Deep Learning Reliability: Towards Mitigating Reliability Threats in Deep Learning Systems by Exploiting Intrinsic Characteristics of DNNs

Muhammad Abdullah Hanif and Muhammad Shafique

1 Introduction

Deep Neural Networks (DNNs) have emerged as a promising set of algorithms for solving complex AI problems such as image classification, object detection and localization, semantic segmentation, speech recognition, language translation, and video processing [1]. The state-of-the-art performance of these models has laid the foundation for DNNs to be used in safety-critical applications as well such as autonomous driving [2] and smart healthcare [3]. Driven by the compute- and memory-intensive nature of DNNs and the need for deploying such high-accuracy models in resource-constrained edge devices, a significant amount of research has been carried out towards designing specialized hardware accellerators that can enable low-cost DNN inference at the edge. Some prominent DNN hardware accelerators include Eyeriss [4], MAERI [5], TPU [6], and MPNA [7].

On the one end, these DNN hardware accelerators promise low-cost and real-time execution of DNNs; however, on the other end, they bring some critical reliability challenges that can significantly degrade the performance and dependability of the system. These reliability threats are specifically important to address for safety-critical systems, as even a single fault at a critical location in such systems can result in severe consequences. For example, in the case of autonomous driving vehicles, a critical fault in the perception unit can result in the misclassification of a traffic sign (e.g., a stop sign) which can lead to a fatal accident. Such faults can even lead to total disruption of traffic service if the vehicle is connected to the traffic infrastructure in a smart city. An overview of different hardware-induced reliability threats, how they

M. A. Hanif (✉) · M. Shafique
New York University Abu Dhabi, Abu Dhabi, UAE
e-mail: mh6117@nyu.edu; muhammad.shafique@nyu.edu

S. Pasricha, M. Shafique (eds.), *Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing*, https://doi.org/10.1007/978-3-031-40677-5_21

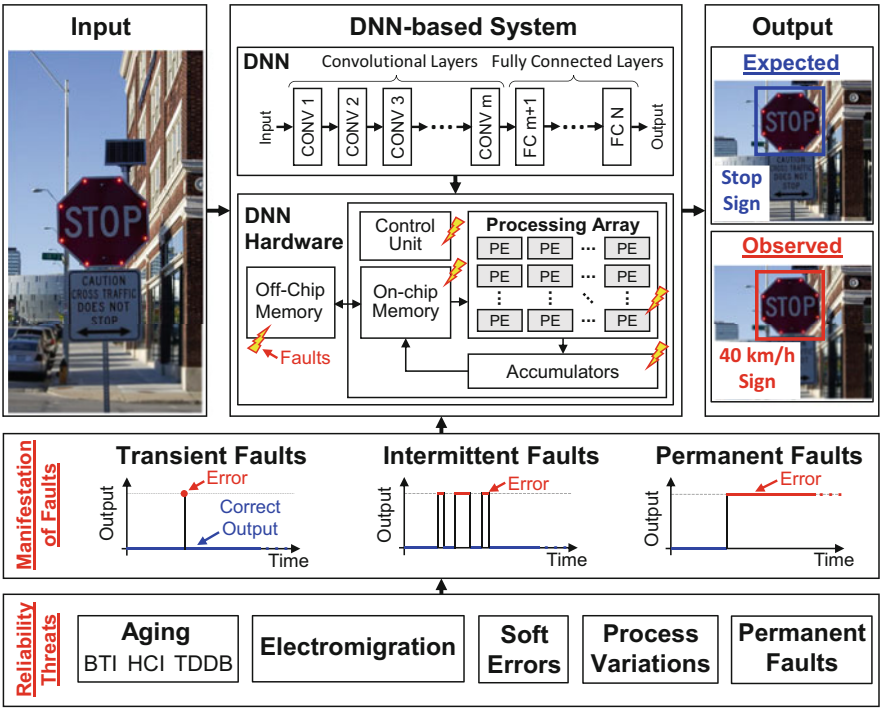

Fig. 1 Overview of different reliability threats and their repercussions. The picture used in the figure is from the COCO dataset

manifest in a system and how can they impact the functionality of a DNN inference is shown in Fig. 1.

Reliability Threats Gradual progress in the fabrication process and the desire for extreme-performance devices has lead us to the era of nano-scale devices. However, electronic devices fabricated using nano-scale technology face various reliability issues. Some of these issues are associated with the limitations of the fabrication process while others are associated with the extreme sizes of the transistors. The following text provides a brief introduction to different hardware-induced reliability threats that can degrade the performance of a system.

- **Soft Errors** are transient bit-flips caused by high-energy particle strikes. These particles can be alpha particles emitted from the impurities in the packaging materials of the chip or neutrons from cosmic radiations [8]. These soft errors can propagate all the way to application layer of a system and results in significant accuracy degradation. External factors such as temperature and altitude can have a notable impact on the Soft Error Rate (SER).
- **Aging** of nano-scale electronic devices occurs due to various physical phenomena such as Bias Temperature Instability (BTI), Time-Dependent Dielectric

Breakdown (TDDB), Hot Carrier Injection (HCI), and Electromigration (EM). It typically results in increased threshold voltage (V_{TH}) [9] or breakdown of dielectric and wires. In the early stages, aging manifests as timing errors in a system, and later it can even transform into permanent faults. Typically, large guardbands are added to the operating frequency of a circuit to ensure reliable operation. Similar to other reliability threats, aging rate also increases with temperature.

- **Process Variations** are variations in the hardware characteristics of transistors that occur due to imperfections in the fabrication process [10]. In general, these variations manifest as timing errors in a system and, therefore, are addressed by adding guardbands, i.e., either by increasing the supply voltage or reducing the operating frequency of the device. Extreme variations can even lead to permanent faults, which can have a significant impact on the manufacturing yield.

Apart from aggressive guard-banding, a number of other techniques have also been proposed to improve the resilience of systems against reliability threats. However, most of these techniques are based on redundancy, e.g., Error Correction Codes (ECC), Dual Modular Redundancy (DMR) [11], and Triple Modular Redundancy (TMR) [12]. The redundancy-based techniques, on the one hand, are highly effective; however, on the other hand, they lead to high performance and energy overheads. This together with the compute- and memory-intensive nature of DNNs makes such techniques infeasible for DNN-based systems. *Therefore, alternate techniques are required that can improve the resilience of DNN-based systems against hardware-induced reliability threats at minimal cost.*

In the following section, we present a brief *overview of DNNs and DNN hardware accelerators*. Then, in Sect. 3, we present an overall *methodology for building reliable DNN inference systems* and also discuss individual *low-cost techniques for mitigating permanent faults, aging, and soft errors*.

2 Preliminaries

2.1 Deep Neural Networks

A neural network can be visualized as a network of interconnected neurons (see Fig. 2a), where a neuron is the fundamental computational unit of the network. The functionality of a typical neuron used in neural networks can be described using the following equation:

$$Out = f\left(\sum_{i=0}^{N} W_i \times A_i + b\right) \tag{1}$$

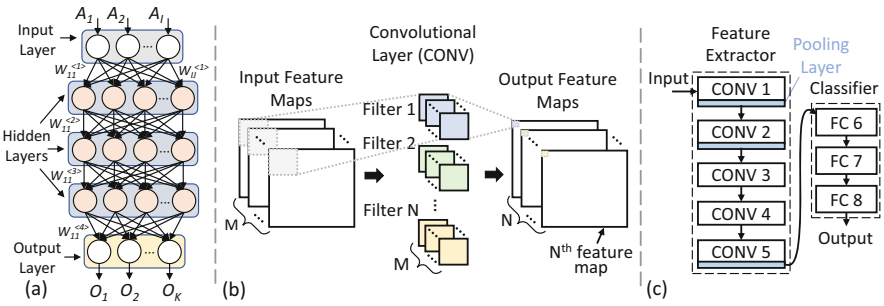

Fig. 2 (**a**) An example MLP network. (**b**) Convolutional layer. (**c**) Architecture of a convolutional neural network

where W_i represents the ith weight, A_i represents the ith activation, N represents the number of weights (and activations) in the input vectors, b represents the bias, and $f(.)$ represents the activation function (a non-linear function to introduce non-linearity). The neurons are typically arranged in layers, and a neural network having more than three layers is termed as a Deep Neural Network (DNN). Figure 2a shows an example of a Fully Connected Neural Network (FCNN), in which all the neurons in each layer are connected with all the neurons in the previous layer and the next layer. A FCNN is also known as a Multi-layer Perceptron (MLP).

Different types of DNNs have been proposed in the literature. Apart from FCNNs, Convolutional Neural Networks (CNNs) are also widely known. CNNs are mainly used to process spatially or temporally correlated data such as images and videos. A CNN is generally composed of multiple convolutional layers and fully connected layers, see Fig. 2c. The convolutional layers are used for extracting features from the input while the fully connected layers are responsible for the final classification based on the features extracted by the convolutional layers. Figure 2b shows a detailed view of a convolutional layer.

Various other types of DNNs also exist, for example, Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), and Graph Neural Networks (GNNs). However, as most of the reliability studies have been demonstrated on MLPs and CNNs, this chapter also considers the same to highlight the effectiveness of different methods.

2.2 DNN Hardware Accelerators

To enable the deployment of DNNs in resource-constrained edge devices, DNN hardware accelerators are employed. Various accelerator designs have been proposed in the literature, where each design supports a specific set of dataflows in

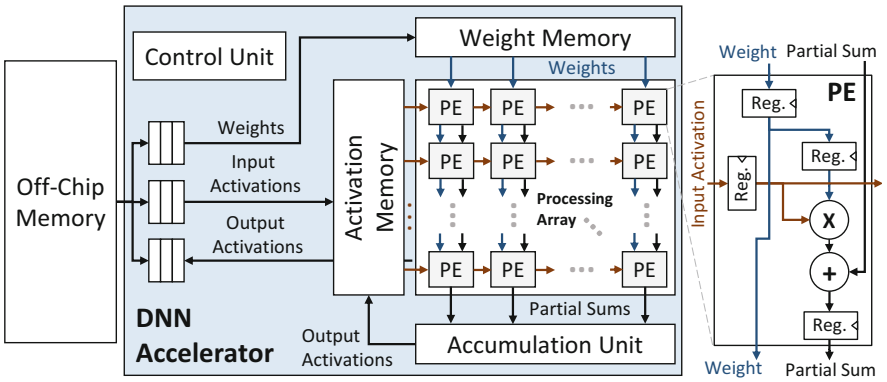

Fig. 3 A systolic-array-based DNN hardware accelerator

a more efficient manner, e.g., see [4, 6, 7, 13]. An overview of a DNN hardware accelerator is shown in Fig. 3. A DNN accelerator is mainly composed of Processing Elements (PEs), partial sum accumulation units, and on-chip memory. Each PE contains some arithmetic modules and some registers. The arithmetic modules are for performing the MAC operations involved in the DNN execution and the registers are for storing weights, activations, and partial sums. The exact configuration of the PEs and their connectivity in the accelerator are based on the supported dataflows.

The accelerator shown in Fig. 3 is a systolic-array-based design composed of homogeneous PEs, similar to the design in [6]. The PEs are connected in a 2D-grid-like manner. The accelerator follows a weight-stationary dataflow, where weights are loaded into the array (through vertical channels) and kept stationary during operations. Note, weights from the same filter/neuron are mapped on the same column, while weights from multiple filters/neurons can be mapped simultaneously by mapping different filters/neurons (from the same DNN layer) to different columns. After the weights have been loaded inside the PEs, the input activations are fed from the left and are multiplied with the weight values to generate products. Each product is then added with the partial sum from the above PE, and the updated partial sum is then passed downstream to be used in the next cycle by the downstream PE. As the size of the processing array is usually limited, a dot-product operation is broken down into multiple chunks (based on the size of the processing array) and a single chunk is mapped onto the array at a time. The partial sums generated by the array are stored in the accumulation units to be added with the corresponding partial sum/s from the other chunks (if any). A more detailed explanation of the architecture can be found in [14].

3 Reliable Deep Learning

This section presents a systematic methodology for building reliable systems for DNN-based applications. The section also highlights the impact of different types of reliability threats on the application-level accuracy of DNNs and presents different low-cost techniques for improving the resilience of DNN-based systems against hardware-induced reliability threats at minimal cost.

3.1 A Systematic Methodology for Building Reliable DNN Systems

Figure 4 presents an overview of our systematic design methodology for building reliable systems for DNN-based applications. The methodology is composed of different design-time steps, post-fabrication steps, and run-time steps.

The **design-time** steps focus on building a hardware accelerator capable of mitigating all types of hardware-induced reliability threats. Towards this, first, a baseline hardware accelerator is designed based on the user-defined performance constraints. Then, the additional circuitry required for mitigating permanent faults (see Sect. 3.3), aging (see Sect. 3.4), and soft errors (see Sect. 3.5) is added to the accelerator design. Note, to achieve high resilience against reliability threats at a low cost, the error resilience of a representative set of DNNs is taken into consideration. The error resilience helps estimate the extent of protection required against each threat. After reinforcing the accelerator with the additional circuitry, the hardware is synthesized using reliability-aware synthesis techniques, for example, using selective hardening where vulnerable nodes in the hardware are hardened using node-level redundancy [15].

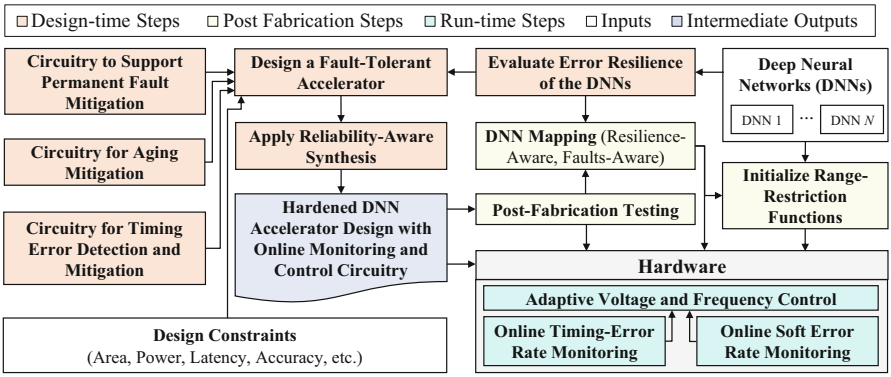

Fig. 4 Our methodology for designing reliable hardware for DNN-based applications (adapted from [16])

The **post-fabrication** steps focus on exploiting the information collected through post-fabrication testing (e.g., fault maps and process variation maps of the fabricated hardware) to define DNN mapping policies. The fault-aware and variation-aware mapping of DNNs can significantly reduce the negative impact of faults and variations on the application-level accuracy and performance characteristics of the system (see Sect. 3.3). The mapping information together with fault maps are also used by range initialization block for soft error mitigation.

The **run-time** steps focus on trading energy for reliability through adaptive voltage and frequency scaling. The online error rate monitoring blocks monitor the frequency of errors, and the system then responds by increasing the supply voltage or decreasing the operating frequency to reduce the frequency of errors, whenever required. Software-level redundancy can also be employed to improve the reliability by processing critical layers (or neurons) multiple times.

3.2 Resilience of DNNs to Reliability Threats

Occasionally, DNNs are assumed to be inherently resilient to errors [17]. However, studies have shown that DNNs respond differently to different types of errors. Errors that occur at critical locations in the system can significantly degrade the application-level accuracy of DNNs while errors at non-critical locations do not impact the accuracy much. This section presents the resilience of DNNs to different types of reliability threats. The section also highlights the importance of low-cost fault-mitigation techniques for dependable performance.

3.2.1 Resilience of DNNs to Permanent Faults

This section presents an empirical analysis from [14] highlighting the impact of stuck-at permanent faults in the computational array of a systolic-array-based DNN accelerator (shown in Fig. 3) on the application-level accuracy of DNNs. The analysis is performed for two different networks trained on two different datasets, i.e., the MNIST and TIMIT datasets. The details of the DNN architectures used are presented in Table 1. For this analysis, a systolic array of 256 × 256 MAC units synthesized using 45nm OSU PDK to generate a gate-level netlist is considered. For permanent faults, stuck-at faults are inserted randomly at internal nodes in the netlist.

Table 1 Datasets and the corresponding DNNs used for analyzing the impact of permanent faults

Dataset	Network architecture	Accuracy (%)
MNIST [18]	Fully connected (L1-L4): 784×256×256×256×10	98.15
TIMIT [19]	Fully connected (L1-L4): 1845×2000×2000×2000×183	73.91

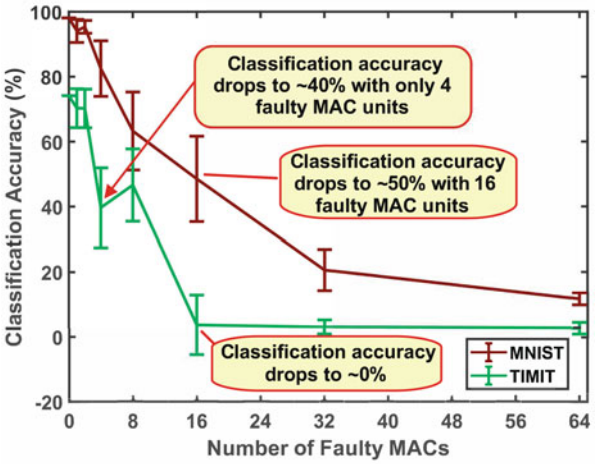

Fig. 5 Impact of Stuck-at-Faults in a systolic-array-based DNN accelerator on the classification accuracy of two different DNNs [20]

Figure 5a shows the impact of using a faulty DNN accelerator on the classification accuracy of two different DNNs. As can be observed from the figure, for both the DNNs, the classification accuracy decreases sharply with the increase in faulty PEs. For example, in the case with the number of faulty PEs equals 16, the average accuracy of the DNN trained on the TIMIT dataset falls to zero, while the accuracy of the DNN trained on the MNIST dataset falls to around 50%. Note that 16 PEs is equivalent to around 0.025% of total PEs in a 256×256 array. This shows that even a very small number of permanent faults in a DNN-based system can significantly degrade the system's performance. This analysis clearly highlights the need for permanent fault mitigation to increase the manufacturing yield of DNN accelerators, as faulty hardware having permanent faults cannot produce reliable results.

3.2.2 Resilience of DNNs to Timing Faults

Timing failures in high-performance nano-scale CMOS devices are a significant reliability concern. These errors arise due to various reasons, e.g., power supply disturbance, crosstalk, process variations, and aging. Moreover, the operating conditions, such as supply voltage, also significantly affect the frequency of timing failures. This section highlights the impact of timing errors on the classification accuracy of a DNN trained on the MNIST dataset using the analysis presented in [14]. The DNN architecture is presented in Table 1 and the considered hardware accelerator is shown in Fig. 3. To illustrate the impact of timing errors on DNN accuracy, [14] considered a Timing Error Propagation (TEP) case where timing errors are allowed to propagate to the output. The timing errors are introduced in

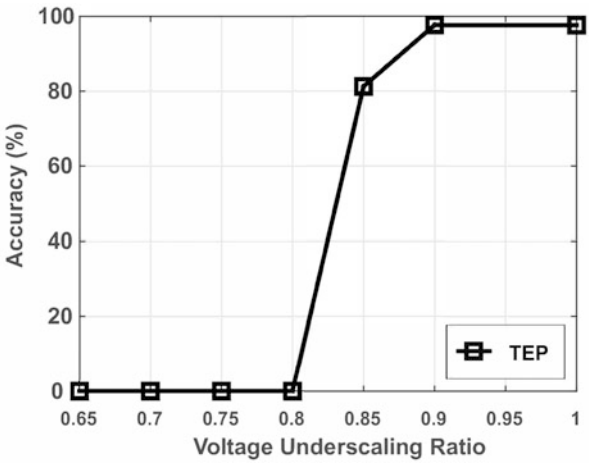

Fig. 6 Impact of timing errors induced through voltage under-scaling on the classification accuracy of a DNN trained on the MNIST dataset [14]

the accelerator array through voltage under-scaling. Figure 6 shows the impact of voltage under-scaling on the classification accuracy of the DNN. Note, as the supply voltage of the array is reduced, timing errors start increasing. Figure 6 clearly shows that in the TEP case, the accuracy of the DNN starts decreasing abruptly as the fault rate starts increasing. Therefore, to ensure reliable execution of DNNs it is essential to mitigate timing errors.

3.2.3 Resilience of DNNs to Memory Faults

To illustrate the impact of memory faults on the accuracy of DNNs, Hanif et al. [16] presented an analysis where they injected random faults in the weight memory of a DNN accelerator. The analysis showed that faults at higher significance bit-locations in the weights can drastically reduce the application-level accuracy of DNNs while faults at lower significance bit-locations do not impact the accuracy much. Moreover, the analysis also showed that the accuracy drop increases sharply with the increase in the fault rate. They also studied the impact of different types of faults individually and showed that 0-to-1 bit-flips have a more severe impact compared to 1-to-0 bit-flips, as 0-to-1 bit-flips at higher significance bit-locations can significantly increase the weight values. This conclusion is also in line with the dropout [21] and DropConnect [22] concepts in the sense that 1-to-0 bit-flips push the weight values toward zero, which is equivalent to dropout at a fine-grained level. Note that the conclusion may differ for different data representation formats. Similar fault injection studies have also been conducted in [23] and [24] to analyze the resilience of DNNs.

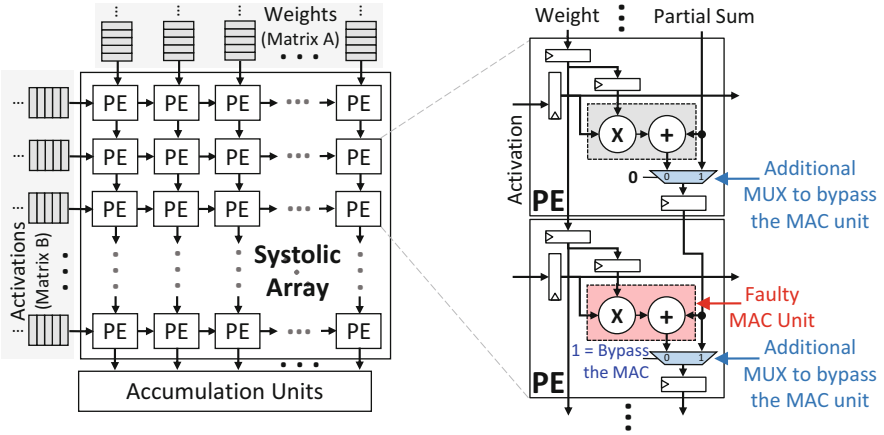

Fig. 7 Modified systolic-array design for permanent fault mitigation through fault-aware pruning

3.3 Permanent Fault Mitigation

As highlighted in Sect. 3.2.1, permanent faults can restrain a system from generating correct outputs. Therefore, it is essential to mitigate such errors to ensure high manufacturing yield. **Fault-Aware Pruning (FAP)** [20] has been proposed to mitigate permanent faults in the computational array of a systolic-array-based DNN accelerator. The key idea behind this approach is to replace critical faults with non-critical faults. In [20], this is achieved through dropping the computations mapped onto the faulty components, as dropping a small percentage of computations in DNNs do not impact the accuracy much, see dropout [21] and DropConnect [22] concepts.

Figure 7 illustrates the modified systolic-array design proposed in [20] for mitigating permanent faults in the MAC units of a systolic-array-based DNN accelerator. As illustrated in the figure, each PE is equipped with an additional MUX to bypass the MAC unit inside the PE. In case a fault is detected in the MAC unit of a PE during post-fabrication testing, the corresponding MUX is configured to bypass the faulty MAC unit. Note, the systolic-array architecture shown in Fig. 7 follows a weight-stationary dataflow where weights from the same neuron/filter are mapped onto the same column and are kept stationary inside the PEs during execution. Hence, the bypass operation corresponds to pruning of the weights mapped onto the faulty units.

To improve the performance of FAP, **Fault-Aware Pruning + Training (FAP+T)** is proposed [20]. The technique is based on the observation that DNNs are typically over-parameterized and pruning a small set of weights during training do not affect the final accuracy much [25]. Figure 8 presents a general flow for training a DNN against permanent faults. As highlighted in the figure, the fault map extracted through post-fabrication testing of the fabricated chip (along with the DNN mapping

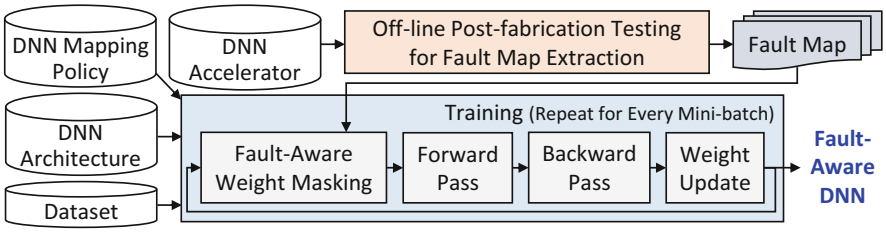

Fig. 8 General flow adopted for fault-aware retraining

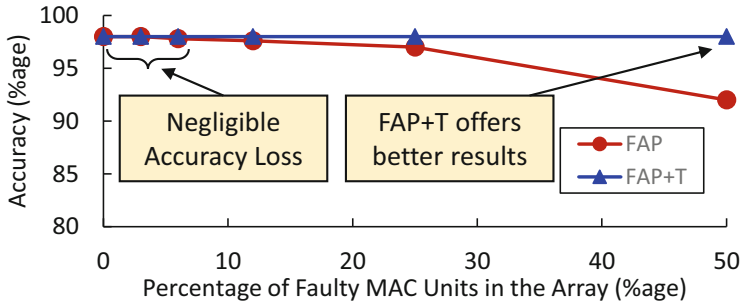

Fig. 9 Impact of using FAP and FAP+T on the classification accuracy of a fully connected DNN trained on the MNIST dataset at different fault rates [20]

policy) is used to force the weights to be mapped onto faulty MAC units to zero in each iteration of the training loop. This enables the DNN to adapt to the faults in the system and offer better performance compared to simple FAP.

Figure 9 highlights the effectiveness of FAP and FAP+T using a fully connected DNN trained on the MNIST dataset. As can be seen from the figure, both FAP and FAP+T help improve the resilience of the DNN against permanent faults in the computational array of a DNN accelerator; however, FAP+T offers better results at higher fault rates, i.e., negligible accuracy loss even when 50% of the total MAC units are faulty.

Although FAP+T is highly effective against permanent faults, its main drawback is that it involves retraining the given DNN, which may not be possible under some scenarios due to the lack of computational resources or a comprehensive training dataset. To address this issue, **Fault-Aware Mapping (FAM)** has been proposed [26]. FAM employs a saliency-driven approach to determine the mapping of the given pre-trained DNN for the given faulty chip. Figure 10 shows the general flow for applying FAM. First, the saliency of each DNN weight is computed using the L1 or L2-norm. Then, using an optimization algorithm and the knowledge of the faults, a mapping policy is determined that leads to the minimum (or a lower) sum of saliency of weights to be pruned due to permanent faults in the computational array of the DNN accelerator. In the end, the parameters of the DNN are rearranged according to the mapping policy (wherever possible) to avoid any run-time data

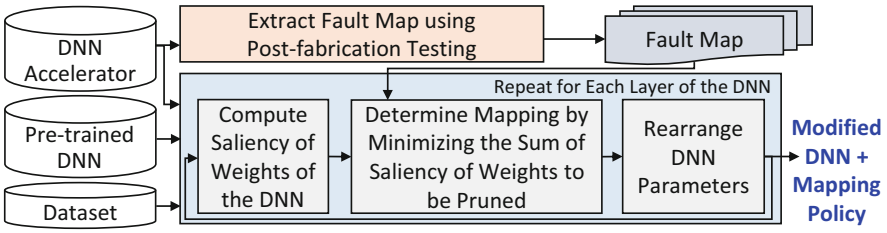

Fig. 10 Fault-aware mapping methodology

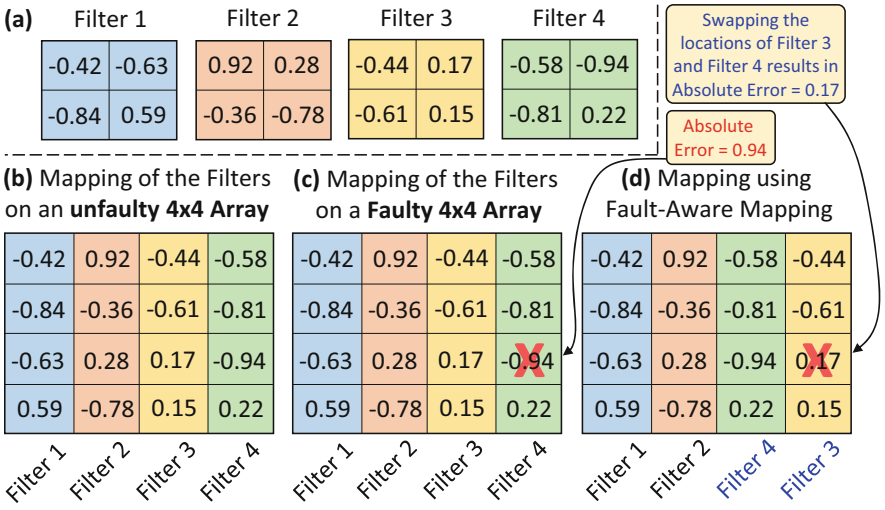

Fig. 11 An illustrative example of fault-aware mapping on a 4×4 systolic array

rearrangement operations. The output DNN and the mapping policy are then used together with FAP for reliable DNN inference. Figure 11 presents an example of how FAM can help in reducing the impact of permanent faults when used with FAP, and Fig. 12 highlights the effectiveness of the approach when used for the VGG11 network trained on the ImageNet dataset. Figure 12 clearly highlights that FAM can be employed without retraining specifically for low-to-moderate fault rates to get better results than only FAP.

3.4 Timing Error Mitigation

Aging in CMOS devices manifests as timing errors. These errors can have a drastic impact on the performance of a DNN interference system, as highlighted in Sect. 3.2.2. Conventional techniques such as aggressive voltage and frequency guard-banding result in significant energy and/or latency overheads. Therefore, it is

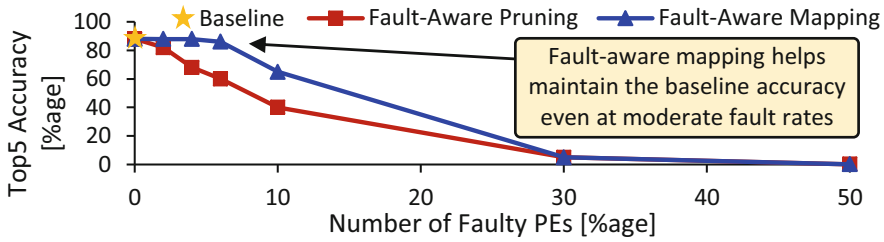

Fig. 12 Impact of fault-aware mapping on the classification accuracy of the VGG11 network trained on the ImageNet dataset

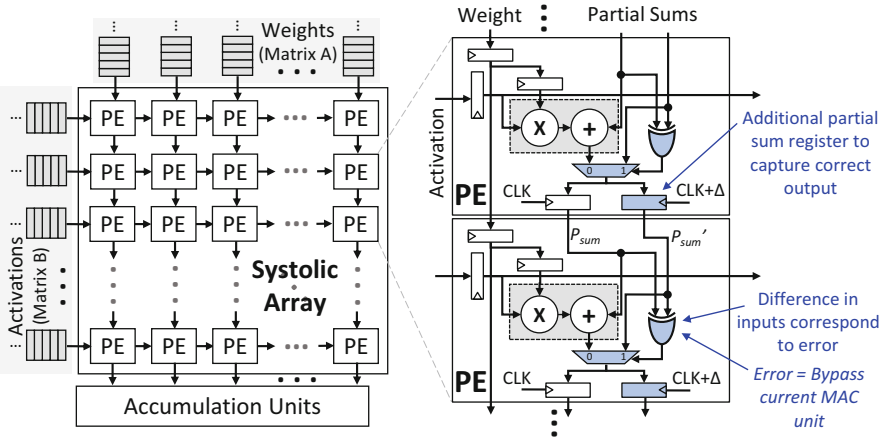

Fig. 13 Architectural modifications required in PEs of a systolic-array-based DNN accelerator to realize TE-Drop

crucial to address these errors at low cost to ensure reliable and resource-efficient DNN execution.

To address timing errors in the computational array of a systolic-array-based DNN accelerator at a low cost, Zhang et al. proposed TE-Drop [14]. TE-Drop works on the principle that the contribution of each individual MAC operation to the overall output of a DNN is very small. Therefore, a small percentage of MAC operations can be dropped without affecting the application-level accuracy of the system. To detect timing errors in the computational array, TE-Drop utilizes Razor flip-flops; however, instead of re-executing the erroneous MAC operation, it captures the correct MAC output in an alternate partial sum register operating on a delayed clock. Then, the succeeding PE is bypassed to feed the correct MAC output back into the computational flow. Figure 13 presents the architectural modifications required to realize the concept.

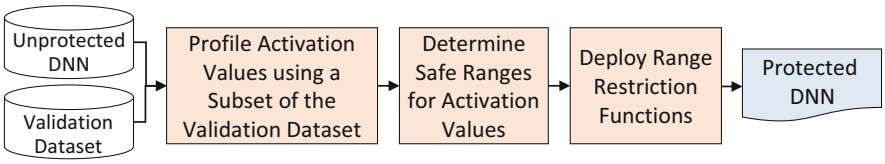

Fig. 14 General flow of range-restriction-based soft error mitigation techniques

3.5 Soft Error Mitigation

As highlighted in Sect. 3.2.3, soft errors at critical locations in a DNN-based system can significantly degrade the application-level accuracy of the system [16, 27]. Therefore, it is crucial to address these errors to ensure reliable DNN execution. Although conventional redundancy-based techniques (e.g., DMR and TMR) are highly effective against soft errors, they result in extreme overheads due to the compute-intensive nature of DNNs. Therefore, specialized low-cost techniques are designed to improve the resilience of these systems against soft errors.

To mitigate soft errors in SRAM-based on-chip memory, Azizimazreah et al. proposed a zero-biased SRAM cell design that has a higher tendency to switch to '0' in case an error occurs in the cell [28]. The intuition behind this design is that 0-to-1 bit-flips in DNNs result in a higher accuracy loss compared to 1-to-0 bit-flips. To mitigate soft errors in the computational array of a DNN accelerator, researchers have proposed different range-restriction techniques, e.g., Ranger [27], that bound the range of the intermediate activation values to a pre-computed safe range. The intuition behind these techniques is that soft errors can result in large activation values that may propagate to the output and impact the classification result. Therefore, abnormally large activation values that fall out of the normal range can be classified as erroneous, and dropping such values can mitigate soft errors due to the inherent resilience of DNNs to pruning. Figure 14 presents the general flow of range-restriction techniques. A similar technique is proposed in [29] for mitigating soft errors in the on-chip memory of DNN accelerators. Apart from the above-mentioned techniques, algorithm-based fault tolerance, such as checksum-based error detection and correction have also been proposed to mitigate soft errors in DNN systems at a low cost [30].

4 Conclusion

The state-of-the-art performance of DNNs for complex AI problems has led to their adoption for safety-critical applications as well. However, these systems have strict robustness constraints that are challenged by the hardware-induced reliability threats introduced due to the use of specialized DNN accelerators. The compute- and memory-intensive nature of DNNs prevents the use of redundancy-based techniques

for mitigating these threats. Towards this, this chapter covered different low-cost techniques for improving the resilience of DNN inference systems against soft and timing errors. The chapter also covered different techniques for mitigating permanent faults. Moreover, the chapter also discussed a holistic methodology for mitigating all types of reliability threats at low overhead costs.

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
2. Fink, M., Liu, Y., Engstle, A., Schneider, S.A.: Deep learning-based multi-scale multi-object detection and classification for autonomous driving. In: Fahrerassistenzsysteme 2018, pp. 233–242. Springer, Berlin (2019)
3. Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., Dean, J.: A guide to deep learning in healthcare. Nat. Med. **25**(1), 24 (2019)
4. Chen, Y., et al.: Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. IEEE J. Emerg. Sel. Topics Circuits Syst. (2019)
5. Kwon, H., Samajdar, A., Krishna, T.: Maeri: Enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 461–475. ACM, New York (2018)
6. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1–12. IEEE, Piscataway (2017)
7. Hanif, M.A., Putra, R.V.W., Tanvir, M., Hafiz, R., Rehman, S., Shafique, M.: MPNA: A massively-parallel neural array accelerator with dataflow optimization for convolutional neural networks (2018). arXiv preprint arXiv:1810.12910
8. Baumann, R.C.: Radiation-induced soft errors in advanced semiconductor technologies. IEEE T-DMR **5**(3), 305–316 (2005)
9. Kang, K., et al.: NBTI induced performance degradation in logic and memory circuits: how effectively can we approach a reliability solution? In: ACM/IEEE ASP-DAC, pp. 726–731 (2008)
10. Raghunathan, B., Turakhia, Y., Garg, S., Marculescu, D.: Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In: 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 39–44. IEEE, Piscataway (2013)
11. Vadlamani, R., Zhao, J., Burleson, W., Tessier, R.: Multicore soft error rate stabilization using adaptive dual modular redundancy. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 27–32. European Design and Automation Association (2010)
12. Lyons, R.E., Vanderkulk, W.: The use of triple-modular redundancy to improve computer reliability. IBM J. Res. Dev. **6**(2), 200–209 (1962)
13. Lu, W., Yan, G., Li, J., Gong, S., Han, Y., Li, X.: FlexFlow: A flexible dataflow accelerator architecture for convolutional neural networks. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 553–564. IEEE, Piscataway (2017)
14. Zhang, J., et al.: Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators. In: ACM/IEEE DAC, pp. 1–6 (2018)
15. Limbrick, D.B., Mahatme, N.N., Robinson, W.H., Bhuva, B.L.: Reliability-aware synthesis of combinational logic with minimal performance penalty. IEEE Trans. Nucl. Sci. **60**(4), 2776–2781 (2013)

16. Hanif, M.A., Khalid, F., Putra, R.V.W., Rehman, S., Shafique, M.: Robust machine learning systems: Reliability and security for deep neural networks. In: 2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS), pp. 257–260. IEEE, Piscataway (2018)
17. Gebregiorgis, A., Kiamehr, S., Tahoori, M.B.: Error propagation aware timing relaxation for approximate near threshold computing. In: Proceedings of the 54th Annual Design Automation Conference 2017, p. 77. ACM, New York (2017)
18. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
19. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27, pp. 2654–2662. Curran Associates (2014). http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf
20. Zhang, J.J., et al.: Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In: IEEE VTS, pp. 1–6. IEEE, Piscataway (2018)
21. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv preprint arXiv:1207.0580
22. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using DropConnect. In: International Conference on Machine Learning, pp. 1058–1066 (2013)
23. Hanif, M.A., Hafiz, R., Shafique, M.: Error resilience analysis for systematically employing approximate computing in convolutional neural networks. In: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 913–916. IEEE, Piscataway (2018)
24. Reagen, B., Gupta, U., Pentecost, L., Whatmough, P., Lee, S.K., Mulholland, N., Brooks, D., Wei, G.Y.: Ares: A framework for quantifying the resilience of deep neural networks. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, Piscataway (2018)
25. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding (2015). arXiv preprint arXiv:1510.00149
26. Hanif, M., et al.: SalvageDNN: salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping. Philos. Trans. R. Soc. A **378**(2164) (2020)
27. Chen, Z., et al.: Ranger: Boosting error resilience of deep neural networks through range restriction (2020). arXiv preprint arXiv:2003.13874
28. Azizimazreah, A., Gu, Y., Gu, X., Chen, L.: Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs. In: 2018 IEEE International Conference on Networking, Architecture and Storage (NAS), pp. 1–10 (2018). https://doi.org/10.1109/NAS.2018.8515692
29. Hoang, L.H., Hanif, M.A., Shafique, M.: FT-ClipAct: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation. In: 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1241–1246. IEEE, Piscataway (2020)
30. Zhao, K., Di, S., Li, S., Liang, X., Zhai, Y., Chen, J., Ouyang, K., Cappello, F., Chen, Z.: FT-CNN: Algorithm-based fault tolerance for convolutional neural networks. IEEE Trans. Parallel Distrib. Syst. **32**(7), 1677–1689 (2020)