Silent Data Corruption Detection Based on Metamorphic Attention Bounds

Xinyu Bai
University of Illinois Urbana-Champaign
xbai@illinois.edu

Abstract

Silent data corruption (SDC) is increasingly recognized as a dominant reliability threat in large-scale systems and machine learning workloads running on GPUs. While a significant fraction of SDCs originate from repeatable device and process defects rather than purely transient faults, production systems still lack practical mechanisms to detect these errors during regular operation [1, 2, 3]. At the same time, modern transformer-based models heavily rely on attention mechanisms, whose highly structured computations offer an opportunity to construct metamorphic relations between inputs and outputs.

This report explores the feasibility of detecting GPU SDCs in attention layers by leveraging analytically derived bounds as metamorphic relations. We first review the landscape of SDC on CPUs, GPUs and emerging large language model (LLM) workloads [4, 6, 7, 8]. We then derive simple upper and lower bounds for standard self-attention and discuss their complexity characteriztics (derivation deferred to the appendix). On top of these bounds, we design injection campaigns that flip single bits in intermediate attention tensors and examine whether corrupted outputs violate the bounds.

Our preliminary experiments start from single-layer attention with synthetic Gaussian inputs, extend to a minimal transformer with character-level data, and finally target GPT-2 layers on WikiText. The results indicate that (i) detection coverage depends strongly on the flipped bit position, with higher mantissa/exponent bits being most informative; (ii) forcing $K = V$ in the tested layer yields better detection than a straightforward generalization of the theoretical bounds; and (iii) combining two mathematically equivalent views of the attention output—$Q \cdot \mathrm{Attn}(X)$ vs. $QK^\top \cdot \mathrm{Softmax}(\cdot)$—improves coverage, when we restrict to faults that flip exponent and sign bits (bits 23–31), to around 24.7% (568 detections out of 2304 injected faults) in our GPT-2 experiments. We close by outlining directions for tightening the bounds, supporting masked attention and backpropagation, and integrating the approach with realistic GPU fault simulators.

1 Introduction

Silent data corruption (SDC)—incorrect computation results that occur without any explicit error signal—has become a first-order reliability concern in large-scale computing systems. Recent field studies from hyperscale operators show that many SDCs are rooted not only in transient radiation-induced faults, but also in manufacturing defects and device aging that cause repeatable miscomputations in specific processors or cores [1, 3]. These corruptions are particularly insidious: they may bypass traditional error-correcting mechanisms, escape logging, and silently contaminate long-running workloads. At the same time, modern AI services rely heavily on accelerators and large-scale distributed training, further amplifying the impact of rare faults. Industry reports from TPU and CPU fleets [2, 5] emphasize that existing test flows leave gaps that allow SDCs to slip into production.

As computing shifts towards accelerator-dominated workloads, understanding and mitigating SDCs in GPU-based applications has become increasingly important. Prior work has characterized the SDC vulnerability of GPUs under various GPGPU workloads [7], and analytic models have been proposed to estimate SDC rates by composing microarchitectural and application-level views of faults [4, 6]. More recently, Ma et al. [8] provided empirical evidence that SDCs in large language model (LLM) training, even when numerically small, can alter optimization trajectories and cause occasional spikes in training loss. These studies collectively show that: (i) SDCs do occur in practice at non-negligible rates, (ii) their manifestation depends strongly on the workload and hardware stack, and (iii) LLM and other deep-learning workloads are both long-running and structurally repetitive, making them vulnerable to silent miscomputations.

In this work we focus on a specific but central building block of modern deep-learning models: the attention mechanism. Standard dot-product attention has a highly structured algebraic form and is instantiated repeatedly across layers and heads in transformer architectures. This structure suggests the possibility of application-level consistency checks that operate at the level of attention tensors rather than raw hardware events. Our guiding question is:

Can we use the mathematical structure of attention to define low-cost metamorphic relations that detect a useful subset of SDCs in practice?

Instead of attempting to monitor the entire GPU stack or to derive global SDC rates, we ask whether analytically derived bounds on attention outputs can serve as lightweight oracles: if the observed outputs of an attention layer violate these bounds under controlled perturbations, we flag a potential silent corruption. Our goal is exploratory and feasibility-oriented rather than to propose a ready-to-deploy production system.

To investigate this question, we build a prototype framework that wraps attention layers with a fault-injection and checking harness. We inject single-bit flips into intermediate attention tensors and test whether the resulting outputs remain consistent with simple bounds derived from the self-attention formulation. Starting from toy setups and progressing to realistic models, we aim to understand when such metamorphic relations are informative, how their detection power depends on fault location and magnitude, and what overhead they introduce.

Concretely, this report makes the following contributions:

- **Metamorphic bounds for self-attention.** We derive simple upper and lower bounds on the outputs of standard dot-product self-attention under a mild assumption ($K = V$ in the tested layer), and we discuss the computational complexity of evaluating these bounds. The detailed derivations are deferred to the appendix.
- **Prototype and empirical study.** We implement a fault-injection and checking harness around attention layers, and conduct experiments in three settings: (i) a single-layer attention module with synthetic Gaussian inputs, (ii) a minimal transformer on character-level data, and (iii) GPT-2 layers evaluated on WikiText. Across these setups, we analyze how detection coverage depends on bit position, model structure, and input distribution.
- **Experimental findings and implications.** Our preliminary results show that detection coverage is dominated by flips in high-order bits, that forcing $K = V$ in the tested GPT-2 layer yields better coverage than a straightforward generalization of the bounds to $K \neq V$, and that combining two mathematically equivalent forms of the attention computation leads to complementary detections, reaching around 25.6% coverage (524 detections out of 2048 injected faults) in one GPT-2 configuration. These results suggest that metamorphic relations for attention, while far from complete, can detect a non-trivial subset of SDCs and could be refined further with tighter bounds and better integration with fault simulators.

2 Background and Related Work

We next summarize prior work on silent data corruption (SDC) in accelerators and on metamorphic testing, and explain how we apply these ideas to attention mechanisms.

2.1 Silent Data Corruptions at Scale in CPU-Based Fleets

Dixit et al. [1] present one of the first large-scale field studies of CPU SDCs in a hyperscale production environment. Drawing on more than eighteen months of monitoring across hundreds of thousands of machines, they characterize defect types in silicon manufacturing that lead to SDCs, walk through a real-world case study of debugging an application-level manifestation, and argue that both hardware resiliency and fault-tolerant software architectures are required to keep SDC rates acceptable at fleet scale. Follow-up work from the same group focuses on the detection side: Dixit et al. [5] compare two complementary approaches to screening for CPU SDCs in production—Fleetscanner, an out-of-production testing campaign, and Ripple, an in-production testing system—and discuss the infrastructure trade-offs they observed over three years of experience.

Complementing these Meta case studies, Bonderson's invited talk "Training in Turmoil: Silent Data Corruption in Systems at Scale" [2] frames SDC as a "bugs-from-hell" problem for large-scale machine learning training on TPUs. The talk argues that traditional ATE and functional testing are insufficient as silicon process technology scales and system sizes grow, and calls for new silicon lifecycle management strategies to close test gaps and contain the quality and cost impact of SDCs. Wang et al. [3] further contribute a quantitative perspective: by analyzing SDC incidents in a large production CPU population, they quantify SDC rates, characterize failure modes across microarchitectures and process nodes, and draw implications for both processor design and system-level reliability modeling.

Taken together, these works establish that SDC is a systemic issue in large-scale CPU fleets, that it is not adequately explained by transient radiation alone, and that effective mitigation is likely to require both hardware and application-level support.

2.2 Modeling and Propagation of Silent Data Corruptions

While the above works focus on field observations and engineering mitigations, other lines of work study how SDCs propagate through applications and how to estimate their rates. Calhoun et al. [4] investigate SDC propagation in sparse linear algebra kernels and iterative solvers, showing that where and how a single-bit corruption is injected can drastically change whether the resulting error is masked, triggers a detectable failure, or leads to an application-level SDC. Their results highlight that naïve single-level fault-injection campaigns can misestimate SDC behavior and motivate more structured modeling.

Hari et al. [6] propose a two-level methodology for estimating SDC rates. The first level models how low-level faults (e.g., particle strikes) in unprotected hardware state manifest as architecture-level corruptions; the second level measures how these architecture-level manifestations propagate to program outputs. Evaluated on GPU architectures, their work demonstrates that using only one of the two levels can significantly overestimate or mis-trend SDC rates, and that composing the two levels yields more accurate and actionable reliability estimates.

These modeling efforts make it clear that SDC behavior depends not just on hardware fault rates but also on how specific applications transform and amplify corrupted values, which motivates application-aware detection strategies.

2.3 GPU and AI Workloads Under SDC

Compared to CPUs, GPUs introduce different protection mechanisms, programming models, and workload characteristics, which can change how often faults manifest as SDCs. Vishwanathan et al. [7] characterize the SDC vulnerability of GPUs under a range of GPGPU workloads. By combining error injection with application-level observation, they show that SDC vulnerability is highly workload-dependent and that general-purpose GPU use in safety- or correctness-critical settings requires careful resilience evaluation.

More recently, SDC has begun to attract attention in the context of deep learning accelerators and large-scale AI training. Ma et al. [8] are the first to empirically study real-world SDCs in large language model (LLM) training. Using access to unhealthy nodes that were swept out of production by cloud fleet management due to SDCs, and enforcing deterministic execution via XLA, they compare healthy and SDC-affected nodes at three levels: individual submodule computations (e.g., attention blocks), single optimizer steps, and long training runs. Their experiments reveal that, while perturbations at the tensor level are often small, SDCs can steer optimization towards different optima and occasionally produce sharp spikes in training loss, underscoring the need for targeted detection and mitigation mechanisms in LLM training systems.

These results suggest that for accelerator-heavy AI workloads, effective SDC detection is unlikely to come from hardware mechanisms alone. Instead, detection techniques will need to be aware of the structure of the workloads they protect. In the case of transformers, this naturally leads to attention modules as structured candidates for application-level checking.

2.4 Metamorphic Testing and Attention Mechanisms

Metamorphic testing is a technique for testing programs in the presence of oracle problems by checking relations between multiple executions rather than individual outputs [10, 11]; when a program's outputs violate such a relation, a defect is suspected.

For numerical and machine-learning workloads, metamorphic relations are often constructed via *input transformations*. Examples include scaling or translating inputs within a range that should preserve certain invariants, permuting inputs in ways that should not affect aggregated results, or injecting small, bounded perturbations that should not change qualitative behavior. Compared to duplicating computation or relying on full ground-truth labels, input-transformation relations can be cheap to instantiate and can be tailored to the structure of a specific operation.

Attention mechanisms in transformers exhibit significant algebraic structure; in particular, standard dot-product self-attention [9] can be written schematically as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{1}$$

where Q, K, and V are the query, key, and value matrices, respectively. This structure suggests several families of metamorphic relations, including:

- **Input transformations:** for example, scaling inputs or adding bounded perturbations along dimensions that should not change the relative attention pattern, and then requiring the resulting outputs to remain within analytically derived bounds.

- **Equivalent representations:** relating different but mathematically equivalent ways of computing the same output (e.g., factoring the attention computation in alternative forms) and requiring them to agree up to numerical tolerances.

In this work, we follow the *input transformation* strand: we derive simple analytic bounds on attention outputs and treat them as metamorphic relations that must hold under a family of controlled input transformations. When GPU-induced faults corrupt intermediate attention tensors, the resulting outputs are more likely to violate these relations, providing an application-level signal of potential SDC without relying on hardware error flags.

3 SDC Manifestations and Fault Models

Based on the literature and our own experimental setup, we consider two broad classes of SDC manifestations that are relevant for attention workloads.

3.1 Permanent Structural Defects

Permanent defects in hardware components (e.g., ALUs, interconnects, register files) can cause stable but incorrect behaviors in specific operations or data paths [1, 4]. At the application level, such defects may appear as:

- **Stable incorrect results**, where certain computations always produce biased or distorted outputs.
- **Numerical anomalies**, such as persistent NaNs, infinities, or denormal numbers in specific kernels.
- **Control-flow irregularities**, where loops or conditionals behave unexpectedly for particular data patterns.

Instruction-level fault simulators that model structural defects can approximate these scenarios by modifying or injecting instructions in the GPU execution stream. In attention workloads, such defects might systematically affect matrix multiplications or softmax operations, leading to recurring anomalies in specific layers or heads.

3.2 Transient Bit-Flip Errors

A more widely studied class of SDCs arises from transient bit flips in registers or memory due to radiation or other environmental factors [4, 6].

GPU-oriented tools such as register-level injectors and operator-level injectors provide practical means of emulating these faults in deep learning frameworks. In our experiments, we adopt a simplified version of this model: we inject single-bit flips into tensors associated with attention layers and observe whether the resulting outputs violate our metamorphic bounds.

4 Metamorphic Bounds for Attention

We then sketch the attention bounds that we use as metamorphic relations; full derivations and proofs are given in Appendix A.

4.1 Dot-Product Bounds for Self-Attention

We first consider a simplified single-head self-attention layer without learned projections, i.e., we assume $Q = K = V = I$ so that $q_j = k_j = v_j = x_j$. Extensions to general linear projections are discussed in Section 4.2.

Let $x_1, \ldots, x_n \in \mathbb{R}^d$ be the input token representations and

$$a_{ij} = \frac{x_i^\top x_j}{\sqrt{d}}, \qquad w_{ij} = \frac{\exp(a_{ij})}{\sum_{\ell=1}^{n} \exp(a_{i\ell})}, \qquad \mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij} x_j. \tag{2}$$

Let $j^* = \arg\max_j a_{ij}$ be the index with the largest scaled dot product with x_i, and define the margin

$$\gamma_i = a_{ij^*} - \max_{j \neq j^*} a_{ij} \;\geq\; 0. \tag{3}$$

We measure how far the attention-weighted average deviates from the best-matching token via

$$\varepsilon_i = x_i^\top x_{j^*} - x_i^\top \mathrm{Attn}(x_i) \;\geq\; 0. \tag{4}$$

Extremum and mean bounds. Because $\mathrm{Attn}(x_i)$ is a convex combination of $\{x_j\}$, the dot product with x_i must lie between the minimum and maximum:

$$\min_j x_i^\top x_j \;\leq\; x_i^\top \mathrm{Attn}(x_i) \;\leq\; \max_j x_i^\top x_j. \tag{5}$$

Using Chebyshev's sum inequality, we further obtain a mean lower bound

$$x_i^\top \mathrm{Attn}(x_i) \;\geq\; \frac{1}{n} \sum_{j=1}^{n} x_i^\top x_j. \tag{6}$$

Both quantities can be evaluated from the pairwise similarities $\{x_i^\top x_j\}$ at cost $\mathcal{O}(n)$ once these similarities have been computed.

Margin-based bounds. A more informative family of bounds follows from analyzing ε_i in terms of the softmax margin γ_i. We show in Appendix A.3 that

$$\sqrt{d}\, \frac{\gamma_i}{1 + e^{\gamma_i}} \;\leq\; \varepsilon_i \;\leq\; \min\left\{ x_i^\top x_{j^*} - \frac{1}{n} \sum_{j=1}^{n} x_i^\top x_j,\; \sqrt{d}\, \tau(\gamma_i, n) \right\}, \tag{7}$$

where $\tau(\gamma_i, n)$ is a tight upper-envelope function derived by optimizing a softmax-entropy term and involves the Lambert-W function (see Equation (28) in Appendix A.3). In particular, a simple loose upper bound is $\varepsilon_i \le \sqrt{d}\,\log n$, while $\tau(\gamma_i, n)$ yields a sharper, margin-aware upper bound.

Complexity. In a standard implementation, computing attention for a single head requires

$$\mathcal{O}(n^2 d)$$

operations to form all dot products and apply the softmax. Given the cached similarities a_{ij} and weights w_{ij} from this computation, evaluating the bounds in (6) and (7) requires:

- $\mathcal{O}(n)$ operations per token to compute $\min_j x_i^\top x_j$, $\max_j x_i^\top x_j$, and the mean $\frac{1}{n}\sum_j x_i^\top x_j$;
- $\mathcal{O}(1)$ operations per token to compute the margin γ_i, the deviation ε_i, and $\tau(\gamma_i, n)$.

Overall, the additional cost of bound checking per head is $\mathcal{O}(n^2)$ given the attention logits and weights, which is asymptotically smaller than the $\mathcal{O}(n^2 d)$ cost of the attention itself when d is large. In our experiments we treat violations of (7) as metamorphic failures potentially indicative of silent data corruption.

4.2 Bounds with Linear Projections

In practice, transformer attention uses learned linear projections $q_j = W_Q x_j$, $k_j = W_K x_j$, and $v_j = W_V x_j$. We briefly summarize how the dot-product bounds extend beyond the simplified $Q = K = V = I$ case.

We focus on the common setting where keys and values share the same projection, $W_K = W_V$, so that $k_j = v_j$. For a fixed query position i, we define

$$a_{ij} = \frac{q_i^\top k_j}{\sqrt{d}}, \qquad w_{ij} = \frac{\exp(a_{ij})}{\sum_{\ell=1}^{n}\exp(a_{i\ell})}, \qquad \mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij} k_j. \tag{8}$$

Let $j^* = \arg\max_j a_{ij}$ and $\gamma_i = a_{ij^*} - \max_{j \ne j^*} a_{ij}$ as before, and define the deviation

$$\varepsilon_i = q_i^\top k_{j^*} - q_i^\top \mathrm{Attn}(x_i). \tag{9}$$

Replacing $x_i^\top x_j$ by $q_i^\top k_j$ and re-running the same analysis, we obtain exactly the same bound shape:

$$\sqrt{d}\,\frac{\gamma_i}{1 + e^{\gamma_i}} \;\le\; \varepsilon_i \;\le\; \min\left\{ q_i^\top k_{j^*} - \frac{1}{n}\sum_{j=1}^{n} q_i^\top k_j,\ \sqrt{d}\,\tau(\gamma_i, n) \right\}, \tag{10}$$

where $\tau(\gamma_i, n)$ is the same Lambert-W upper-envelope function as in the simplified case.

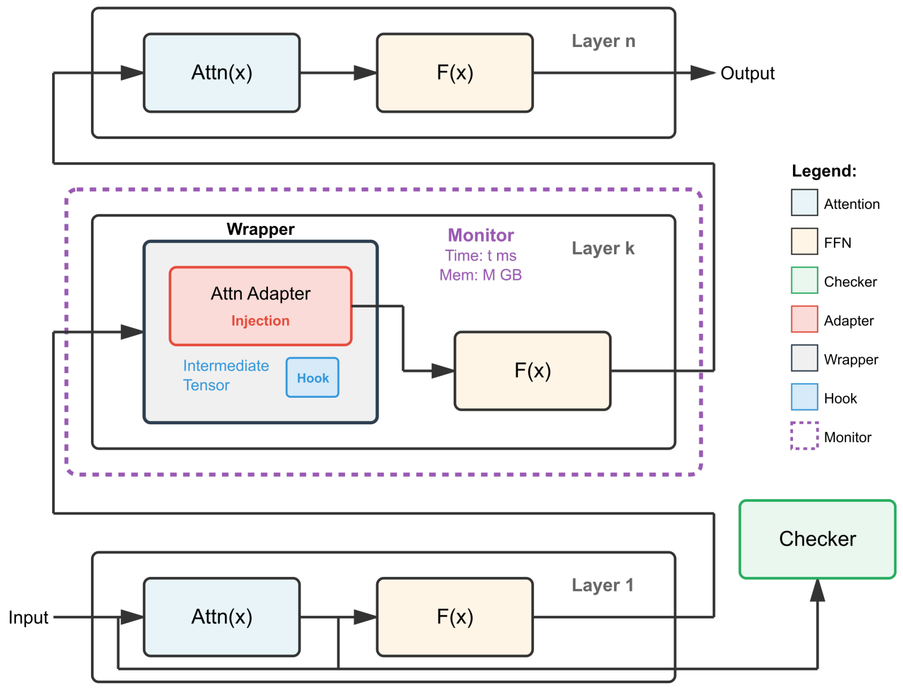

Figure 1: Overview of our attention instrumentation pipeline. A selected transformer layer k is wrapped with an attention adapter that exposes intermediate tensors and performs bit-flip injection, while the other layers remain unchanged. A monitor records runtime and memory overhead, and a metamorphic checker consumes the model outputs and analytic bounds to flag violations.

Complexity. Compared to the $Q = K = V = I$ setting, we must additionally apply the linear maps W_Q and W_K to each token, which costs $\mathcal{O}(d^2)$ per token in a naive implementation. For a single head, the overall per-layer cost becomes

$$\mathcal{O}(nd^2) \; + \; \mathcal{O}(n^2 d) \; = \; \mathcal{O}\big(n(n + d^2)\big),$$

while evaluating the bounds in (10) still adds only $\mathcal{O}(n^2)$ work given the cached logits and weights. In our GPT-2 experiments we use this projected form with $W_K = W_V$ and treat violations of (10) as metamorphic failures.

5 Experimental Methodology

5.1 Overall Pipeline

Figure 1 shows the overall pipeline of our experiments.

The main components are:

1. **Attention wrapper.** We replace or wrap the original attention module with an adapter that exposes intermediate tensors (e.g., Q, K, V, logits, softmax outputs) and allows controlled fault injection.

2. **Fault injector.** For each trial, we sample a random seed, attention layer, intermediate module, tensor index, and bit position; we then flip the chosen bit.

3. **Metamorphic checker.** We recompute the relevant expressions (e.g., $Q \cdot \mathrm{Attn}(X)$) and verify whether the observed outputs remain within the theoretical bounds.

4. **Monitor.** We log detection outcomes as well as runtime and memory overheads incurred by the wrapper and checker.

5.2 Single-Layer Attention with Gaussian Inputs

As a first step, we study a single-layer self-attention module with synthetic Gaussian inputs. The inputs are sampled i.i.d. from a Gaussian distribution with varying ranges; we focus on two configurations:

- **Narrow range** $\sigma = 1.0$: inputs are tightly concentrated, leading to relatively small attention logits and outputs.

- **Expanded range** $\sigma \approx 1.67$: inputs have larger magnitude variation, producing more spread in logits and outputs.

For each configuration, we inject single-bit flips into selected intermediate tensors and record whether the resulting outputs violate our bounds.

5.3 Minimal Transformer on Character-Level Data

We next consider a single-layer transformer that augments the attention module with embedding, feedforward, and normalization sublayers. The model operates on a character-level dataset where each character is treated as a token. In our prototype, an alphabet of size 26 is mapped to a 64-dimensional embedding, creating a relatively sparse representation.

Fault injection is again applied at attention-layer tensors. During early experiments, we observed frequent NaN values caused by the mask operation in the implementation. Disabling the mask reduced NaN occurrences but did not dramatically improve detection rates, likely due to the high sparsity and limited information content in the toy dataset.

5.4 GPT-2 on WikiText

Finally, we target a GPT-2 model evaluated on WikiText. We instrument the attention layers of GPT-2 by:

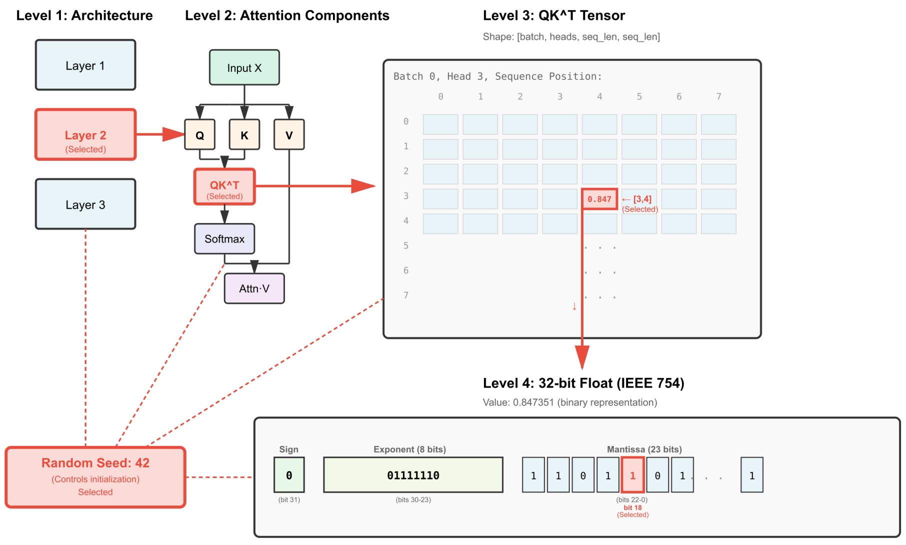

Figure 2: Example injection configuration in GPT-2 attention. Given a random seed, we pick a layer and attention submodule (here the $QK^\top$ scores), choose a tensor entry by its batch, head, and sequence indices, and flip a sampled bit in its 32-bit floating-point representation (here mantissa bit 18). The metamorphic checker then tests whether the corrupted run violates the attention bounds.

- Replacing the original attention implementation with an *adapter* that performs the same computations but exposes intermediate Q, K, V, attention logits $QK^\top/\sqrt{d}$, and softmax weights.
- Adding an *outer wrapper* that returns copies of relevant tensors and applies bit-flip injections according to a configurable schedule.
- Including a *monitor* that tracks the runtime and memory overhead of the adapter and wrapper.

We conduct campaigns where, for each trial, we randomly choose:

- A random seed (for data order and dropout).
- A specific attention layer in GPT-2.
- An intermediate module within that layer.
- A tensor index within the chosen module.
- A bit position in the floating-point representation.

For each injected fault, we evaluate the scalar quantity

$$\varepsilon_i = q_i^\top k_{j^*} - q_i^\top \mathrm{Attn}(x_i), \tag{11}$$

and check whether it violates the bounds from Section 4.2. Here j^* and γ_i are defined as in Section 4.1, and in the GPT-2 experiments we enforce $K = V$ in the injected layer so that the projected bounds in (10) apply.

Under the constraint $K = V$, the term $q_i^\top \mathrm{Attn}(x_i)$ admits two numerically equivalent computation paths, which we exploit as two metamorphic oracles:

- **Output path.** With $k_j = v_j$, the attention output at position i is

$$\mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij} k_j, \qquad w_{ij} = \frac{\exp(a_{ij})}{\sum_\ell \exp(a_{i\ell})}, \quad a_{ij} = \frac{q_i^\top k_j}{\sqrt{d}}.$$

  Taking the dot product with q_i gives

$$q_i^\top \mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij}\, q_i^\top k_j. \tag{12}$$

  In matrix form, if $Q \in \mathbb{R}^{n\times d}$ and $Y = \mathrm{Attn}(X) \in \mathbb{R}^{n\times d}$, then the diagonal of $QY^\top$ contains $\{q_i^\top \mathrm{Attn}(x_i)\}_{i=1}^n$.

- **Logit path.** Using $a_{ij} = q_i^\top k_j/\sqrt{d}$ and the same weights w_{ij}, we can rewrite (12) as

$$q_i^\top \mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij}\, q_i^\top k_j = \sqrt{d} \sum_{j=1}^{n} w_{ij}\, a_{ij}. \tag{13}$$

  Let $A = QK^\top/\sqrt{d} \in \mathbb{R}^{n\times n}$ and $W = \mathrm{Softmax}_{\mathrm{row}}(A) \in \mathbb{R}^{n\times n}$. Then (13) can be implemented using only A and W as

$$q_i^\top \mathrm{Attn}(x_i) = \sqrt{d} \sum_{j=1}^{n} W_{ij} A_{ij}, \tag{14}$$

  i.e., as a row-wise sum of the Hadamard product $A \odot W$. No direct access to $\mathrm{Attn}(x_i)$ is required in this path.

In fault-free runs, the two paths (12) and (13) are mathematically identical and both must satisfy the bounds in (10). Under bit-flip injection, different choices of intermediate module affect different subsets of tensors (Q, K, V, A, W, or the output), so one path may violate the bounds while the other does not. In our experiments, we therefore apply the same upper and lower bounds to both computations of $q_i^\top \mathrm{Attn}(x_i)$ and flag a trial as detected if either path exhibits a violation; this union of detections achieves the best coverage in our GPT-2–WikiText campaigns.

6 Results

6.1 Single-Layer Attention: Effect of Input Range

We first isolate a single multi-head attention block with a small synthetic configuration ($B = 2$, $H = 4$, $L = 8$, $D_h = 16$) and feed it Gaussian inputs

$x \sim \mathcal{N}(0, \sigma^2)$. For each trial we randomly choose an injection location $\ell \in \{\texttt{q}, \texttt{scores}, \texttt{weights}, \texttt{out}\}$, a bit position $b \in \{0, \ldots, 31\}$, and flip that bit in a single tensor element; the metamorphic checker is then applied to the corrupted forward pass. Each (ℓ, σ) configuration is exercised 512 times, for a total of 4096 single-bit injections.

To study how the dynamic range of the inputs affects detectability, we vary the standard deviation of the Gaussian. When $\sigma = 1.0$, the overall detection rate is only 9.4%; increasing to $\sigma \approx 1.67$ (most activations fall in a narrow band around $[-5, 5]$) roughly doubles the coverage to 18.5%. Averaged across locations, the output tensor `out` is the easiest to monitor: its detection rate increases from 14.7% to 39.5% as σ grows, whereas the internal tensors `q`, `scores`, and `weights` remain below 20% even in the wider-range setting (11.1% $\to$ 16.6%, 5.7% $\to$ 10.7%, and 6.1% $\to$ 7.2%, respectively). This behavior is consistent with the intuition that our attention bounds become more informative when activations and logits are larger: with small-magnitude inputs there is substantial "slack" between the true outputs and the theoretical upper/lower bounds, so many injected faults remain hidden inside that margin.

Detection is also extremely non-uniform across bit positions (Fig. 3). For all locations and both values of σ, flips in low-order mantissa bits (roughly positions 0–20) are almost never detected, whereas coverage rises sharply once we enter the exponent and sign region. For example, at $\sigma = 1.0$ the checker never fires for bits 0–20 in `q`, but detects between 38% and 56% of flips at bits 24 and 30–31; for `out`, detection reaches 50–62% around bits 23–25 and 30. At larger σ the same pattern persists, but the high-order bits become even more prominent: flipping bit 24 of `out` is detected in essentially all trials, and bit 30 of `weights` or `out` is detected in 94–100% of injections. In other words, the metamorphic bounds are largely insensitive to fine-grained mantissa perturbations but react strongly to exponent or sign flips that change a value by orders of magnitude.

These observations match the floating-point semantics and the structure of our metamorphic relation. A mantissa flip typically perturbs a value by a few units in the last place (ULPs); with small inputs, this is absorbed by the softmax and kept within the analytic bounds. In contrast, exponent or sign flips rescale or invert individual entries of $QK^\top$ or the attention weights, producing large, localized deviations that frequently violate at least one of the bounds. The dependence on injection location is also intuitive: corrupting `out` directly alters the quantity that the checker compares, whereas faults in `q`, `scores`, or `weights` may be partially canceled or attenuated on both sides of the metamorphic equation.

Overall, the single-layer experiments indicate that detection coverage is simultaneously sensitive to (i) the input distribution and (ii) the specific bit that is flipped. Even under favourable conditions (Gaussian noise, small model, no masking), the absolute coverage remains modest (on the order of 10–20%), suggesting that richer metamorphic relations or additional invariants will be necessary to robustly detect GPU SDCs in realistic multi-layer attention stacks.

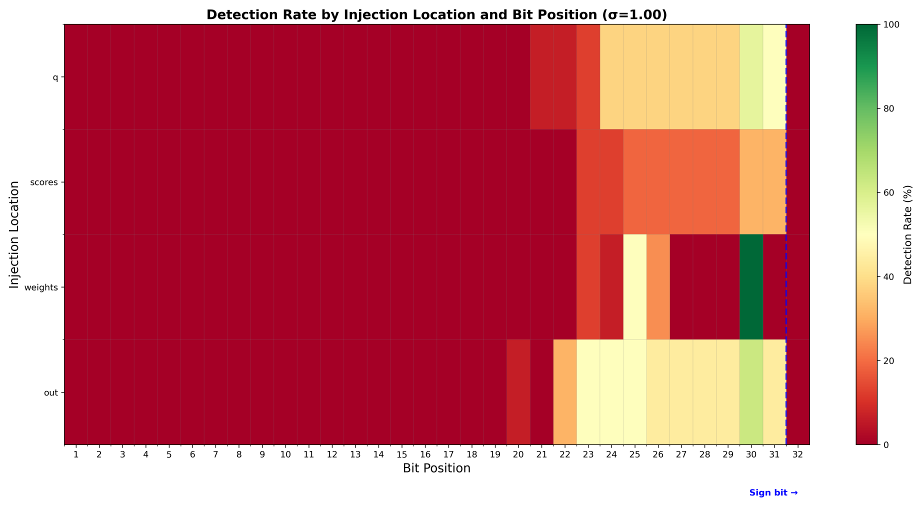

Figure 3: Detection rate by bit position and injection location for the synthetic single-layer attention experiment with $\sigma = 1.0$. Rows are locations (`q`, `scores`, `weights`, `out`), columns are mantissa and exponent/sign bits, and color encodes detection rate (%). The dashed vertical line marks the sign bit, and the sharp transition near high-order exponent/sign bits shows where the metamorphic checker becomes highly sensitive.

6.2 Minimal Transformer: Impact of Masking and Sparsity

We next move from the synthetic single-layer attention block to a minimal autoregressive transformer trained as a tiny language model on a GPT-generated character-level corpus. The model uses a 38-symbol vocabulary (letters, digits and punctuation), a 64-dimensional embedding, four heads with head dimension 16, and a block size of 32. After about 400 training steps the training loss stabilizes around 2.4, giving a network that produces reasonable but still noisy next-character predictions. For fault injection we mirror the previous setup: for each trial we select a location $\ell \in \{\mathtt{q}, \mathtt{scores}, \mathtt{weights}, \mathtt{out}\}$, flip one bit of a fixed tensor element at index $(b{=}0, h{=}0, i{=}2, j{=}7)$, and evaluate the metamorphic checker on the corrupted forward pass over 16 mini-batches, yielding in total $4 \times 16 \times 32 = 2048$ bit flips.

Effect of masking. In the first implementation the attention layer included both a causal mask and a padding mask. These were realized in the standard way by adding large negative constants to masked entries of the score matrix $QK^\top$ before softmax. Under fault injection this turned out to be brittle: flipping bits in the masked score entries frequently changed the sentinel values (e.g. from -10^4 to a finite number or even to NaN), which in turn caused the softmax to overflow or to propagate NaNs through the attention weights and outputs. Such failures are trivially detectable—simple runtime checks for NaNs would flag them—but they are not informative about the *subtle* silent data corruptions that our metamorphic bounds are designed to catch. To focus the evaluation on these harder cases, we therefore disabled the padding mask and ran the experiments on fixed-length sequences without masked positions; the results below correspond to this mask-free configuration.

Overall detection behavior. Even without masking, the bounds remain far from exhaustive. Across all locations and bits the overall detection rate is 26.2% (537/2048). Detection is highly location-dependent: injections into `weights`, `q`, and `out` are detected in roughly a third of the trials (33.2%, 32.8%, and 31.3%, respectively), whereas perturbing the score matrix `scores` is detected only 7.6% of the time. This mirrors the analytic structure of the metamorphic relation, which ties the input/output activations and attention weights together more tightly than the intermediate, scaled logits $QK^\top$.

Bit-wise statistics further highlight this asymmetry. For `q`, `weights`, and `out`, most bit positions achieve a fairly uniform detection rate around 31%, with a pronounced spike for bit 30 of `weights` (the highest-order exponent/sign bit in our indexing), which is detected in 94% of injections. In contrast, `scores` exhibits zero detection for all mantissa and low-order exponent bits (positions 0–22) and non-zero coverage only for high-order bits 23–30 (25–31%), where flipping the bit induces large, order-of-magnitude changes in particular logits. Thus, as in the single-layer experiment, the checker is much more responsive to coarse exponent/sign faults than to fine-grained mantissa perturbations, but in this more realistic transformer the sensitivity concentrates on locations that directly enter the bounds (`q`, `weights`, `out`) rather than on the scores.

Batch-to-batch variability. Another striking aspect is the large variation across mini-batches. For `q`, `weights`, and `out`, the per-batch detection rate ranges from almost 0% to 100%, with standard deviations of 44–48 percentage points; `scores` is more stable but also much lower, with 0–25% detection and an 11% standard deviation. Inspection of the stored bounds shows that batches where detection "spikes" to 100% tend to have very small upper bounds but comparatively large ϵ, so even moderate changes to outputs quickly violate the inequality; in batches with broader, more forgiving bounds, most bit flips stay within the allowed region. This suggests that the efficacy of the metamorphic relation is not only a function of the fault itself, but also of the instantaneous confidence and sharpness of the attention distribution in that batch.

Role of sparsity and representation. Finally, the moderate overall coverage can be related to the sparsity and redundancy of the character-level representation. A 64-dimensional embedding for a small alphabet yields many directions in which the model can move without significantly altering the predicted distribution; moreover, we inject a single bit flip into one head and one

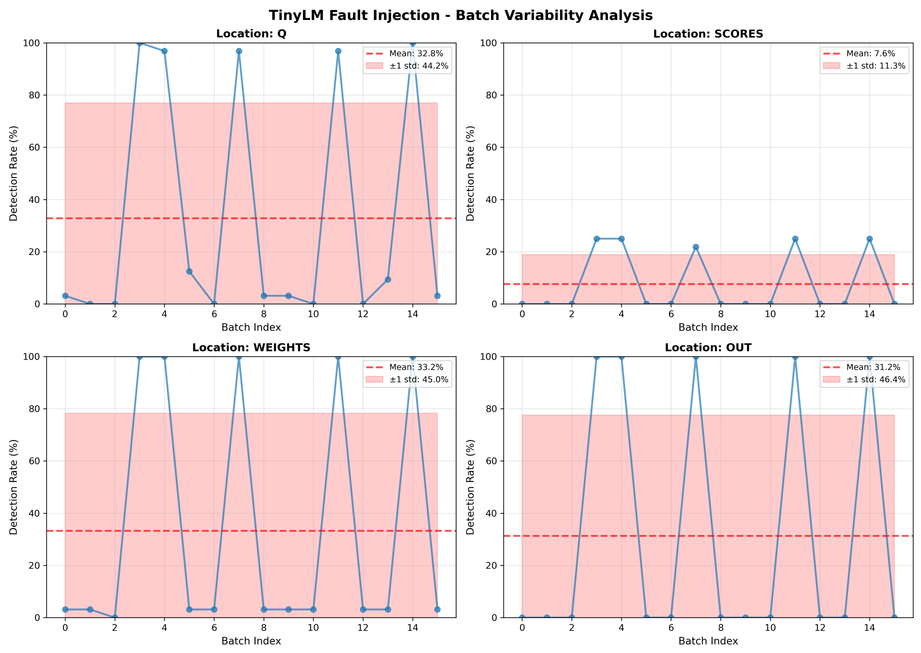

Figure 4: Batch-wise detection rates under TinyLM fault injection. Each panel plots detection rate (%) across 16 batches for one injection location (`q`, `scores`, `weights`, `out`); the dashed line is the mean and the shaded band is ± 1 standard deviation. Large fluctuations across batches highlight how masking and aggregation can amplify or hide a single flipped bit even under the same checker.

position of the sequence, while the loss and the bounds aggregate information across all heads and tokens. Many injected faults therefore either land on "uninformative" subspaces of the embedding, or their effect is diluted by uncorrupted heads and positions, so that the metamorphic inequality remains satisfied.

Overall, the TinyLM experiments illustrate that realistic architectural features—masking, sparsity of embeddings, and multi-head aggregation—interact with fault patterns in non-trivial ways. Bounds derived for idealised dense attention remain applicable but provide only partial coverage, indicating that practical SDC detection for transformers will likely require combining such metamorphic checks with additional invariants or lightweight runtime instrumentation.

6.3 GPT-2: Bit Position and Metamorphic Forms

To move beyond the synthetic and TinyLM settings, we instrument a single attention layer of GPT-2 ("small", 12 layers, 12 heads) fine-tuned on WikiText. For each run in the parameter sweep we select a configuration consisting of

- one of four random seeds $\{0, 42, 123, 3407\}$,

- one of four attention layers $\{0, 3, 6, 9\}$,
- one of four injection locations $\{\mathbf{k}, \mathbf{q}, \texttt{scores}, \texttt{weights}\}$,
- one of four fixed tensor indices $(0, 0, 10, 20)$, $(0, 3, 53, 43)$, $(1, 6, 32, 1)$, or $(1, 9, 31, 62)$,
- and one bit position $b \in \{0, \dots, 31\}$ in the IEEE-754 single-precision representation.

This yields $4\times4\times4\times4\times32 = 8192$ single-bit fault configurations per experiment. For each configuration we run a clean forward pass to establish reference bounds, inject a bit flip into the chosen tensor entry, recompute the attention outputs, and record whether the metamorphic inequality is violated and how much the loss changes.

Dominance of bit position. Across all four GPT-2 experiments the bit position is by far the most important predictor of whether a fault is detected. In the baseline setting where we do not enforce $K = V$ and use the generalized $QK^\top$–softmax form ("$k \neq v,\ s@w$"), no violations are observed for mantissa and low-order exponent bits $b \leq 22$, and detection only begins to rise for bits $b \geq 23$. Even there the rates are modest (up to $\approx 19\%$) except for the second-highest exponent/sign bit $b = 30$, which is detected in 51.6% of configurations and also accounts for the largest average loss change ($|\Delta\mathcal{L}| \approx 8.6\times10^{-3}$). When we enforce $K = V$ (see below), the same qualitative pattern holds but with substantially higher rates in the high-order region: bits 23–29 reach 10–30% detection, and bit 30 alone is detected in $\approx$ 73–75% of configurations depending on the metamorphic form.

In contrast, all other sweep dimensions are comparatively flat. For the $k \neq v,\ s@w$ experiment the average detection rate is 4.5% overall, with only mild variation across locations (2.9–5.9%), layers (2.3–7.8%), random seeds (all 4.5%), and tensor indices (2.3–6.3%). The three $K = V$ experiments behave similarly: changing the layer, seed, or index shifts the detection rate by at most a few percentage points, whereas conditioning on the bit position produces order-of-magnitude differences. This supports our earlier observations from the single-layer and TinyLM experiments that detection is primarily governed by whether the flipped bit belongs to the mantissa (fine perturbations) or to the exponent/sign region (order-of-magnitude changes). Figure 5 visualizes these trends as heatmaps over bit position while sweeping tensor indices, layers, injection locations, and random seeds.

NaNs at high-order bits. A noteworthy side effect of flipping bit 30 (and, less frequently, bit 31) is the appearance of NaNs in the intermediate tensors. In the $k \neq v,\ s@w$ experiment we observe NaNs in 80 out of 8192 runs; 20 of these cases both produce NaNs and violate the metamorphic bound, almost exclusively involving bit 30 at locations $\mathbf{k}$ or $\mathbf{q}$ in deeper layers. After enforcing $K = V$, the number of NaN runs roughly halves to 40 per experiment, with

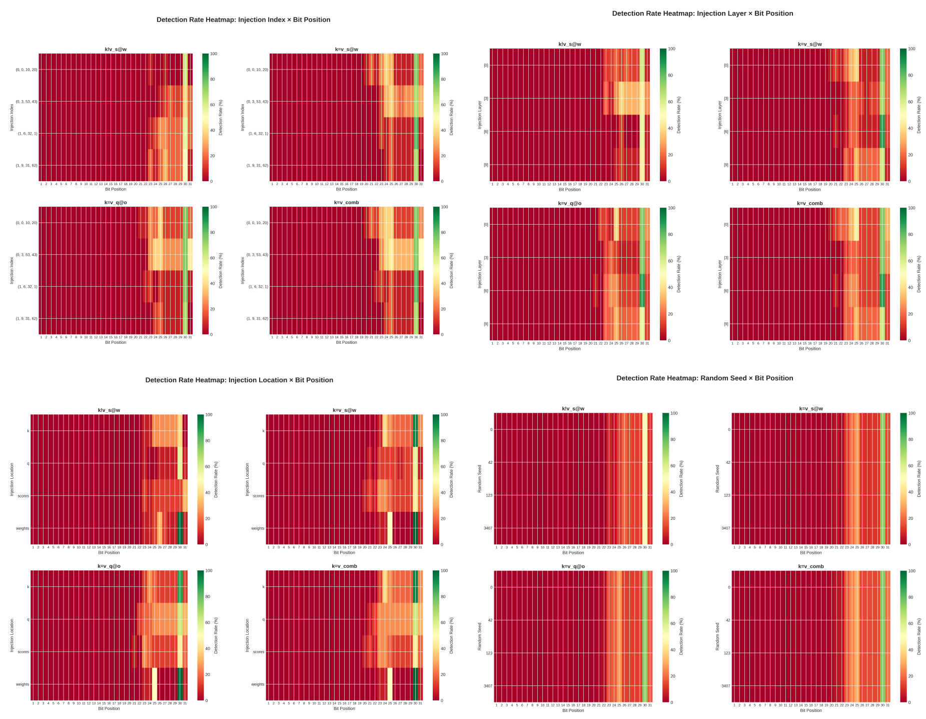

Figure 5: Detection rate heatmaps for GPT-2 metamorphic checks. Each panel shows detection rate (%) over flipped bit position b and, respectively, injection index, injection layer, injection location, and random seed (top left to bottom right). Within each panel, the four subplots correspond to the four GPT-2 experiments ($k \neq v$, $s@w$, $k = v$, $s@w$, $k = v$, $q@o$, $k = v$, `comb`). The vertical structure within panels highlights that bit position dominates the other sweep dimensions in determining detection.

12 NaN + violation cases each. Qualitatively this is expected: toggling a high-order exponent bit can push individual logits or attention weights to extremely large magnitudes, which in turn can overflow during softmax or subsequent matrix multiplications and propagate NaNs. Since our goal is to study subtle SDCs rather than trivial NaN failures, we exclude NaN runs from the aggregate detection statistics reported below; doing so does not affect the monotonic trend with bit position.

6.3.1 Forcing $K = V$ vs. Generalized Bounds

Under the generalized metamorphic relation we treat K and V as independent and formulate the bound in terms of the score and weight tensors (`scores` $\cdot$ `weights`, abbreviated as "$s@w$"). This configuration yields 372 violating configurations out of 8192 (4.5% coverage) once NaN runs are removed. However, in many practical attention implementations K and V are derived from the same underlying projection, and setting $K = V$ during inference does not change the overall scale of the attention output due to the softmax normalization. We therefore also evaluate the theoretical bounds derived under this simplifying assumption.

When we force $K = V$ and continue to use the $s@w$-based check ("$k = v,\ s@w$"), the number of detected violations increases to 504 configurations (6.2% coverage). The distribution over bit positions again shows that almost all of this gain comes from the high-order region: bit 30 is now detected in 73.4% of cases compared to 51.6% without the constraint, and bits 23–27 contribute more violations as well. Crucially, both the generalized and the $K = V$ bounds hold exactly for all fault-free runs, confirming that *either* formulation is sound, but the $K = V$ specialization is empirically more sensitive to perturbations that distort the combined score–weight structure.

6.3.2 Combining Two Equivalent Forms

Under $K = V$ we can express the scalar quantity under test in two algebraically equivalent ways:

1. using logits and attention weights, $f_{\texttt{s@w}} = (\texttt{scores} \cdot \texttt{weights})_{\text{diag}}$,
2. or using the attention output explicitly, $f_{\texttt{q@o}} = (Q \cdot \text{Attn}(X))_{\text{diag}}$, abbreviated as "$q@o$".

Each form yields its own instance of the metamorphic inequality. We therefore run two separate sweeps:

- $k = v,\ s@w$: 504 violating configurations (6.2% coverage);
- $k = v,\ q@o$: 524 violating configurations (6.4% coverage).

The bit-wise detection curves of these two experiments are extremely similar: for bits $b \leq 22$ both forms detect essentially nothing; bits 23–29 reach 15–30% detection, and bit 30 again dominates at 73.4% for both forms. Yet the sets of violating configurations are not identical. When we take the union of detections from both checks (experiment "$k = v,\ \texttt{comb}$"), the number of violating configurations rises to 596 (7.3% coverage), corresponding to a relative improvement of about 17% over the better single form.

A complementary view is to condition on the floating-point region where faults occur. If we average detection only over the exponent and sign bits (bits 23–31), the mean detection rates rise to 16.1% for $k \neq v,\ s@w$, 21.0% for $k = v,\ s@w$, 22.2% for $k = v,\ q@o$, and 24.7% for the combined checker $k = v,\ \texttt{comb}$, corresponding to 568 detections out of 2304 injected faults in the latter case. In other words, most of the extra coverage we obtain from the $K = V$ specialization and from combining the two forms comes from faults that flip high-order exponent or sign bits, while mantissa flips remain largely invisible.

This suggests that, although $f_{\texttt{s@w}}$ and $f_{\texttt{q@o}}$ are mathematically equivalent in exact arithmetic, they differ in their numerical sensitivity to specific perturbations. Some faults affect the logits and weights in ways that are more visible through the $s@w$ form; others primarily distort the aggregated attention output and are more easily detected by the $q@o$ form. Leveraging multiple equivalent views of the same underlying computation therefore offers a principled way to increase SDC coverage without changing the base model.

6.4 Runtime and Overhead

A central question for practical deployment is the additional cost of our instrumentation and metamorphic checks. For each GPT-2 experiment we record (1) the wall-clock time and peak memory of the reference run, (2) the corresponding metrics under fault injection, and (3) the incremental time spent in the checker routines. Table 1 summarises the results. Figure 6 complements this summary with a per-bit view of injection, detection, and memory overheads for all four GPT-2 experiments.

Across all four GPT-2 sweeps, the fault-injection wrapper itself introduces a small and largely configuration-independent overhead: the measured runtime changes by roughly 7% (slightly negative due to noise and caching effects), and peak memory increases by about 5%. These costs stem from (i) allocating mutable copies of the selected tensors so we can flip bits without modifying the original forward pass, and (ii) saving a small number of intermediate outputs required by the bounds.

The metamorphic checks add an additional per-batch computation whose magnitude depends on the chosen form:

- For the generalized and $k = v$, $s@w$ experiments, the checker primarily computes $(p \cdot \texttt{scores})_{\text{diag}}$ and a few reductions, which has the same asymptotic cost as a single extra head-wise matrix multiplication but over a relatively small block size. In practice, the detection routines account for $\approx 13\%$ of the total runtime per configuration, with an average absolute detection time of 2.5 ms.

- For the $k = v$, $q@o$ experiment, we additionally evaluate $(\text{Attn}(X) \cdot Q)_{\text{diag}}$, which introduces another batched matrix multiplication of similar shape. Here the checker accounts for $\approx 19.8\%$ of runtime, with an average detection time of 4.1 ms.

- In the combined $k = v$, `comb` setting we compute both forms in a single pass and take their union. As expected from the theory, the measured detection time rises only slightly over $q@o$ (to 4.2 ms, or 20.1% of runtime), since the two forms share most intermediate tensors and only require a few extra reductions.

These measurements line up with the asymptotic analysis from Section 5.4. If the base attention block has cost $\mathcal{O}(BHL^2 d_h)$, then the $s@w$ checker adds

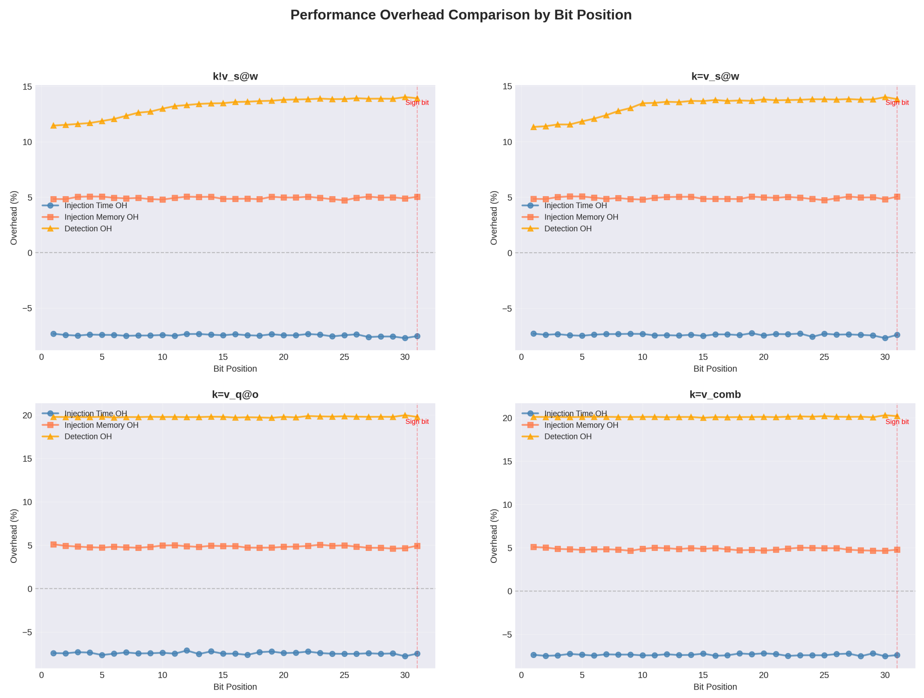

Figure 6: Performance overhead of GPT-2 metamorphic checking as a function of flipped bit position b. Each panel corresponds to one of the four GPT-2 experiments ($k \neq v,\ s@w$, $k = v,\ s@w$, $k = v,\ q@o$, $k = v,$ `comb`), and plots the relative overhead (%) in injection time, additional memory for injected tensors, and detection time versus a clean baseline run. The vertical dashed line marks the sign bit. Overheads stay nearly flat across bit positions, indicating that the moderate costs reported in Table 1 are largely independent of which bit is flipped and are instead driven by the choice of metamorphic form.

roughly one extra matrix–vector product per head, while $q@o$ and `comb` add up to two. Given that we instrument only one layer and that modern GPUs are highly optimized for batched GEMMs, the resulting overheads of $\approx$ 13–20% appear acceptable for offline SDC testing and even for selective online monitoring (e.g., checking only a subset of layers or timesteps). Further engineering—such as fusing the checks into the main attention kernel or using lower precision for the bound computation—could reduce this cost, but we leave such optimizations to future work.

7 Discussion and Future Directions

Our experiments demonstrate that attention-based metamorphic relations can detect a non-trivial subset of bit-flip-induced SDCs in GPU workloads, espe-

Table 1: Runtime and memory overhead of metamorphic checking in GPT-2 parameter sweeps. “FI time” compares the fault-injected forward pass (without bounds) to a clean baseline; “Detect time” is the absolute and relative cost of the metamorphic checks themselves. All numbers are averaged over 8192 configurations per experiment, NaN runs included.

Experiment	FI time Δ	Mem. Δ	Detect time (s)	Detect time (%)
$k \neq v,\ s@w$	-7.5%	$+4.9\%$	0.0025	13.2%
$k = v,\ s@w$	-7.4%	$+4.9\%$	0.0025	13.2%
$k = v,\ q@o$	-7.4%	$+4.8\%$	0.0041	19.8%
$k = v,$ `comb`	-7.4%	$+4.8\%$	0.0042	20.1%

cially when targeting high-order bits and using multiple equivalent forms of the computation. At the same time, the overall coverage remains limited, and many practical challenges remain.

7.1 Extending the Bounds

From a theoretical standpoint, several extensions are worth pursuing:

- **Masked attention and QKV modelling.** Real transformer implementations use masking (e.g., causal masks, padding masks) and often employ more complex QKV projection patterns. Extending the bounds to properly account for masks and richer QKV structures is a natural next step.
- **Gradient-level bounds.** During training, errors in backpropagation gradients may be as harmful as errors in forward activations. Developing bounds for gradients through attention layers could enable metamorphic checks in training regimes.
- **Floating-point–aware bounds.** Different floating-point formats exhibit different vulnerability patterns. Bounds that explicitly account for exponent and mantissa structure might yield more targeted detection criteria.

7.2 Integrating with Fault Simulators and Architectures

On the experimental side, our current setup uses a relatively simple injection model. Future work should explore:

- **Integration with realistic fault simulators.** Porting the approach to GPU fault simulators that model both structural defects and transient faults would allow more realistic campaigns and bigger models.
- **Modern attention variants.** Contemporary architectures employ grouped-query attention (GQA), multi-head layouts with heterogeneous dimensions, and mixed-precision arithmetic. Adapting the bounds and metamorphic checks to these variants is likely to uncover new failure modes.

- **Approximation-based approaches.** If tight analytic bounds are difficult to obtain, approximate bounds with empirically calibrated correction factors could still be useful as practical detectors.

7.3 Relation to Existing SDC Work

Our work can be seen as sitting between hardware-level SDC studies [1, 3, 6, 7] and application-level characterizations for LLM training [8]. Instead of trying to estimate global SDC rates or assess vulnerability for arbitrary workloads, we focus on a single but central primitive: attention. This narrow focus allows us to exploit structural properties that are invisible at the hardware level and too detailed for high-level training metrics.

8 Conclusion

We have described an exploratory study of GPU silent data corruption detection for attention-based models using metamorphic relations derived from attention bounds. Motivated by field studies of SDC at scale [1, 2, 3, 4, 6, 7, 8], we proposed to treat attention layers as structured computation blocks amenable to lightweight consistency checks.

Our preliminary experiments show that, even with simple bounds and single-bit flip injections, we can detect a meaningful subset of SDCs, particularly those involving high-order bits. Forcing $K = V$ in the tested layer and combining multiple equivalent forms of the attention output further improves coverage. While the current prototype is far from a complete solution, it provides evidence that attention-specific metamorphic relations are a promising ingredient in a broader SDC detection toolkit for deep learning workloads. Together, these results suggest that simple, structure-aware metamorphic bounds can provide non-trivial SDC coverage for attention layers at modest runtime and memory cost.

Future work will focus on deriving tighter and more general bounds, supporting masked attention and gradients, integrating with realistic GPU fault simulators, and extending the approach to modern attention architectures.

A Proofs of Attention Bounds

This appendix sketches the main derivations of the bounds stated in Section 4.1. We follow the notation introduced there.

A.1 Extremum and Norm Bounds

Recall that for a single head we write

$$\mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij} x_j$$

with $w_{ij} \in (0,1)$ and $\sum_j w_{ij} = 1$. For any fixed $c \in \mathbb{R}^d$,

$$c^\top \mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij}\, c^\top x_j \tag{15}$$

is a convex combination of the scalars $\{c^\top x_j\}$. It therefore lies in the interval

$$\min_j c^\top x_j \;\le\; c^\top \mathrm{Attn}(x_i) \;\le\; \max_j c^\top x_j,$$

which yields the extremum bound used in the main text. Setting $c = x_i$ and applying Cauchy–Schwarz to each $x_i^\top x_j$ immediately gives the corresponding norm bound

$$-\|x_i\| \max_j \|x_j\| \;\le\; x_i^\top \mathrm{Attn}(x_i) \;\le\; \|x_i\| \max_j \|x_j\|.$$

A.2 Chebyshev Mean Lower Bound

We rewrite

$$x_i^\top \mathrm{Attn}(x_i) = \sum_{j=1}^{n} w_{ij}\, x_i^\top x_j = \sqrt{d}\, \frac{\sum_{j=1}^{n} a_{ij} b_{ij}}{\sum_{j=1}^{n} b_{ij}}, \quad a_{ij} = \frac{x_i^\top x_j}{\sqrt{d}}, \quad b_{ij} = \frac{\exp(a_{ij})}{\sum_{\ell=1}^{n} \exp(a_{i\ell})}. \tag{16}$$

Consider the two sequences $\{a_{ij}\}_{j=1}^{n}$ and $\{b_{ij}\}_{j=1}^{n}$ ordered so that both are non-decreasing in j. Since the softmax weights b_{ij} are an increasing function of a_{ij}, the two sequences are positively associated, and Chebyshev's sum inequality gives

$$\frac{1}{n}\sum_{j=1}^{n} a_{ij} b_{ij} \;\ge\; \Big(\frac{1}{n}\sum_{j=1}^{n} a_{ij}\Big)\Big(\frac{1}{n}\sum_{j=1}^{n} b_{ij}\Big). \tag{17}$$

Using $\sum_j b_{ij} = 1$ and multiplying by $\sqrt{d}$ yields

$$x_i^\top \mathrm{Attn}(x_i) \;\ge\; \frac{1}{n}\sum_{j=1}^{n} x_i^\top x_j, \tag{18}$$

which is the mean lower bound (6) in the main text.

A.3 Margin Bounds and Lambert-W Function

Let

$$j^* = \arg\max_j a_{ij}, \qquad \gamma_i = a_{ij^*} - \max_{j\neq j^*} a_{ij} \geq 0.$$

We define the deviation

$$\varepsilon_i = x_i^\top x_{j^*} - x_i^\top \mathrm{Attn}(x_i) = \sqrt{d} \sum_{j\neq j^*} w_{ij} \left(a_{ij^*} - a_{ij}\right) \;\geq\; 0. \tag{19}$$

Lower bound. For all $j \neq j^*$ we have $a_{ij^*} - a_{ij} \geq \gamma_i$, hence

$$\varepsilon_i \;\geq\; \sqrt{d}\,\gamma_i \sum_{j\neq j^*} w_{ij} = \sqrt{d}\,\gamma_i(1 - w_{ij^*}). \tag{20}$$

Using

$$w_{ij^*} = \frac{e^{a_{ij^*}}}{\sum_j e^{a_{ij}}} = \frac{1}{1 + \sum_{j\neq j^*} e^{-(a_{ij^*}-a_{ij})}} \;\leq\; \frac{1}{1+e^{-\gamma_i}}, \tag{21}$$

we obtain $1 - w_{ij^*} \geq 1/(1+e^{\gamma_i})$ and hence the relaxed lower bound

$$\varepsilon_i \;\geq\; \sqrt{d}\,\frac{\gamma_i}{1+e^{\gamma_i}}. \tag{22}$$

Upper bound. For the upper bound, introduce

$$\mu_{ij} = e^{-(a_{ij^*}-a_{ij})} \quad (j \neq j^*), \qquad S_i = \sum_{j\neq j^*} \mu_{ij}.$$

A direct manipulation shows that

$$\varepsilon_i = \sqrt{d}\left(a_{ij^*} - \sum_j w_{ij} a_{ij}\right) = \sqrt{d}\,\frac{\sum_{j\neq j^*} \mu_{ij}(-\log \mu_{ij})}{1+S_i}. \tag{23}$$

Interpreting $f(x) = -x\log x$ as a concave function and applying Jensen's inequality to $\{\mu_{ij}\}_{j\neq j^*}$ with fixed sum S_i yields

$$\sum_{j\neq j^*} \mu_{ij}(-\log \mu_{ij}) \;\leq\; (n-1)\, f\!\left(\frac{S_i}{n-1}\right) = S_i\left[\log(n-1) - \log S_i\right]. \tag{24}$$

Thus

$$\varepsilon_i \leq \sqrt{d}\,\Phi(S_i), \qquad \Phi(S) = \frac{S}{1+S}\,\log\frac{n-1}{S}. \tag{25}$$

Using $a_{ij^*} - a_{ij} \geq \gamma_i$ again gives

$$S_i = \sum_{j\neq j^*} e^{-(a_{ij^*}-a_{ij})} \;\leq\; (n-1)e^{-\gamma_i},$$

so

$$\varepsilon_i \le \sqrt{d} \sup_{0 \le S \le (n-1)e^{-\gamma_i}} \Phi(S). \tag{26}$$

Optimizing $\Phi(S)$ over $S > 0$ leads to the equation

$$\frac{d\Phi}{dS} = 0 \quad \Longleftrightarrow \quad Se^S = \frac{n-1}{e}, \tag{27}$$

whose solution is $S^* = W\big((n-1)/e\big) = \mathcal{W}_n$. One can check that $\Phi(S)$ increases on $[0, \mathcal{W}_n]$ and decreases on $(\mathcal{W}_n, +\infty)$, so the constrained maximum occurs at

$$\tau(\gamma_i, n) = \sup_{0 \le S \le (n-1)e^{-\gamma_i}} \Phi(S) = \begin{cases} \Phi((n-1)e^{-\gamma_i}), & (n-1)e^{-\gamma_i} \le \mathcal{W}_n, \\ \Phi(\mathcal{W}_n), & (n-1)e^{-\gamma_i} > \mathcal{W}_n, \end{cases} \tag{28}$$

which is the expanded form of $\tau(\gamma_i, n)$ used in the main text. Combining the lower and upper bounds yields the final inequality (7) in Section 4.1.

A.4 Extending to Linear Projections

In practice, attention uses linear projections $q_j = W_Q x_j$, $k_j = W_K x_j$, and $v_j = W_V x_j$. We sketch how the bounds from Section 4.1 extend beyond the $Q = K = V = I$ setting.

Identical transform $W_Q = W_K = W_V = W$. Let $z_j = W x_j$ and

$$a_{ij} = \frac{z_i^\top z_j}{\sqrt{d}}, \qquad \text{Attn}(z_i) = \sum_j w_{ij} z_j.$$

With

$$j^* = \arg\max_j a_{ij}, \quad \gamma_i = a_{ij^*} - \max_{j \ne j^*} a_{ij}, \quad \varepsilon_i = z_i^\top z_{j^*} - z_i^\top \text{Attn}(z_i),$$

the analysis of Section 4.1 carries over verbatim after replacing x_j by z_j, yielding the same bound shape:

$$\sqrt{d}\, \frac{\gamma_i}{1 + e^{\gamma_i}} \ \le\ \varepsilon_i \ \le\ \min\left\{ z_i^\top z_{j^*} - \frac{1}{n} \sum_j z_i^\top z_j,\ \sqrt{d}\, \tau(\gamma_i, n) \right\}.$$

Shared KV projection $W_K = W_V$. In our GPT-2 experiments we use $W_K = W_V$ and a separate W_Q. Then

$$q_j = W_Q x_j, \quad k_j = v_j = W_K x_j,$$

and

$$a_{ij} = \frac{q_i^\top k_j}{\sqrt{d}}, \qquad \text{Attn}(x_i) = \sum_j w_{ij} k_j.$$

With

$$j^* = \arg\max_j a_{ij}, \quad \gamma_i = a_{ij^*} - \max_{j \neq j^*} a_{ij}, \quad \varepsilon_i = q_i^\top k_{j^*} - q_i^\top \mathrm{Attn}(x_i),$$

we obtain exactly the projected bound used in the main text:

$$\sqrt{d}\,\frac{\gamma_i}{1+e^{\gamma_i}} \;\le\; \varepsilon_i \;\le\; \min\left\{ q_i^\top k_{j^*} - \frac{1}{n}\sum_{j=1}^{n} q_i^\top k_j,\; \sqrt{d}\,\tau(\gamma_i, n) \right\}. \tag{29}$$

Linear alignment $K = SV$. If keys and values are related by a fixed linear map S (so $k_j = Sv_j$), define $\tilde{q}_i = S^\top q_i$. Then

$$q_i^\top k_j = \tilde{q}_i^\top v_j,$$

and the logits and weights can be viewed as coming from $(\tilde{q}_i, v_j)$. Defining

$$\varepsilon_i = \tilde{q}_i^\top v_{j^*} - \tilde{q}_i^\top \mathrm{Attn}(x_i),$$

the same derivation yields the bound

$$\sqrt{d}\,\frac{\gamma_i}{1+e^{\gamma_i}} \;\le\; \varepsilon_i \;\le\; \min\left\{ \tilde{q}_i^\top v_{j^*} - \frac{1}{n}\sum_{j} \tilde{q}_i^\top v_j,\; \sqrt{d}\,\tau(\gamma_i, n) \right\}.$$

Lipschitz-type alignment. More generally, suppose that for a given query q_i there exists $L \ge 0$ such that

$$\left| q_i^\top v_{j^*} - q_i^\top v_j \right| \;\le\; L \left| q_i^\top k_{j^*} - q_i^\top k_j \right| \quad \text{for all } j. \tag{30}$$

Then

$$\varepsilon_i = \sum_j w_{ij} \bigl(q_i^\top v_{j^*} - q_i^\top v_j \bigr) \;\le\; L \sum_j w_{ij} \bigl(q_i^\top k_{j^*} - q_i^\top k_j \bigr),$$

and reusing the previous analysis gives

$$\varepsilon_i \;\le\; L \cdot \min\left\{ q_i^\top k_{j^*} - \frac{1}{n}\sum_{j} q_i^\top k_j,\; \sqrt{d}\,\tau(\gamma_i, n) \right\}. \tag{31}$$

Thus, under a Lipschitz-type alignment between K and V, the projected bounds retain their functional form up to a multiplicative factor L in the upper bound.

References

[1] Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, and Sriram Sankar. Silent Data Corruptions at Scale. *arXiv:2102.11245*, 2021.

[2] Rich Bonderson. Silent Data Corruption in Systems at Scale. In *ITC Silicon Lifecycle Management Workshop*, 2021.

[3] 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 SOSP*, 2023.

[4] Jon Calhoun, Marc Snir, Luke N. Olson, and William D. Gropp. Towards a More Complete Understanding of SDC Propagation. In *Proceedings of HPDC*, 2017.

[5] Harish Dattatraya Dixit, Laura Boyle, Gautham Vunnam, Sneha Pendharkar, Matt Beadon, and Sriram Sankar. Detecting Silent Data Corruptions in the Wild. *arXiv:2203.08989*, 2022.

[6] Siva Kumar Sastry Hari, Paolo Rech, Timothy Tsai, Mark Stephenson, Arslan Zulfiqar, Michael Sullivan, Philip Shirvani, Paul Racunas, Joel Emer, and Stephen W. Keckler. Estimating Silent Data Corruption Rates Using a Two-Level Model. *arXiv:2005.01445*, 2020.

[7] Manoj Vishwanathan, Ronak Shah, Kyung Ki Kim, and Minsu Choi. Silent Data Corruption (SDC) Vulnerability of GPU on Various GPGPU Workloads. Proceedings of the International SoC Design Conference (ISOCC), 2015.

[8] Jeffrey Ma, Hengzhi Pei, Leonard Lausen, and George Karypis. Understanding Silent Data Corruption in LLM Training. *arXiv:2502.12340*, 2025.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

[10] Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, 2016.

[11] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, 51(1):4:1–4:27, 2018.