval and discovery. We used the LLM to help discover and organize related work by generating candidate paper lists and topical groupings from our queries. All suggested references were independently verified by the authors for accuracy and relevance, and any unverifiable citations were discarded. Final literature selection and summaries were written and vetted by the authors.

Accountability and authorship. The authors take full responsibility for the paper's contents. The LLM is not an author and did not contribute intellectual novelty or research ideation.

Data handling. We did not share proprietary data, unpublished results, or personal information with the LLM beyond brief, high-level paraphrases necessary for editing. All experimental results, analyses, figures, and tables were produced and checked by the authors.

Plagiarism and factual checks. LLM-edited text was reviewed line-by-line by the authors. Technical claims, equations, and citations were verified against primary sources and our own experiments.