This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2025.3544537

Understanding Error Propagation in Deep-Learning Neural Networks Accelerators and Applications

Guanpeng Li[1], Siva Kumar Sastry Hari[2], Michael B. Sullivan[2], Timothy Tsai[3], Karthik Pattabiraman[4], Joel S. Emer[2], Stephen W. Keckler[2]
[1]University of Iowa, [2]NVIDIA, [3]Independent Scholar, [4]University of British Columbia

***Abstract*—Deep neural networks (DNNs) are making their way into safety-critical systems, but they can be vulnerable to soft errors in hardware. Traditional duplication-based resilience methods are too expensive. This work studies how DNN inference's deployment parameters impact error resilience via fault injection studies. Based on the error propagation analysis, it proposes new, efficient protection techniques specifically for DNN systems. This research laid the groundwork for further studies in DNN application resilience and safety.**

***Index Terms*—Resilience, Soft Errors, Deep Neural Networks**

I. Introduction

Deep learning neural networks (DNNs) are becoming increasingly prevalent in both cloud and edge applications. Autonomous systems such as vehicles and robots are also employing DNNs to perform complex tasks such as sensor data processing, perception, localization, path prediction, and even trajectory generation. While there has been significant research on improving DNN performance, there have been few studies on safety and reliability, which are paramount in such systems. Thus, our knowledge is limited about the impact of soft errors on DNN systems. This paper fills the critical gap in our understanding of DNNs' reliability under soft errors. Soft errors, or hardware transient faults, are typically caused by high-energy particles striking electronic devices causing them to malfunction. They can lead to application failures and safety violations. This is particularly concerning for safety-critical systems like self-driving cars that process sensor data in real-time to make driving decisions using an in-vehicle computer. Figure 1 illustrates how a soft error in a DNN, when deployed in an AV, can potentially result in a collision. The rate and severity of errors could violate safety standards such as ISO 26262 for automotive vehicles [1].

This paper presents some of the first experiments to characterize the propagation of soft errors from the hardware to the application software of DNN systems. Based on these results it devises highly effective, yet low-cost error mitigation mechanisms in both software and hardware.

Traditional high-level replication-based methods that duplicate or triplicate computations (e.g., Triple Modular Redundancy or TMR) are effective, but incur high energy and area overheads. These are both critical resources, especially in edge real-time systems. Researchers have also proposed methods to duplicate work at a finer granularity (thread- or instruction-level) that offer lower overheads, but the overheads are still high for practical real-time applications. Furthermore, these techniques consider neither the error propagation characteristics nor the architectures of DNN algorithms and accelerators.

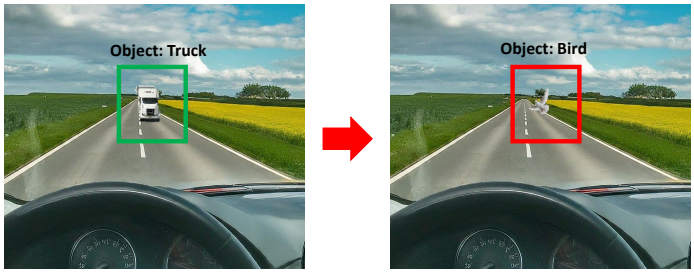


Fig. 1: Illustrative example of a Silent Data Corruption (SDC) due to a soft-error in a classification DNN used for an Autonomous Vehicle (AV).

This study first investigates the error propagation characteristics of DNNs, and leverages these results to develop cost-effective error mitigation solutions. The key findings are as follows.

- SDC rate varies across different DNNs and data types. However, the variation with the data type outweighs the variation with the DNN type. For example, a floating-point 32 bit data type with 21 bits for the exponent and 10 bits for the mantissa has a 260× higher SDC rate than the one with 5 bits for the exponent and 26 bits for the mantissa.
- The effect of a faulty bit in the higher-order exponent bits is significantly greater in terms of SDC rates.
- Faulty values that highly deviate from zero in the DNN's intermediate layers are more likely to lead to SDCs.
- Faults in the later layers of the DNN are more likely to lead to an SDC as they have a greater likelihood of propagating to the DNN's outputs and causing an SDC.

Further, we found that the FIT [1] rates of DNNs of accelerator platforms at the time of publishing of the original work [2] exceeded those suggested by safety standards, and hence, mitigation of errors was needed to meet the safety standards. Consequently, we proposed both software and hardware techniques to mitigate the effects of soft errors in DNN accelerators.

[1]One FIT or Failures in Time is defined as one failure in a billion hours of operation.

Authorized licensed use limited to: MIT. Downloaded on March 16,2025 at 19:19:59 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2025.3544537

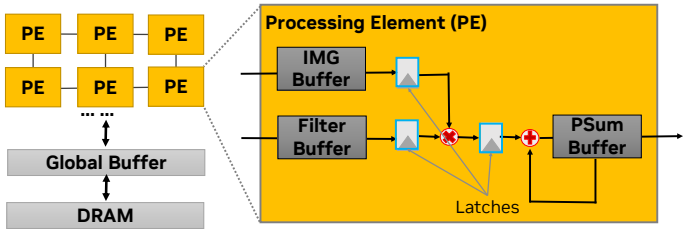


Fig. 2: Architecture of general DNN Accelerators

II. Soft Errors in DNN Accelerators

In this work, we study the error characteristics of convolutional neural networks (CNNs), a class of DNNs. A CNN consists of multiple computation layers – convolutional and fully-connected are the two main types of layers. Each CONV layer applies a kernel (or filter) on the input feature maps (ifmaps) to extract underlying visual characteristics and generate the corresponding output feature maps (ofmaps). Each computation result is saved in an activation after being processed by an activation function (e.g., ReLU), which in turn, becomes the input to the next layer.

While many specialized accelerators have been proposed for inferencing, all DNN accelerators perform multiply-and-accumulate (MAC) operations for each feature map leveraging parallel compute, and exploit temporal and spatial localities in data within and across each feature map to allow the data to be reused. DNN accelerators employ an array of processing engines (PEs), each of which computes MACs. Figure 2 shows the architecture of a general DNN accelerator. The accelerator is connected to DRAM and the data is cached on-chip in a global buffer. Each PE consists of a multiplier and an adder as execution units to perform MACs. Researchers propose accelerators with different data reuse characteristics (weight, input, and output reuse) offering different trade-offs. Eyeriss [3] considers all of the three data localities in its dataflow by implementing *Filter SRAM, Img REG* and *PSum REG* in each PE for data reuse. We adopted the original Eyeriss microarchitectural parameters at 65nm and projected them to 16nm technology node. Based on this assumption, we calculate the number of PEs and buffer sizes by keeping the total area constant.

ISO 26262 deals with the functional safety of road vehicles [1] and offers multiple levels of certification. ASIL-D (strictest) requires the System on Chip (SoC) running DNN inferencing to have a soft error FIT rate less than 10 FIT. A DNN accelerator should be a fraction of the total area of the SoC.

III. Evaluation Methodology

We use fault injection to understand the effect of soft-errors on DNN accelerators and applications. We consider faults that occur in the data paths (latches) and buffers (SRAMs and latch-arrays) of each PE. We inject single bit-flip faults in the sequential elements. We use four commonly deployed pre-trained DNNs for classification (at the time of original publication) and datasets – ConvNet with CIFAR-10 and AlexNet, CaffeNet, and NiN with ImageNet. We consider six data types that offer trade-offs between representable value ranges, precision, and implementation overhead. The considered data types are double (FP64, E11M52[2]), float (FP32, E8M23), float16 (FP16, E5M10), FxP_32_26 (32-bit fixed point datatype with 26 fractional bits and 5 integer bits), FxP_32_10, and FxP_16_10.

Some of the results are conducted using error injections using an open-source C++-based DNN simulator called Tiny-CNN. The loop-nest in the simulator was modified to capture the Eyeriss micro-architecture (by adding levels for re-use buffers). 3000 random faults were injected in each of the hardware components during the inference, one fault per execution, resulting in an error bar between 0.11% and 0.34%. If an error injection experiment corrupts the output of the network, the severity of the corruption will have a different effect on the outcome. Therefore, we define Silent Data Corruption (SDC) based on the classification's outcome. Whenever the top-ranked element predicted by the DNN is different from that predicted by its fault-free execution, we call it SDC-1. If it is not among the top five predicted elements, we call it SDC-5. If the confidence score of the top-ranked element varies by $>$10% or $>$20%, we call it SDC-10%or SDC-20%, respectively.

IV. Charactarization Results

A. Datapath Faults

Models trained with ImageNet dataset exhibit little difference in the SDC probability for the four different types of SDCs. Since these models output 1000 dimensions, whenever the top-ranked output is changed by the error, the new ranking is mostly outside of the top five elements and the confidence score changes $>$20%. Since we observed little difference between the SDC types, we focus on SDC-1moving forward.

SDC probabilities vary considerably across data types. We observed higher SDC probability for data types with larger ranges. We investigate this by studying the SDC probability based on the bit-position of the injected fault for a given data type. Figure 3 shows the results for NiN using FP16 data type. Results for other models and data types follow the same trend – the corruptions in high-order exponent bits are likely to cause SDCs, but not the mantissa and sign bits. We also observed that bit-flips that go from 0 to 1 in the high-order exponent bits are more likely to cause SDCs than those going from 1 to 0. We also observe that the data type with limited range offers more resilience. For example, FP16 NiN observed significantly lower SDC probability than FP32 NiN.

We investigate the corrupted neuron values by comparing them to the fault-free values. We randomly sampled a set of activations while running AlexNet using FP16, and compared

[2] n-bit floating-point data type with a number of exponent bits and b number of mantissa bits is shown as FPn, EaMb. n-bits fixed-point data type with a bits for the integer part and f bits for the fractional part is shown as FxP_n_f. Both the data types assume a sign bit, not shown in the respective representation.

Authorized licensed use limited to: MIT. Downloaded on March 16,2025 at 19:19:59 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted,
but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2025.3544537

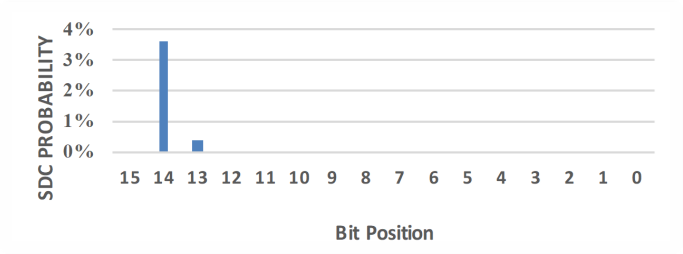
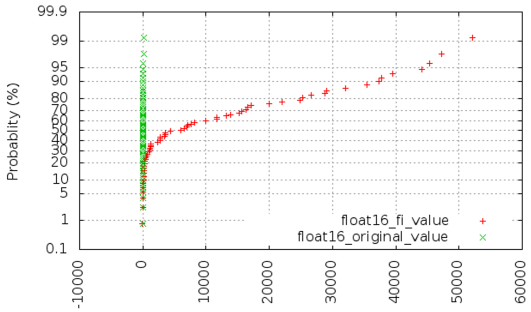
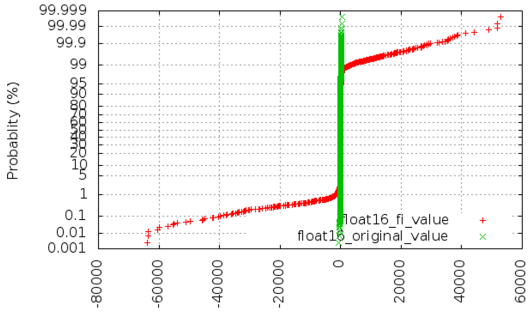


Fig. 3: SDC probabilities (y-axis) based on the corrupted bit position (x-axis) for FP16 data type in NiN are shown here.



(a) SDCs.



(b) Benign.

Fig. 4: Values before (green dots) and after (red dots) error occurrence in AlexNet using FP16. X-axis shows the activations' values.

their values before (in green) and after (in red) error injections. The results are grouped based on the inference outcome – SDC or benign (no change in the outcome) shown in Figures 4a and 4b, respectively. The results show that a large deviation in an activation's value due to an error is likely to cause an SDC. More than 80% of errors that lead to a large deviation lead to an SDC in Figure 4a. On the other hand, only 2% of errors that cause large deviations do not affect the inference outcome (stay benign). This is likely because large deviations make it harder for values to converge back to their fault-free values.

We also investigate how errors in different layers propagate through the model. For AlexNet and CaffeNet, we observed that the SDC probability increases with the layer count. We observe that the use of Local Response Normalization (LRN) normalizes the faulty values and helps to mitigate the effect of large deviations. NiN does not employ a normalization layer, and hence, it has a relatively flat SDC probability across the layers.

B. Error propagation through CNNs

This work motivated several follow-up studies that also aim to quantify the probability with which low-level errors propagate to the output of the CNNs conducting perception tasks. These perception tasks include processing the per-sensor data in AVs using DNNs. These studies computed this probability for each frame processing (we refer to it as P_{CNN}) [2], [4]–[6]. The follow-up studies use more recent models and larger datasets. These studies also injected transient errors. The errors were either injected in simulated hardware or real silicon using software tools, or via high-energy particle experiments. We summarize the findings in Table I. While the errors were not injected using same fault model or data type during inference, the compiled results show that the CNNs are naturally highly resilient, i.e., only a small fraction of the injected faults will propagate to the output.

V. Low-cost DNN Error Mitigation

A. Symptom-based Error Detectors (SED)

A symptom-based detector catches errors by leveraging application-specific anomalies and failure symptoms. Unusual (anomalous) variable values, loop iteration count, or addresses are a few examples of a symptom. For the DNNs, we propose a value range-based symptom that uses the value ranges of activations and looks for anomalous values to avoid SDCs during inferences. This detector follows our observation that an error which makes the magnitude of an activation very large is likely to cause an SDC.

First, we need to determine the usual fault-free value ranges [-X, Y] of the neurons in each of the layers of the model under study for representative inputs. We can then use these value ranges as the bounds for the detector. To account for unseen inputs, we apply a 10% margin on the per-layer value ranges [-1.1*X, 1.1*Y] for symptom detection. The ranges are obtained only once before the model is deployed. To lower the cost of error detection, the activations can be streamed to the host to perform the range checks in parallel. The checks can also be merged with the activation function.

After deploying the symptom-error detector, we measure its coverage using fault injections in each hardware component, using all three FP data types in AlexNet, CaffeNet, and NiN. We measured precision (the number of benign faults detected / the number of injected faults injected), and recall: (the number of detected SDC-causing faults / the number of total SDC-causing faults). The average precision and recall were high–90.2% and 92.5%, respectively.

In subsequent work, we enhance the technique by leveraging key insights and introducing Ranger [7], a low-cost fault correction mechanism. Ranger mitigates transient fault-induced errors without requiring re-computation. While DNNs exhibit natural resilience to benign faults that do not affect output integrity, they remain vulnerable to critical faults,

Authorized licensed use limited to: MIT. Downloaded on March 16,2025 at 19:19:59 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2025.3544537

Problem	Dataset	Model	Precision	Injection level	P_{CNN}	Source
Classification	ImageNet	NiN	FP32/FP16	Flip-flops and SRAMs	0.0162/0.0028	[2]
Classification	ImageNet	AlexNet	FP32/FP16	Flip-flops and SRAMs	0.0038/0.0032	[2]
Classification	CIFAR-10	AlexNet	FxP_32	Neuron output	0.0032	[5]
Classification	ImageNet	AlexNet	INT8	Neuron output	0.008	[4]
Classification	ImageNet	GoogleNet/ResNet50	INT8	Neuron output	0.003/0.0025	[4]
Classification	ImageNet	ShuffleNet/SqueezeNet	INT8	Neuron output	0.007/0.0035	[4]
Classification	ImageNet	VGG19	INT8	Neuron output	0.0015	[4]
Classification	ImageNet	VGG16	FxP_32	Neuron output	0.0104	[5]
Classification	Traffic Sign	VGG11	FxP_32	Neuron output	0.001	[5]
Detection/Driving	Driving	DAVE/Common.ai	FxP_32	Neuron output	0.0012/0.00092	[5]
Detection	Driving	A DriveWorks model	N/S	Flip-flops and SRAMs	Very low	[6]

TABLE I: Estimates of the probabilities with which a low-level transient error propagates to different CNN outputs.

which can produce erroneous outputs. Ranger automatically transforms DNNs to selectively constrain value ranges, effectively converting large deviations caused by critical faults into benign faults. This enables the DNNs to utilize their inherent fault tolerance. Our evaluation across eight DNNs shows that Ranger improves error resilience by 3x to 50x, with no accuracy loss and negligible performance overhead.

B. Selective Latch Hardening (SLH)

Protecting the latches in the datapath can be vital for highly dependable systems as they become the reliability bottleneck once all buffers are protected (e.g., by ECCs). There have been several hardened latch designs that differ in their overheads and levels of protection. For example, Strike Suppression (RCC) technique provides 6.3× FIT reduction with 1.15× area overhead. Reundant Node (SEUT) and Triplicated (TMR) are two other techniques that offer 37× and 1,000,000× protection with 2× and 3.5× area overhead, respectively.

Since we observed and characterized asymmetric SDC sensitivity in different bits, we can leverage this model to selectively harden each latch using the most efficient hardening technique to achieve sufficient error coverage at a low cost. Our experiments real a super-linear FIT rate reduction by protecting a faction of the latches in the design, i.e., we get a high enough FIT reduction by protecting a smaller fraction of the latches. When we employ the use of different latch hardening methods that offer different protection vs. per-latch overhead characteristics, we observed even better results. For example, applying the three hardening techniques together can reduce the latch FIT rate by 100x, while incurring only about 20% latch area overheads.

Additionally, the non-uniform vulnerability of individual bits provides insights into the monotonicity of SDC locality. In our follow-up work, we observe that widely-used ML computations often exhibit monotonic behavior. Building on this, we introduce BinFI [8], an efficient fault injector designed to identify safety-critical bits in ML applications. This allows us to approximate the error propagation behavior of an ML application as a monotonic function. BinFI employs a binary search-like fault injection technique to efficiently pinpoint safety-critical bits while assessing overall resilience. Our results show that BinFI identifies 99.56% of safety-critical bits with 99.63% precision, significantly outperforming random fault injection techniques at much lower cost.

VI. Impact

The original publication [2] has been cited over 599 times as of October 2024, according to Google Scholar, making it one of the top 10 most cited papers in the history of the SC conference within just seven years of its release, as per ACM statistics [9]. This work has had a significant impact on the following four areas of research.

Resilience Studies of DNN accelerators and applications: Prior to our work, it was believed that DNNs were largely resilient to soft errors due to their built-in tolerance to perturbations. We demonstrated in our work that this is not necessarily the case, and even *a single transient fault can cause safety violations in DNNs*. Our work has inspired several follow-up studies that have examined the error resilience of DNN accelerators from various perspectives, considering different applications and fault models. For example, ARES [10] systematically characterized the impact of DNN accelerator parameters on resilience. Other studies have explored the effects of transient and permanent faults on accelerators [11] and sensor faults in DNN applications deployed in AVs [12]. Moreover, research has confirmed that a single transient fault can lead to safety violations in DNNs [13]. Finally, other studies [14]–[16] have also underscored the critical need to assess the reliability of AI/ML platforms.

Fault Injection Techniques for DNNs: Another area of research that was spawned by our paper is fault injection (FI) into DNN accelerators and applications. Our work was among the first to perform FI into DNN accelerators running real DNN workloads. Subsequently, several specialized tools and techniques for performing FI into DNN frameworks and applications have been developed. PyTorhcFI, TensorFI, and BinFI are few such examples. Many of these techniques leverage specific properties of DNNs to optimize the fault space, which can be probibitively large. Our work was the first to propose optimization techniques for FI into DNNs.

Hardware Techniques for DNN Resilience: Another novel aspect of our paper was the use of selective latch hardening techniques to protect critical latches in the DNN accelerator. Our FI experiments allowed us to identify the critical latches, i.e., the latch elements that are most sensitive to soft errors.

Authorized licensed use limited to: MIT. Downloaded on March 16,2025 at 19:19:59 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2025.3544537

We then employed latch hardening techniques to them subject to a total area overhead bound. We found that we were able to reduce the FIT rate by 37× with a 2× area overhead. This result has influenced other publications (e.g., [17]), and the design of commercial products as well (details are confidential).

Software Techniques for DNN Resilience: Finally, our technique has influenced the design of software-based techniques for DNN resilience. In our paper [2], we had proposed symptom-based detectors to identify SDC-causing errors in intermediate layers of the DNN. Our work has influenced other work such as Clipper [18], Ranger [7] and FT-ClipACT [19], which aim to provide protection from Rowhammer attacks, transient faults and permanent faults respectively for DNN applications. The common theme in these techniques is to leverage the properties of the DNN's intermediate values being bounded (as our study found), and check for conformance to these properties at runtime to detect faults. Schorn et al. [20] designed error resilient DNN accelerators based on our observation that network structure influences error resilience.

REFERENCES

[1] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of iso 26262: Using machine learning safely in automotive software," *arXiv preprint arXiv:1709.02435*, 2017.

[2] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. ACM, 2017.

[3] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, p. 367–379, jun 2016.

[4] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, and S. K. S. Hari, "Pytorchfi: A runtime perturbation tool for dnns," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2020, pp. 25–31.

[5] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, "Binfi: An efficient fault injector for safety-critical machine learning systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356177

[6] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley, and Y. Huang, "Resiliency of automotive object detection networks on gpu architectures," in *IEEE International Test Conference (ITC)*, 2019, pp. 1–9.

[7] Z. Chen, G. Li, and K. Pattabiraman, "A low-cost fault corrector for deep neural networks through range restriction," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021.

[8] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, "Binfi: An efficient fault injector for safety-critical machine learning systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019.

[9] "Sc paper citation history," https://dl.acm.org/topic/conference-collections/sc?sortBy=citedstartPage=0pageSize=50, accessed: 2025-01-29.

[10] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A Framework for Quantifying the Resilience of Deep Neural Networks," in *Proceedings of the Design Automation Conference*, 2018.

[11] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 270–281.

[12] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "AVFI: Fault Injection for Autonomous Vehicles," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, 2018.

[13] Y. Ibrahim, H. Wang, J. Liu, J. Wei, L. Chen, P. Rech, K. Adam, and G. Guo, "Soft errors in dnn accelerators: A comprehensive review," *Microelectronics Reliability*, vol. 115, p. 113969, 2020.

[14] P. Rech, "Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions," *IEEE Transactions on Nuclear Science*, 2024.

[15] F. Su, C. Liu, and H.-G. Stratigopoulos, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives," *IEEE Design & Test*, vol. 40, no. 2, pp. 8–58, 2023.

[16] A. Ruospo, E. Sanchez, L. M. Luza, L. Dilillo, M. Traiola, and A. Bosio, "A survey on deep learning resilience assessment methodologies," *Computer*, vol. 56, no. 2, pp. 57–66, 2023.

[17] D. Stutz, N. Chandramoorthy, M. Hein, and B. Schiele, "Bit error robustness for energy-efficient dnn accelerators," in *Proceedings of Machine Learning and Systems*, vol. 3, 2021, pp. 569–598.

[18] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *USENIX Conference on Security Symposium*. USA: USENIX Association, 2019, p. 497–514.

[19] L.-H. Hoang, M. A. Hanif, and M. Shafique, "Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *Design, Automation & Test in Europe Conference And Exhibition (DATE)*, 2020, pp. 1241–1246.

[20] C. Schorn, T. Elsken, S. Vogel, A. Runge, A. Guntoro, and G. Ascheid, "Automated design of error-resilient and hardware-efficient deep neural networks," *Neural Computing and Applications*, vol. 32, 2020.

BIBLIOGRAPHY

Guanpeng Li is an assistant professor in the Computer Science Department at the University of Iowa, USA. Li received a Ph.D. degree in electrical and computer engineering from the University of British Columbia, Canada. He is a member of the IEEE. Contact him at guanpeng-li@uiowa.edu.

Siva Kumar Sastry Hari is a principal research scientist in the architecture research group at NVIDIA, Santa Clara, CA, USA. Hari received a Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, IL, USA. He is a senior member of IEEE. Contact him at shari@nvidia.com.

Michael B. Sullivan is a senior research scientist in the architecture research group at NVIDIA, Santa Clara, CA, USA. Sullivan received a Ph.D. degree in computer architecture from the University of Texas at Austin, TX, USA. Contact him at misullivan@nvidia.com.

Timothy Tsai is an independent scholar. Until recently, he was a senior research scientist at NVIDIA, Santa Clara, CA, USA. Tsai received a Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, IL, USA. Contact him at timothytsai@gmail.com.

Karthik Pattabiraman is a professor of electrical and computer engineering at the University of British Columbia, Canada. He received his Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, IL, USA. He is a senior member of the IEEE. Contact him at karthikp@ece.ubc.ca.

Joel S. Emer is a senior distinguished research scientist in the computer architecture group at NVIDIA, Westford, MA, USA and a professor of the practice at the Massachusetts Institute of Technology, MA, USA. Emer received his Ph.D. degree from the University of Illinois at Urbana-Champaign. He is an IEEE fellow. Contact him at jemer@nvidia.com.

Authorized licensed use limited to: MIT. Downloaded on March 16,2025 at 19:19:59 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2025.3544537

Stephen W. Keckler is the vice president of architecture research at NVIDIA Corporation, Santa Clara, CA, USA and an adjunct professor of computer science at the University of Texas at Austin, Austin, TX, USA. Keckler received a Ph.D. degree in computer science from the Massachusetts Institute of Technology. He is an IEEE Fellow. Contact him at skeckler@nvidia.com

Authorized licensed use limited to: MIT. Downloaded on March 16,2025 at 19:19:59 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.