Silent Data Corruptions in AI Systems: Evaluation and Mitigation

Xun Jiao, Fred Lin, Harish D. Dixit, Joel Coburn, Daniel Moore, Sriram Sankar
Meta

Abstract

The increasing complexity, heterogeneity, and scale of AI hardware systems make them increasingly susceptible to hardware faults, e.g., silent data corruption (SDC). Tackling this challenge requires answering the question: How to evaluate and mitigate the impact of SDCs on AI systems? For evaluation, we propose a novel quantitative metric, Parameter Vulnerability Factor (**PVF**) [15], inspired by architectural vulnerability factor (AVF) in computer architecture, aiming to standardize the evaluation of SDC impact on AI models. **PVF** focuses on SDC occurring in model parameters – we define a model parameter's **PVF** as the probability that a corruption in that particular model parameter would result in an incorrect output. Through extensive fault injection (FI), we obtain **PVF** for a set of open-source models including DLRM, CNN, and BERT, as well as three Meta's production ranking and recommendation model, based on which we present unique insights. For mitigation, we propose **Dr. DNA** [21], a novel approach to detect and mitigate SDCs by formulating and extracting a set of unique SDC signatures from the Distribution of Neuron Activations (DNA). Through an extensive evaluation across 10 different models, results show that **Dr. DNA** achieves 100% SDC detection rate for most cases, 95% detection rate on average and >90% detection rate across all cases, representing 20% - 70% improvement over baselines. **Dr. DNA** also mitigates the impact of SDCs by recovering model performance with <1% memory overhead and <2.5% latency overhead.

1 Introduction

The reliability of AI systems directly translates to the dependability, safety, functionality, and efficiency of services and operations running on top of them. For instance, in recommendation model inference, a reliable model is essential for accurate personalized recommendations, crucial for achieving positive business outcomes. Unfortunately, as AI hardware systems become increasingly complex, heterogeneous and scaled up, and as transistor technology plunges into the deep-nanometer regime, the reliability of AI hardware systems faces a mounting challenge and a rising susceptibility to hardware faults that can be caused by manufacturing defects, aging components, or environmental factors [2, 3, 9, 28].

In particular, hardware faults that are not reported by standard fault reporting mechanisms but leading to erroneous application behavior have become increasingly prominent and harder to detect in production systems, due to their elusive nature and subtle manifestations. We refer to these as **silent data corruption (SDC)**, e.g., bit flips, which has been observed in CPU systems by Meta [6], and confirmed by Google and Alibaba [13, 29]. In AI hardware, Nvidia reported that "Hopper architecture GPUs may intermittently experience SDC resulting in incorrect results" [1], and Google reported hard-to-debug SDCs in their TPU systems [10]. Recently, in a 54-day period of Llama 3 405B pre-training job at Meta, SDC occurred 6 times on GPU clusters, which could lead to the pause of the entire 16K-GPU cluster due to checkpointing and recovery [7].

The 2nd Workshop on Hot Topics in System Infrastructure (HotInfra'24), co-located with SOSP'24, November 3, 2024, Austin, TX, USA

In this paper, we consider SDC occurring in model parameters (weights, bias, embeddings, etc) because they make most of the total storage needed for a model, which is aligned with previous studies [17, 19, 26, 27]. This can create what is referred to as *parameter corruption*, where AI model parameters are corrupted and their original values are altered, potentially leading to model output corruptions. Combating this challenge requires answers to:

- Q1: How to evaluate the impact of SDC? Specifically, how likely is an SDC to result in an incorrect model output, and how do different parts (such as modules and layers) of the models exhibit varying vulnerability levels to SDCs? Existing works used metrics such as accuracy drop [22, 27] or SDC rate [2, 19], focusing on model-level vulnerability, but there is a lack of an unified parameter-level vulnerability metric that can answer this question: *How likely is a parameter corruption to result in an incorrect model output?* The answer to this question is critical in AI hardware design, especially when mapping AI model parameters or software variables to hardware blocks which may have varying fault protection capabilities.
- Q2: How to mitigate the impact of SDC? Specifically, how to detect insidious SDCs causing only minor changes in model parameters, and how to detect SDCs in early stages before it propagates to output with small overhead? Existing techniques include model re-parameterization [11, 20], detecting outliers in parameters [4, 8], or detecting model loss [24], or selective protections of the most vulnerable parameters [16]. However, these SDC detection/mitigation techniques face challenges including low accuracy, long latency, and notable overhead.

To address these questions, we propose **PVF** (for Q1) and **Dr. DNA** (for Q2), which summarizes our earlier papers [15, 21].

2 PVF: Evaluating Impact of SDC

2.1 Overview

PVF is inspired by architectural vulnerability factor (AVF) in computer architecture community. AVF quantifies the vulnerability of a processor's microarchitecture to soft errors [25]. An architectural structure's AVF is the probability that a fault in that particular structure will result in a program output error. Similarly, we define a model parameter's PVF as the probability that a corruption in that particular model parameter will result in an incorrect model output, e.g., a wrong click prediction or a wrong image classification. Similar to AVF as a statistical concept, PVF needs to be derived through a large number of FI experiments that are statistically meaningful. PVF is a versatile metric that can be tailored to different AI models/tasks and is also adaptable to different hardware fault models. While we focus on applying PVF to inference in this paper, the same metric and process can be extended to training stage to evaluate the effects of parameter corruptions on the model's convergence capability. A key distinction between training and inference is that during training, the parameters are dynamically updated in different iterations, making PVF a function of SDC's temporal location as well, in addition to its spatial location.

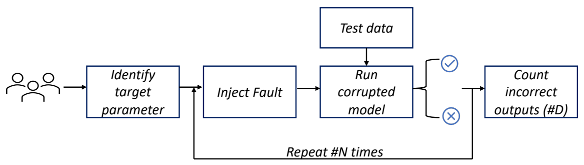

Figure 1: Compute PVF via Fault injection (FI) [15]

2.2 Compute PVF via Fault Injection (FI)

Fig. 1 shows an overall flow to compute PVF through a FI process: (1) Identify Target Parameters in the given AI model that we want to compute PVF; examples include an embedding table, a layer, or a specific weight tensor; (2) Inject Faults: For the target parameter, we inject faults based on the given fault model, e.g., random bit flip which is the most typical way to model SDCs [17, 19, 26, 27]; (3) Evaluate the Corrupted Model: Run the AI model inference with the injected faults on the test data, and compare the model's output with ground-truth output. Step 2-3 is considered as one FI experiment; (4) Repeat Step 2-3 for N times (e.g., 1 million) wherein each FI we use different random faults and inputs, and record the number of incorrect output (D); (5) Compute PVF as the D/N. Note, because the model can have wrong predictions without any hardware errors, we only focus on cases where correct predictions become incorrect. Note depends on specific scenario, there may be different evaluation metrics or definitions for "incorrectness", e.g., at service-level, normalized entropy (NE) score is typically used.

2.3 Case Study on AI Models

We apply PVF to various AI models including CNN, BERT, and DLRM in [15]. Due to space limit, we only present one example based on DLRM in this paper. DLRM (deep learning recommendation model) was developed by Meta for personalized content recommendation, and has constituted 79% of the overall AI inference cycles at the Meta data center [14]. DLRM is mainly composed of embedding tables and dense layers that has top MLP layers (top-MLP) and bottom MLP layers (bot-MLP).

Fig. 2 illustrates the PVF of three DLRM parameter components, embedding table, bot-MLP, and top-MLP, under 1, 2, 4, etc to 128 random bit flips during each inference. We observe different vulnerability levels across different parts of DLRM. For example, under a single bit flip, the embedding table has relatively low PVF; this is attributed to embedding tables being highly sparse. However, top-MLP can have 0.4% PVF under even a single bit flip. This is significant – for every 1000 inferences, four inferences will be incorrect. This highlights the importance of protecting specific vulnerable parameters for a given model based on the PVF measurement.

With 128 bit flips, PVF increases to 40% and 10% for top-MLP and bot-MLP components respectively, while observing multiple NaN values. Top-MLP component has higher PVF than bot-MLP. This is attributed to the top-MLP being closer to the final model,

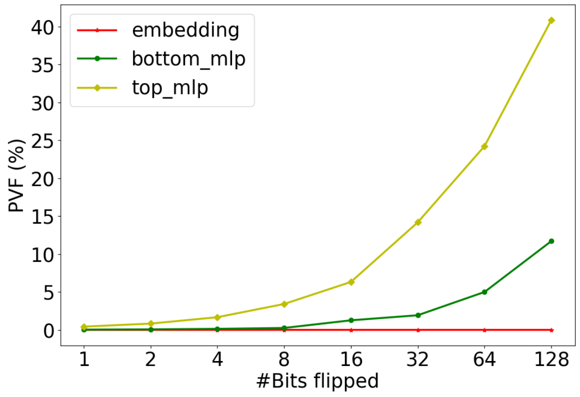

Figure 2: PVF of DLRM Parameters under multi-bit flip [15]

and hence has less of a chance to be mitigated by inherent error masking probability of neural layers. For more results on DLRM, please refer to [15].

2.4 Case Study on Production Models

We apply PVF to three production rank and recommendation models at Meta, and observe similar phenomenon as the open-source DLRM, where the dense layers are more vulnerable than embedding. Further, we observe that, rather than the sign bit, the MSBs in the exponential field (referred to as *E-MSB*) of floating point weights are especially vulnerable. We have observed (1) bit flip in *E-MSB* may lead to NaN, (2) *E-MSB* of certain dense weights would have a close to 100% PVF, meaning *E-MSB* flip in those weights would almost certainly lead to an incorrect output, and (3) the average PVFs of all dense weights *E-MSBs* across three model are >20%.

3 Dr. DNA: Mitigating Impact of SDC

3.1 Overview

To mitigate SDC impact, we propose **Dr. DNA** [21] which leverages the unique SDC signatures in DNNs. The key idea behind **Dr. DNA** is the hypothesis that an SDC will imprint a unique signature in the Distribution of Neuron Activations (DNA) for a DNN model. Fig. 3 shows an overview of the proposed **Dr. DNA** which features two stages: offline profiling and SDC signature formulation, and online detection and mitigation. Profiling is done completely offline to obtain formulated SDC signatures based on DNA statistics, while detection and mitigation is performed online to identify and mitigate SDCs during inference. Due to space limit, please refer to [21] for more details.

3.2 Offline SDC Signature Formulation

3.2.1 Profiling Neuron Activation. The first stage of **Dr. DNA** is to profile neuron activation information at the end of the model training. Specifically, a cohort of random indices of neurons are selected for collecting neuron activations of each layer as "profiling sites". During model inference, the value of those profiling sites are recorded for each sample. After one full pass of the profiling set, a histogram can be elaborated for each profiling site, showing the distribution of neuron activation values. Suppose we select

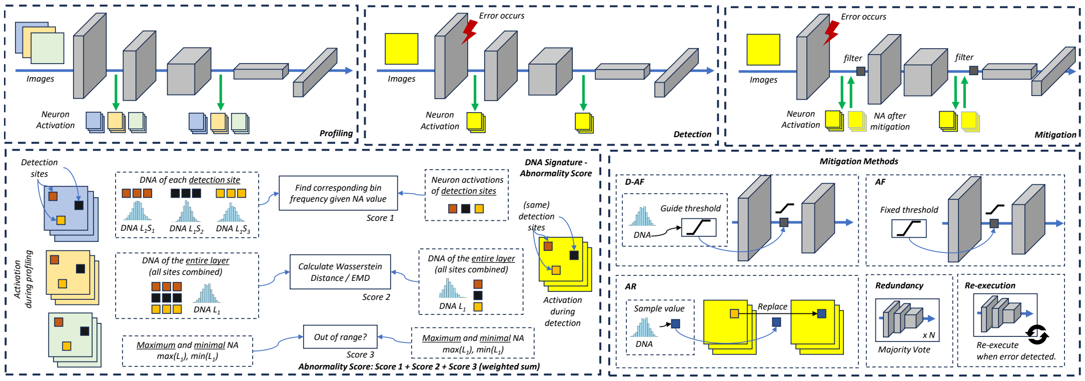
Images
Neuron Activation
Profiling
Error occurs
Detection
filter
NA after mitigation
Mitigation
Detection sites
Activation during profiling
DNA of each detection site
DNA L_1S_1
DNA L_1S_2
DNA L_1S_3
Find corresponding bin frequency given NA value
Score 1
Neuron activations of detection sites
DNA Signature - Abnormality Score
(same) detection sites
Activation during detection
DNA of the entire layer (all sites combined)
DNA L_1
Calculate Wasserstein Distance / EMD
Score 2
Maximum and minimal NA max(L_1), min(L_1)
Out of range?
Score 3
Abnormality Score: Score 1 + Score 2 + Score 3 (weighted sum)
Mitigation Methods
D-AF
Guide threshold
DNA
AF
Fixed threshold
AR
Sample value
Replace
Redundancy
x N
Majority Vote
Re-execution
Re-execute when error detected.

Figure 3: Overview of Dr. DNA [21]

k profiling sites for each layer and there are n total layers, **Dr. DNA** will develop $k \times n$ profiling sites and histograms. In addition to developing the histograms, we also record two additional sets of statistics during profiling to extend the concept of DNA for SDC detection: the distribution of the profiled neuron activations within one layer depicted by another histogram of each layer, and the extreme neurons of each layer determined by the highest and lowest activation value.

3.2.2 SDC Signature and Abnormality Score. The profiling sites from the profiling stage are used as "detection sites" during detection. In the forward pass of a sample, **Dr. DNA** will leverage the activation values from those detection sites to calculate detection metrics. Based on the three metrics, we can define an "abnormality score" to quantitatively describe how the SDC signature, i.e., DNA of a given inference are abnormal from the previously profiled SDC-free DNA (details in [21]).

3.3 Online Detection and Mitigation

3.3.1 Early Detection of SDC. During detection, if the cumulative abnormality score of a sample is considerably higher than the profiled scores, i.e., the margin is higher than a threshold, **Dr. DNA** will terminate the inference earlier and indicate the detection of errors. The threshold can be empirically configured, or through experimental evaluations under the error injection.

3.3.2 Mitigation Methods. **Activation Filtering (AF)**: A commonly used approach to mitigate the SDC is to clip outlier activation values, which has been investigated to improve the fault tolerance [5, 12]. By limiting the activation range, abnormally large values can be filtered out. The implementation of AF is straightforward and has almost no overhead other than modifying the activation function. **DNA-Guided Activation Filtering (D-AF)**: In **Dr. DNA**, activation clipping [4] can be guided using the profiled DNA statistics, where the clipping threshold of an activation for filtering in **Dr. DNA** are automatically determined from DNA. **Activation Re-distribution (AR)**: We also propose a finer-grained mitigation method based on **Dr. DNA**, referred to as activation re-distribution (AR). When SDC is detected, the abnormal activation value would be replaced by a randomly sampled activation based on the histograms.

3.4 Experimental Results

Table 1: Comparison between Dr. DNA and baselines [21].

Detection	Rate	FP	Margin
Activation (Cosine Sim.)	0.2	0.39	2.3%
Activation (EMD)	0.16	0.38	1.6%
Min/Max	0.73	0.07	7.7%
Dr. DNA	✓	0.03	10.6%
Mitigation	Efficacy	Memory	Latency
Redundancy	Excellent	up to 300%	Trivial
Re-execution	Unusable	Trivial	High
AF	Poor	Trivial	Trivial
Dr. DNA (D-AF)	Good	<0.5%	<3%
Dr. DNA (AR)	Good	<0.5%	<18%

Table 1 presents the results across 3 vision tasks, 5 different datasets, and 10 different models, under 4 different error models. We compare **Dr. DNA** with baseline detection methods: Activation (Cosine Sim) [18, 23], Min/Max [4, 12], and Activation (EMD), a simplified version of **Dr. DNA** without using individual detection sites metric. Results show that **Dr. DNA** outperform all the baseline methods with higher detection rate and lower false positives. We compare **Dr. DNA** with baseline mitigation methods: Redundancy based on triple modular redundancy (TMR), Re-execution [24], and Activation Filtering [4, 8, 30]. Results show that D-AF can effectively mitigate the impact of SDCs by recovering model performance with small overhead, outperforming baseline methods.

4 Conclusion

This paper summarizes our recent works [15, 21] on tackling SDC challenge in AI systems through novel evaluation metrics and mitigation methods. **PVF** is designed to quantify the vulnerability of AI models to parameter corruptions. Through fault injection, PVF can be calculated for any target parameter component of a given AI model. **Dr. DNA** formulates and leverages the unique SDC signatures derived from the Distribution of Neuron Activations (DNA) to effectively detects and mitigates SDCs early during DNN inference. We hope that our studies will inspire further research in academia and industry to address the critical issue of SDCs in AI systems.

References

[1] 2023. tesla-release-notes: https://docs.nvidia.com/datacenter/tesla/tesla-release-notes-535-129-03/index.html.

[2] Udit Kumar Agarwal, Abraham Chan, and Karthik Pattabiraman. 2023. Resilience Assessment of Large Language Models under Transient Hardware Faults. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 659–670.

[3] Chun-Kai Chang, Sangkug Lym, Nicholas Kelly, Michael B Sullivan, and Mattan Erez. 2018. Evaluating and accelerating high-fidelity error injection for hpc. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 577–589.

[4] Zitao Chen, Guanpeng Li, and Karthik Pattabiraman. 2021. A low-cost fault corrector for deep neural networks through range restriction. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 1–13.

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085* (2018).

[6] Harish Dattatraya Dixit et al. 2022. Detecting silent data corruptions in the wild. *arXiv preprint arXiv:2203.08989* (2022).

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[8] Florian Geissler, Syed Qutub, Sayanta Roychowdhury, Ali Asgari, Yang Peng, Akash Dhamasia, Ralf Graefe, Karthik Pattabiraman, and Michael Paulitsch. 2021. Towards a safety case for hardware fault tolerance in convolutional neural networks using activation range supervision. *arXiv preprint arXiv:2108.07019* (2021).

[9] Peter Hazucha et al. 2003. Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-/spl mu/m to 90-nm generation. In *IEDM*.

[10] Yi He, Mike Hutton, Steven Chan, Robert De Gruijl, Rama Govindaraju, Nishant Patil, and Yanjing Li. 2023. Understanding and Mitigating Hardware Failures in Deep Learning Training Systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–16.

[11] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. 2020. Defending and harnessing the bit-flip based adversarial weight attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14095–14103.

[12] Le-Ha Hoang et al. 2020. Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation. In *DATE*.

[13] Peter H Hochschild, Paul Turner, Jeffrey C Mogul, Rama Govindaraju, Parthasarathy Ranganathan, David E Culler, and Amin Vahdat. 2021. Cores that don't count. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. 9–16.

[14] Samuel Hsia et al. 2023. MP-Rec: Hardware-Software Co-design to Enable Multi-path Recommendation. In *ASPLOS*.

[15] Xun Jiao, Fred Lin, Harish D Dixit, Joel Coburn, Abhinav Pandey, Han Wang, Jianyu Huang, Venkat Ramesh, Wang Xu, Daniel Moore, et al. 2024. PVF (Parameter Vulnerability Factor): A Quantitative Metric Measuring AI Vulnerability and Resilience Against Parameter Corruptions. *arXiv preprint arXiv:2405.01741* (2024).

[16] Navid Khoshavi, Arman Roohi, Connor Broyles, Saman Sargolzaei, Yu Bi, and David Z Pan. 2020. Shieldenn: Online accelerated framework for fault-tolerant deep neural network architectures. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[17] Sung Kim et al. 2018. MATIC: Learning around errors for efficient low-voltage neural network accelerators. In *DATE*.

[18] Troya Çağıl Köylü, Cezar Rodolfo Wedig Reinbrecht, Said Hamdioui, and Mottaqiallah Taouil. 2021. Deterministic and statistical strategies to protect anns against fault injection attacks. In *2021 18th International Conference on Privacy, Security and Trust (PST)*. IEEE, 1–10.

[19] Guanpeng Li et al. 2017. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In *SC*.

[20] Jingtao Li, Adnan Siraj Rakin, Yan Xiong, Liangliang Chang, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. 2020. Defending bit-flip attack through dnn weight reconstruction. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[21] Dongning Ma, Fred Lin, Alban Desmaison, Joel Coburn, Daniel Moore, Sriram Sankar, and Xun Jiao. 2024. Dr. DNA: Combating Silent Data Corruptions in Deep Learning using Distribution of Neuron Activations. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, (ASPLOS)*. 239–252.

[22] Abdulrahman Mahmoud et al. 2020. Pytorchfi: A runtime perturbation tool for dnns. In *DSN-W*.

[23] Abdulrahman Mahmoud, Siva Kumar Sastry Hari, Christopher W Fletcher, Sarita V Adve, Charbel Sakr, Naresh Shanbhag, Pavlo Molchanov, Michael B Sullivan, Timothy Tsai, and Stephen W Keckler. 2020. Hardnn: Feature map vulnerability evaluation in cnns. *arXiv preprint arXiv:2002.09786* (2020).

[24] Abdulrahman Mahmoud, Siva Kumar Sastry Hari, Christopher W Fletcher, Sarita V Adve, Charbel Sakr, Naresh R Shanbhag, Pavlo Molchanov, Michael B Sullivan, Timothy Tsai, and Stephen W Keckler. 2021. Optimizing Selective Protection for CNN Resilience.. In *ISSRE*. 127–138.

[25] Shubhendu S Mukherjee, Christopher Weaver, Joel Emer, Steven K Reinhardt, and Todd Austin. 2003. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36*. IEEE, 29–40.

[26] Brandon Reagen et al. 2016. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ISCA*. IEEE.

[27] Brandon Reagen et al. 2018. Ares: A framework for quantifying the resilience of deep neural networks. In *DAC*.

[28] Behrooz Sangchoolie, Karthik Pattabiraman, and Johan Karlsson. 2017. One bit is (not) enough: An empirical study of the impact of single and multiple bit-flip errors. In *2017 47th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 97–108.

[29] Shaobu Wang, Guangyan Zhang, Junyu Wei, Yang Wang, Jiesheng Wu, and Qingchao Luo. 2023. Understanding Silent Data Corruptions in a Large Production CPU Population. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 216–230.

[30] Jinyu Zhan, Ruoxu Sun, Wei Jiang, Yucheng Jiang, Xunzhao Yin, and Cheng Zhuo. 2021. Improving fault tolerance for reliable DNN using boundary-aware activation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 10 (2021), 3414–3425.